

MINISTRY SCIENCES AND HIGHER EDUCATION RUSSIAN FEDERATION
FEDERAL STATE AUTONOMOUS EDUCATIONAL INSTITUTION OF HIGHER EDUCATION
"NORTH CAUCASIAN FEDERAL UNIVERSITY"

METHODOLOGICAL INSTRUCTIONS
By execution laboratory works
By discipline "Python Program Designing" for
students in the program
11.03.02 Infocommunication technologies and communication systems
Profile
"Intelligent Data Processing in Infocommunication Systems and Networks "

Content

Introduction	4
Order execution Works	5
Laboratory Job No. 1: "Linear programs"	8
Tasks To laboratory work No. 1 "Linear programs"	13
Practical Work No. 2: "Branching Computing Processes", Task 1	16
Assignment for Practical Work #2: "Branching Computing Processes."	
Assignment 1	23
Practical Work #2: "Branching Computing Processes." Task 2	31
Assignment for practical work No. 2 "Branching computational processes".	
Exercise 2	34
Practical Job No. 3: "Organization cycles." Exercise 1	44
Tasks To practical work No. 3 "Organization cycles." Exercise 1	47
Practical Job No. 3: "Organization cycles." Exercise 2	48
Tasks To practical work No. 3 "Organization cycles." Task	50
Practical Job No. 3: "Organization cycles." Exercise 3	51
Exercise To practical work No. 3 "Organization cycles". Exercise 3	54
Practical Job No. 4: "One-dimensional arrays"	57
Exercise To practical work No. 4 "One-dimensional arrays"	62
Practical Job No. 5: "Two-dimensional arrays And functions"	68
Exercise To practical work No. 5 "Two-dimensional arrays And functions"	76
Practical Job No. 6: Files	83
Exercise To practical work No. 6 Files	94
Practical Job No. 7: "Programming "graphs". Module turtle	95
Assignment for Practical Work #7 "Graphics Programming." Turtle Module ...	114
Practical Job No. 8: GUI, module Tkinter»	115
Exercise To practical work No. 8 GUI, module Tkinter»	135
Conclusion	139
APPLICATION 1	140
APPLICATION 2	142
List literature	143

Introduction

1. Target And tasks development disciplines (module)

The goal of mastering the course: "Programming in Python" is to develop a set of competencies in programming on various platforms.

Objectives of mastering the course: to study programming in modern information systems environments: creation of modular programs, elements of the theory of modular programming, to study the features of the formation of visual interfaces for various types of operating systems

2. Place disciplines (module) V structure educational programs

The "Python Programming" course is a required course in Block 1 of the Discipline (modules). It is completed in the third semester.

3. Scroll planned results training By discipline (module), correlated with the planned results of mastering the educational program

Code, formulation of competence	Code, formulation indicator	Planned learning outcomes for the discipline (module), characterizing stages of competence development, indicators
PC-1 Capable of developing requirements and design software	ID-1PK-1 understands the principles of software architecture and types of software architecture ; understands the methods and tools for software design provision, methods and means design bases data; understands the methods and tools for designing software interfaces; understands the methods and techniques for formalizing problems; understands the capabilities of modern and advanced software development tools and technical means.	understands the principles construction of software architecture and types software architecture ; understands design methods and tools software, methods and tools for designing databases; understands the methods and tools for designing software interfaces; understands the methods and techniques for formalizing problems; understands the capabilities of modern And promising development tools software products, technical means.
	ID-2PK-1 analyzes the fulfillment of requirements; develops options for implementing requirements; evaluates and justifies recommended solutions; selects means for implementing software requirements ; uses existing standard solutions and software design patterns.	Conducts an analysis of the fulfillment of requirements; develops options for implementing requirements; conducts an assessment And justification recommended solutions; selects the means of implementing software requirements; uses existing standard solutions And templates design software .

	ID-3PK-1 evaluates the time and labor intensity of implementing software requirements; applies methods and tools for designing software, data structures, databases, and software interfaces.	carries out an assessment of the time and labor intensity of implementing software requirements ; applies methods and design tools software , structures data, databases, programming interfaces.
PC-2. Capable of carrying out conceptual, functional and logical systems design average And large scale and complexity.	ID-5PK-2 Determines architecture of programming environments.	defines the architecture programming environments
	ID-6PK-2 Applies methods and algorithms for optimizing executable code.	applies methods and algorithms for optimizing executable code.
	ID-7PK-2 Uses modern software development methodologies and programming technologies.	uses modern software development methodologies provision and technology programming.
	ID-8PK-2 Produces software implementation options .	develops software implementation options .

Order execution work

1. Carefully read and understand the conditions of the problem to be solved.
2. Familiarize yourself with the necessary theoretical material using the literature listed in the manual. It is recommended that books [1] and [2] (see bibliography) be used as the primary source of theoretical and practical material.
3. Develop an algorithm for solving the problem. Clarify the sequence of its steps.
4. Study the examples provided in the literature, including this textbook. If necessary, run them on a computer, and then use the fragments to write your own solution.
5. Prepare your own solution and debug it using a computer.
6. ***Prepare report*** .

Form report

Each report is prepared in the form of an explanatory note and must contain the following elements:

- title sheet;
- text explanatory notes V typewritten or handwritten view;
- list used literature or websites Internet;
- A Python program listing, with the program output as an appendix. A fragment of the program output (not the complete solution) is acceptable.

An example of how to format a report on practical work can be found in the material for practical work No. 2, task 1.

Requirements To design

These requirements apply to typewritten reports submitted using Word. When formatting reports using other software, use modes that most closely match these requirements. Handwritten reports must be neatly formatted and legible.

Graphic images (drawings, flowcharts) can be designed in pencil. If you have the skills, graphic images can be designed in Word or using Excel, Paint, or Visio.

A4 paper . The main text, except for the title page, should be written in **Times New Roman**, size **12** , single- or one-and-a-half-spaced.

The following margin sizes should be used in the page settings :

- Indent above - 1.5;
- Indent from below - 2.5;
- Indent left - 2.5;
- Indent on the right - 1.0.

The program listing is formatted in a fixed-width font. **Courier New** size **10-12**.

Long instructions Can write down on several lines. This can be done in two ways:

- V at the end lines mark symbol reverse slash - \ . For The symbol must be followed by a line feed character (Enter). This is an obsolete form;
- place expression V round brackets (recommended) or, When defining a list or dictionary - square and curly brackets.

Details Can find V examples To laboratory works.

Content explanatory notes

1. Production tasks.
2. Brief theoretical intelligence about features operators and methods used (theoretical introduction).
3. Description programs:

- general information (programming language, operating system, processor type);
- description logical structures programs;
- description algorithm solutions tasks (V view block diagrams);
- description input And weekend data programs;
- description subroutines;
- test examples, list of abnormal and acceptable values of input data.

Scroll laboratory works

While covering the topic "High-Level Programming in Python," students complete the following lab assignments:

Subject: Structural programming

1. Linear programs.
2. Branching computing processes: Exercise 1, Exercise 2.
3. Organization cycles: Task 1, Exercise 2, Task 3.
4. One-dimensional arrays.
5. Two-dimensional arrays and functions.
6. Files.
7. Programming graphics. Module turtle.
8. GUI, module Tkinter.

Choice options

Practical work may have several tasks, such as laboratory work. work No. 2 And No. 3, And set options. Student for each practical work (task), he/she completes his/her own version, which is associated with the record book number:

- divide the last two digits of the record book **nm** by the number of options in the task **z** .

$$N = nm \% z + 1 ,$$

where % is the receipt remainder for integer division, and N – task option number .

Example: There are 24 options in the practical assignment. The final numbers in the record books of students A and B are 30 and 47, respectively.

Student A performs exercise $30 \% 24 + 1 = 7$, A student B: $47 \% 24 + 1 = 24$.

Assignments for practical work may be modified by the instructor. Current assignments can be obtained electronically from the department or from the instructor. The electronic version's length The size of laboratory work in pdf format is about 4 MB.

The student, by agreement with the teacher, can choose another available number of practical work.

Practical Job No. 1: "Linear programs"

Objective of the work:

To give students practical experience in preparing a simple program and in writing mathematical expressions in the Python programming language.

Production tasks

Write a program to calculate using the given formulas. Prepare test examples in advance using a calculator. or an Excel spreadsheet.

$$1) y = \lg^2 \left(\frac{x^2}{2} - 1 \right) + \frac{2 \cos(x - \pi/6)}{1/2 + \sin \alpha}; \quad 2) y = 2 \frac{\log(3 + \sin(x)) (3 - \cos(\pi/4 + 2x))}{1 + \lg^2(2x/\pi)}.$$

Theoretical introduction

When calculating such expressions, it's necessary to analyze the range of valid values for the arguments used in the expression. For example, the denominator of a fraction may be zero, causing the program to terminate due to a division-by-zero error. It's also necessary to consider the valid range of arguments for the functions used. For example, the base of the logarithm must be greater than zero and not equal to one, and the function being logarithmed must be greater than zero.

Care should be taken when handling expressions that, for example, involve taking a square root or, in general, raising a value to a power whose exponent is not an integer. This could result in a complex number or cause an error that will cause the program to terminate.

In our examples the denominators are $1/2 + \sin^2 \alpha$ and $1 + \lg^2(2x/\pi)$ always unequal zero. Except this, in second example, base logarithm $3 + \sin(x)$ Not negatively And Not equals unit, A logarithmizable function $3 - \cos(\pi/4 + 2x)$ Always more zero.

For mathematical calculations in Python there are as built-in, as well as additional functions and methods. Additional mathematical functions can be applied after connecting the math module :

```
import math
```

or

```
from math import *
```

IN first case functions are being considered How methods object math And should sign up So :

```
import math
print(math.sin(math.pi/4))
```

```
print(math.sqrt(2)/2)
```

In the second case, the function call can be made in a form more familiar to us:

```
from math import *  
print(sin(pi/4))  
print(sqrt(2)/2)
```

However, this method of importing can disrupt the program's namespace, as it can create a conflict between the names of variables used by the programmer and the names of imported functions. When importing, you can limit the import to only the necessary functions, for example:

```
from math import (pi, sin, cos,  
                  tan, log)
```

This example demonstrates how to import the necessary functions and how to place the instructions on multiple lines. These functions can also be used in the usual manner.

The set of functions and methods of the `math` module is given in Table 1, see Appendix 1. More information about the functions of the `math` module can be obtained from the documentation or on the Internet.

In cases where the required function is missing in the programming language, it can be written or calculated using known formulas. For example,

$$\text{ctg}(x) = 1/\tan(x) = \cos(x)/\sin(x) .$$

Variable names should be chosen carefully and either use symbols accepted in mathematics or physics, or phrases that reflect the purpose of the variable.

For example, the symbol α can be replaced with `a` or `alpha` . The color value can be stored in the `Color` variable , and the volume in the `Volume` variable . or `Capacity` .

Please contact attention on color coloring book variable at set `V` editor IDLE. She should be black.

To solve the problem you will need to input and output data. In our case, these are integer or real numbers.

Data entry

Input data Can execute With keyboards function `input()` : `m = input([str])`

At this on the screen will display the string `str` , and variable `m` will receive the string value entered by the user. The string type can be converted, for example, to an `int` or `float` if the entered value is a number.

To enter multiple values, you can use the `split()` method , which allows you to split a string into substrings (*split* – to split).

For example, to enter the values of the parameter α and the variable x can be done like this:

```
a, x = input(' Enter data (a, x): ').split() a
= float(a)
x = float(x)
```

The separator used is specified as a parameter to the `split()` method . If no separator is specified, a space is used. When entering a real number, the integer part is separated from the decimal part by a period.

If the user doesn't enter any data (simply pressing Enter) or enters invalid characters, such as letters, instead of numbers and periods, the program will terminate abnormally at the `float()` casting step . The resulting exception can be handled using the `try` statement . For more information, see, for example, [1].

In the following example, the user must enter the first number as an integer and the second as a float. If the input is incorrect, an error will occur. exceptional situation `ValueError` And control The program will pass the data to the `except` block . This block can be used to anticipate potential situations and make the necessary decisions, such as forcing the user to enter data correctly.

```
import sys, traceback
while True:
    a = x = ""
    try:
        a,x= (input('Введите [a: int, x: float]:')
              .split())
        a = int(a)
        x = float(x)
        break
    except ValueError:
        if a == "" and x == "":
            a = x = 0
            continue
        print("a - int, x - float. Пример: 3 4.5")
print(a, x,)
```

IN body "eternal" cycle `while True:` , V block `try` , two variables are initialized, followed by an instruction for entering data. If a number is entered incorrectly, for example, a float value instead of an integer, or instead of numbers - letter, will arise exceptional

ValueError situation . Control will be passed to the except module , where, in the conditional statement, it is checked whether the data was entered. If it was If Enter is pressed , the variables will be set to zero, and control will be transferred to the statement following the loop. If data entry was made, an exception occurred during data type conversion. In this case, a warning message is issued, and control will be transferred to the beginning of the loop (the entry must be repeated).

Note: If you do not initialize variables before the input() statement, then when an exceptional situation occurs (only Enter is pressed during input), control will be transferred to the except module , where a new exceptional situation will occur in the if conditional statement : variable a not defined.

Data output

Data can be output to the monitor screen using the print() function . This function allows for formatted output, using both C-like formatting and the Python format string.

Next lines demonstrate, How Can format conclusion.

```
for x in range(1,11):
    print('%2d %3d %7.2f' % (x, x*x, x*x*x))
    print("{0:.2f} {1:2f} {2:.4f}".format(a, x, y))
```

Letter in the number format determines the type of number to be output. So, d – is an integer type, f – a real number. The number in the format indicates the number of digits that will be used to display the number. For a real number, the number of decimal places to display is indicated after the period.

The second line uses a C-like format, in which the number format begins with a percent " % ". In this format, the arguments are separated from the format portion of the string by the % character. – percentage. The third line uses a Python format string, in which the positions of argument values in the format string are separated by curly braces.

Note that the format strings themselves begin and end with either a single or double quote. Python accepts both types of quotes for string delimitation. It's important that the beginning and end be the same.

It should also be understood that there may be other symbols or words in the spaces between formatting symbols:

```
print('x=%2d x^2=%3d x^3=%7.2f'
      % (x, x*x, x*x*x))
print("a={0:.2f} x={1:.2f} y={2:.4f}"
      .format(a, x, y))
```

Usage formatted lines does conclusion data more intelligible.
For more detailed information please contact us To textbooks or Internet.

Solution tasks

Let's return to our examples and write them using Python's rules. The first expression will look like this:

1. $y = \tan(x^2/2 - 1)^2 + 2 \cos(x - \pi/6) / (1/2 + \sin(a)^2)$

Second expression let's imagine V view two:

2.
$$\begin{aligned} \text{tmp} &= \log(3 - \cos(\pi/4 + 2x), \\ &\quad 3 + \sin(x)) / (1 + \tan(2x/\pi)^2) \\ y &= \text{pow}(2, \text{tmp}) \end{aligned}$$

Description algorithm

For calculations must provide the input of two variables x and a . Since their type and representation precision are not specified in the problem, we will choose a real type (`float`) for them. To optimize the expression, we will use the intermediate variable `tmp`.

1. Enter values a And x , transform To type `float`.
2. Calculate expression 1.
3. Withdraw result calculations.
4. Calculate meaning variable `tmp`;
5. Calculate expression 2.
6. Withdraw result calculations.

Listing programs

```
# -*- coding: cp1251 -*-
from math import *
a = float(input('Enter parameter A: '))
x = float(input('Enter meaning x: '))
y=tan(x**2/2-1)**2+(2*cos(x-pi/6))/(1/2+sin(a)**2)
print("{0:.2f} {1:.2f} {2:.4f}".format(a, x, y))
tmp=log(3-cos(pi/4+2*x),
        3+sin(x))/(1+tan(2*x/pi)**2)
y=pow(2,tmp)
print("{0:.2f} {1:.4f}".format(x, y))
```

Results testing programs

a	x	First expression		Second expression
		Calculator	Program	
- 2	- 2	1.196954	1.1970	1.1184
0	- 2	-0.834654	-0.8347	1.1184
0	0	5.889618	5.8896	1.6880
2	0	3.730931	3.7309	1.6880
1.5	0.5	2.771242	2.7712	1.7955
4	3	-1.326566	-1.3266	1.0517

Note: This table was being drawn up V text manual editor.

Tasks To practical work No. 1 Linear programs

Write a program to calculate using two formulas. Prepare at least five test cases. Perform the calculations first using a calculator or an office application, such as Excel. or Calc. The calculation results for both formulas must match. Express functions missing from the language using existing ones.

1. $z_1 = 2 \sin^2(3\pi - 2\langle) \cdot \cos^2(5\pi + 2\langle)$; $With = \frac{1}{4} - \frac{1}{4} \sin(\frac{5}{2}\pi - 8\alpha)$.
2. $fr = \cos \alpha + its \alpha + \cos 3 \alpha + its 3 \alpha$ $z_2 = \sqrt{2} \cos \alpha \cdot its(\frac{\pi}{4} + 2 \alpha)$.
 o ;
 m
 1
3. $z_1 = \frac{its 2 \alpha + its 5 \alpha - his 3 \alpha}{\cos \alpha + 1 - 2 \sin^2 2 \alpha}$ $z_2 = 2 \sin \alpha$.
4. $z_1 = \frac{2 \cdot \cos \alpha \cdot his 2 \alpha - its \alpha}{\cos \alpha - 2 \cdot without \alpha}$ $z_2 = \operatorname{tg} 3 \alpha$.
5. $z_1 = 1 - \frac{1}{4} \cdot without 2 \alpha$ $z = \cos^2 \alpha + \cos^4 \alpha$.
 $= 1 - \frac{1}{4} without^2 2 \alpha + \cos$
 2α ;
 1 $\frac{1}{4}$ 2
6. $z_1 = \cos \alpha + \cos 2 \alpha + \cos 6 \alpha + \cos 7 \alpha$;
 $z = 4 \cos \frac{\alpha}{5} \cdot \cos \frac{5}{2} \alpha \cdot \cos 4 \alpha$.
7. $With = \cos^2(3\pi - \frac{\alpha}{4}) - \cos^2(11\pi + \frac{\alpha}{4})$ $with \neq \frac{2}{2} \sin \alpha$
 1 $\left(\frac{3}{8} \pi - \frac{\alpha}{4} \right)$ $\left(\frac{11}{8} \pi + \frac{\alpha}{4} \right)$; 2 $\frac{2}{2}$ $\frac{2}{2}$.
8. $z_1 = \cos^4 x + without^2 and +^1$ $z_2 = \sin(y + x) \cdot \sin(y - x)$.
 $without^2 2 x - 1$; $\frac{1}{4}$
9. $from_1 = (\cos \alpha - something \beta)^2 - (\sin \alpha - \sin \beta)^2$;
 $With = -4 \sin^2 \frac{\alpha - \beta}{2} \cdot \cos(\alpha + \beta)$.
10. $z_1 = \frac{\sin\left(\frac{\pi}{2} + 3\alpha\right)}{1 - \sin(3\alpha - \pi)}$; $z_2 = \operatorname{ctg}\left(\frac{5}{4}\pi + \frac{3}{2}\alpha\right)$.
 $)$
 $1 - 2 \sin^2 \alpha$ $z = \frac{1 - \tan \langle}{1 - 2 \sin^2 \alpha}$
11. $z_1 =$

13.
$$z = \frac{\text{without } \alpha + \cos(2\beta)}{\cos \alpha - \text{without}(2\beta - \alpha)};$$

$$z_2 = \frac{1 + \sin 2\beta}{\cos 2\beta}.$$

$$14. \quad z_1 = \frac{\cos \alpha + \sin \alpha}{\cos \alpha - \sin \alpha}; \quad z_2 = \operatorname{tg} 2\alpha + \sec 2\alpha.$$

$$15. \quad z_1 = \frac{\sqrt{2b+2\sqrt{b^2-4}}}{\sqrt{b^2-4}+b+2}; \quad z_2 = \frac{1}{\sqrt{b+2}}.$$

$$16. \quad z_1 = \frac{x^2+2x-3+(x+\sqrt{x^2-9})}{x^2-2x-3+(x-1)\sqrt{x^2-9}}; \quad z_2 = \sqrt{\frac{x+3}{x-3}}.$$

$$17. \quad z_1 = \frac{\sqrt{(3m+2)^2-24m}}{2\sqrt{m}}; \quad z_2 = \sqrt{m}.$$

$$18. \quad z_1 = \left(\frac{a+2}{\sqrt{2a}} - \frac{a}{\sqrt{2a}+2} + \frac{2}{a-\sqrt{2a}} \right) \cdot \frac{\sqrt{a} \sqrt{2}}{a+2}; \quad z_2 = \frac{1}{\sqrt{a}+\sqrt{2}}.$$

$$19. \quad z_1 = \frac{(1+a+2-1-a+a^2) \cdot (5-2a)}{a^2 \left(\frac{2a+a^2}{2a-a^2} \right)}; \quad z_2 = \frac{4-a^2}{2}.$$

$$20. \quad z_1 = \frac{(m-1)\sqrt{m}-(n-1)\sqrt{n}}{\sqrt{m^3n+nm+m^2-m}}; \quad z_2 = \frac{\sqrt{m}-\sqrt{n}}{m}.$$

$$21. \quad z_1 = \frac{1+\sin^4(-a)-\cos^4(-a)}{a}; \quad z_2 = 1-2\operatorname{tg}^2 a.$$

$$22. \quad z_1 = \sqrt{\frac{1-\sin(\frac{\pi}{2}-a)}{2}} + \sqrt{\frac{1+\cos(2\pi-a)}{2}}; \quad \pi < a < \frac{3\pi}{2}.$$

$$z_2 = \frac{\cos \frac{\pi}{2}}{2} - \frac{\cos \frac{3\pi}{2}}{2}; \quad \pi < \frac{\pi}{2} < \frac{3\pi}{2}.$$

$$23. \quad \text{With } z_1 = \frac{1+6ac-1}{a^3-8c^3-a-2c}; \quad z_2 = \frac{1}{a^3-8c^3-a^2+2ac+4c^2};$$

$$z_2 = 1-2c+a.$$

$$24. \quad z_1 = \frac{1}{a-b+c} \cdot \left(1 + \frac{b^2+c^2-a^2}{2BC} \right); \quad z_2 = \frac{a-b-c}{2} \cdot a.$$

$$25. \quad z_1 = \frac{\operatorname{th}^2(\pi + a) + \operatorname{th}^2\left(\frac{\pi}{2} + a\right)}{\cos\left(\frac{3\pi}{2} + a\right)} \operatorname{ctg}(1.5\pi - a); \quad z_2 = \frac{1}{\cos a}.$$

26. $z_1 = \frac{\operatorname{tg}(x - \frac{\pi}{2}) \cos(\frac{3\pi}{2} + x) - \sin^3(3.5\pi - x)}{\cos(x - 0.5\pi) \operatorname{tg}(1.5\pi + x)};$ $z_2 = \sin^2 x.$
27. $z_1 = a^{\sqrt{\log_a b}} - b^{\sqrt{\log_b a}} + \operatorname{tg}(ab + 3 \square / 2)$ $z_2 = \operatorname{tg}(ab + \frac{3}{2}\pi);$
;
28. $z_1 = \frac{\sin^2 a - \operatorname{tg}^2 a}{\cos^2 a - \operatorname{ctg}^2 a};$ $z_2 = \operatorname{tg}^6 a;$
29. $z_1 = \frac{1 - 2\sin^2 a}{2\operatorname{ctg}(\frac{\pi}{2} + a) \cos^2(\frac{\pi}{2} - a)} + e^a;$ $z_2 = 1 + e^a;$
30. $z_1 = \frac{\sin^4 a + 2\sin a \cos a - \cos^4 a}{\operatorname{tg} 2a - 1};$ $z_2 = \cos 2a;$

Practical Job No. 2: Branching Computing Processes, Task 1

Objective of the work:

To provide students with practical experience using conditional branching statements in the Python programming language. This course consists of two tasks.

The following material is presented in the form of a completed practical work.

Practical work #2 Branching computing

processes **Task 1**

Option No. 1

By discipline:
"Programming on Python»

Completed by
Student ____ of the
course group ____
____ Ivano
v I.I.

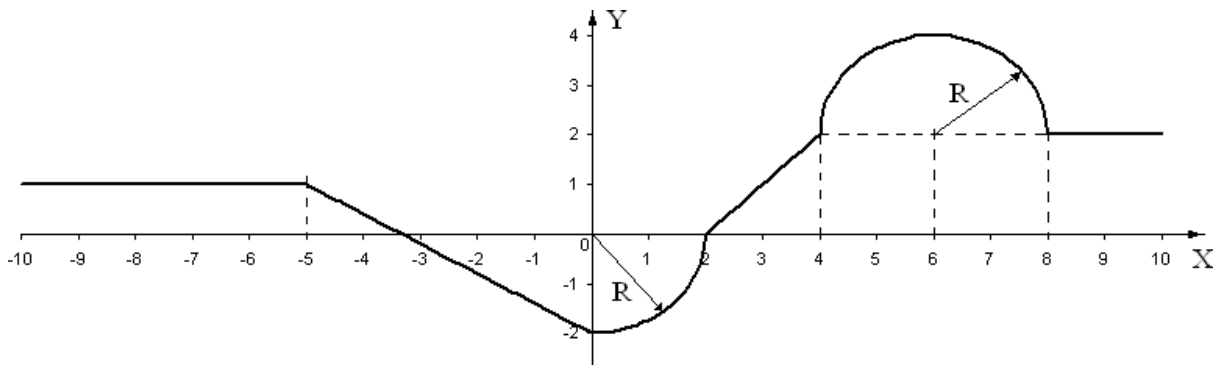
Grade _____

_____. OH.

Shayakhmetov " ____ "
____ 202__
year

Production tasks

Write program, which By introduced meaning argument calculates the value of a function given as a graph.



Theoretical Part

To solve the problem, a branching operator was used, which in Python has the following form:

```
if <Logical expression>:
    <Block - is being carried out, If condition
    true> [ elif <Logical expression>:
        <Block - is being carried out, If condition true>
    ]
[ else:
    <Block - is being carried out, If All conditions
    false>
]
```

A <block> is a set of nested statements that are separated by an equal number of spaces (usually four).

To enter data, use the `input()` statement, which returns a string. Entered values must be converted to a numeric format before being used in arithmetic expressions.

Data output is performed by the `print()` statement, which uses formatted data output.

Schedule functions presented fragments straight lines, described equation $y = kx + b$ and arcs circles. IN in general case equation circle Maybe be presented So: $(x - a)^2 + (y - b)^2 = R^2$.

Unknown parameters, corner tilt And bias direct, A So We will determine the coordinates of the center of the arcs using the data from the graph.

For a straight line on the interval $(-5, 0)$ we can write the following system equations:

$$\begin{cases} 1 = k \cdot (-5) + b \\ -2 = k \cdot 0 + b \end{cases}$$

From second equations follows, What $b = -2$, A from first - $k = -3/5$.

For semicircle With center (6, 2) equation circle will accept view:
 $(x - 6)^2 + (y - 2)^2 = 2^2$

Let's rewrite the equation as follows:

$$(y - 2)^2 = 2^2 - (x - 6)^2$$

$(y - 2)^2 = 4 - (x - 6)^2$ it follows that $y = 2 + \sqrt{4 - (x - 6)^2}$. The sign before the root is chosen for the case when the upper part of the semicircle is considered.

Having completed necessary calculations for all fragments functions, we obtain a system of equations, which we write in the following form:

$$y = \begin{cases} 1 & x < -5 \\ -\frac{3}{5}x - 2 & -5 \leq x < 0 \\ -\sqrt{4 - x^2} & 0 \leq x < 2 \\ \frac{x - 2}{\sqrt{4 - (x - 6)^2}} & 2 \leq x < 4 \\ 2 & 4 \leq x < 8 \\ + \frac{2}{2} & x \geq 8 \end{cases}$$

Function defined on everything It $x \in (-\infty; +\infty)$ At the same time, does not have any special points in the range.

Description programs

The program is written in the algorithmic language Python 3.6, implemented in the environment OS Windows 10 and consists of from the parts responsible for data input, calculation and presentation of data on the monitor screen.

Description algorithm

1. Enter meaning argument x And transform his To type `float` .
2. Determine which interval of the function's domain it belongs to and calculate the value of the function y according to the appropriate formula.
3. Withdraw meaning x And y .

Description input And weekend data

Input data comes from the keyboard and output data is displayed on Monitor for viewing. Input and output data are of type `float` .

Listing programs (simple combination conditional operator)

```
# -*- coding: cp1251 - *-
from math import * # Now Can So :
                        # print(sin(pi/4))
x = float(input(' Enter meaning x='))
if x < -5: y = 1
if x >=-5 and x<0: y = -(3/5)*x-2
if x >= 0 and x<2: y = -sqrt(4-x**2)
if x >= 2 and x<4: y = x-2
```

```

if x >= 4 and x<8: y = 2+sqrt(4-(x-6)**2)
if x >= 8: y = 2
print("X={0:.2f}      Y={1:.2f}".format(x, y))

```

The flow chart of the algorithm for this solution is given in Appendix 1 (Fig. 1) to practical work.

It should be noted that in this algorithm notation, the check is performed for all conditional statements, including those following the one evaluated. For example, if x is -3, then the second statement will be executed, but the comparison operation will also be performed in all subsequent statements. The number of checks can be reduced by writing the program using nested conditional statements.

Listing programs (complicated) combination conditional operator)

```

# -*- coding: cp1251 - *-
from math import * # Now Can So :
                        # print sin(pi/4)
x = float(input('Введите значение x='))
if x < -5:
    y = 1
elif x >=-5 and x<0:
    y = -(3/5)*x-2
elif x >= 0 and x<2:
    y = -sqrt(4-x**2)
elif x >= 2 and x<4:
    y = x-2
elif x >= 4 and x<8:
    y = 2+sqrt(4-(x-6)**2)
else: y = 2
print("X={0:.2f}      Y={1:.2f}".format(x, y))

```

The block diagram of the solution algorithm is given in Appendix 2 (Fig. 2) to the practical work.

Result work programs

```

Enter meaning argument:      - 6
X= - 6.00      Y= 1
Enter meaning argument:      -
                               3.33
X= - 3.33      Y= - 0.00
Enter meaning argument:      6
X= 6.00      Y = 4.00

```

List used literature

1. N.A. Prokhorenok, V.A. Dronov, Python 3 and PyQt 5. Application Development: St. Petersburg: BHV-Petersburg, 2017

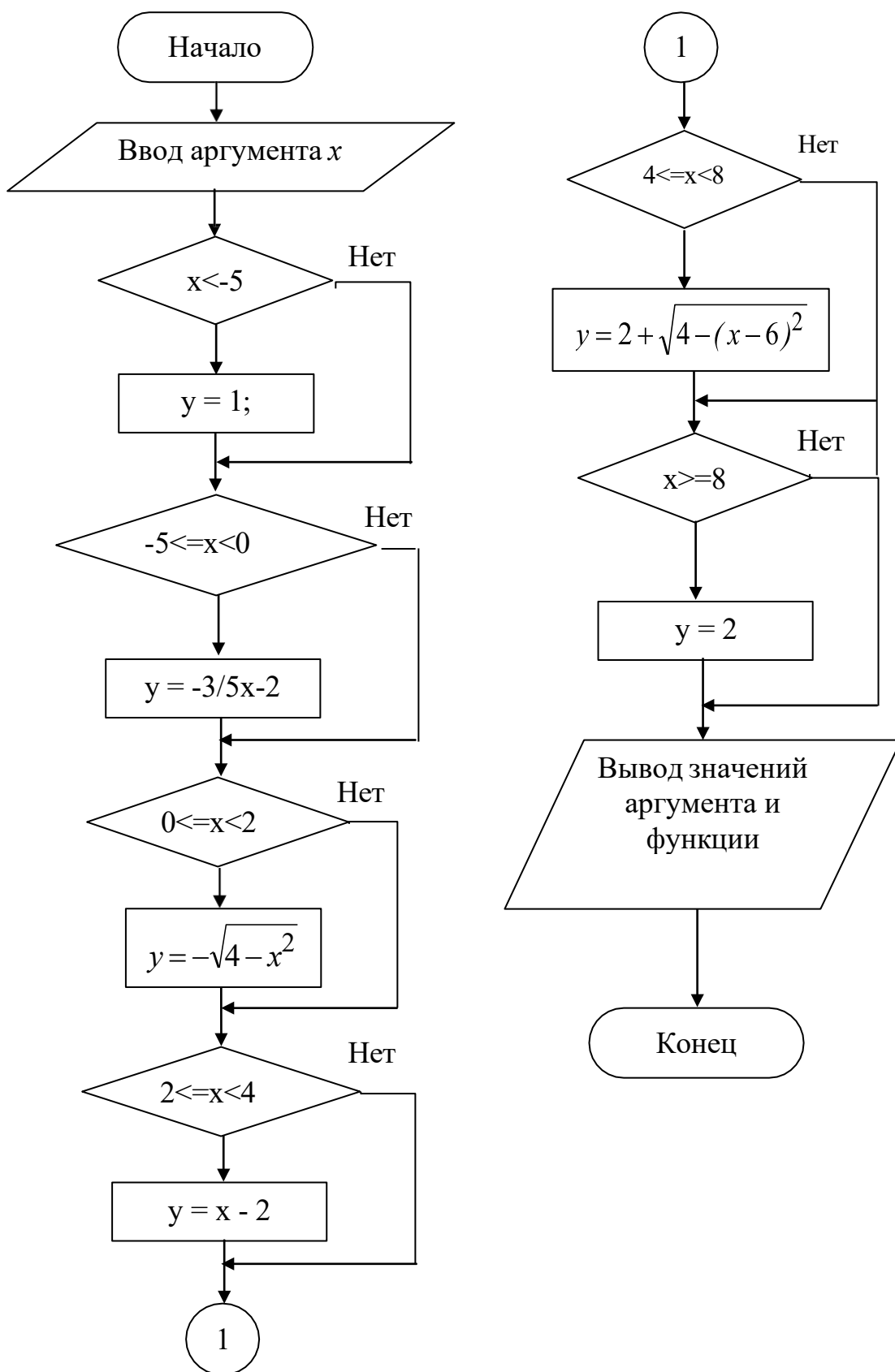


Fig. 1 – Flowchart of the program algorithm (first part)

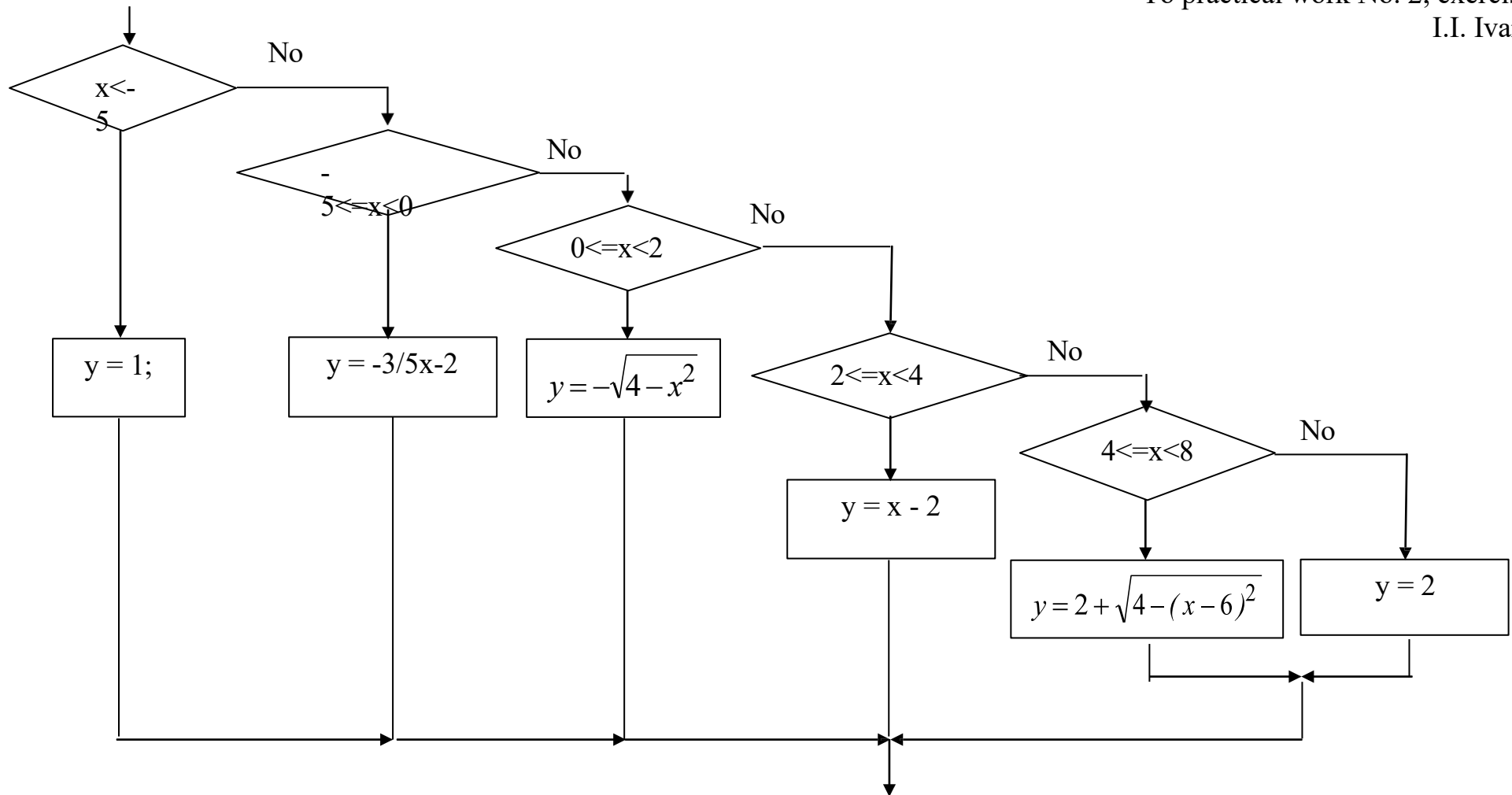


Fig. 2 – Block diagram of the program algorithm (second Part: only the logical part is shown)

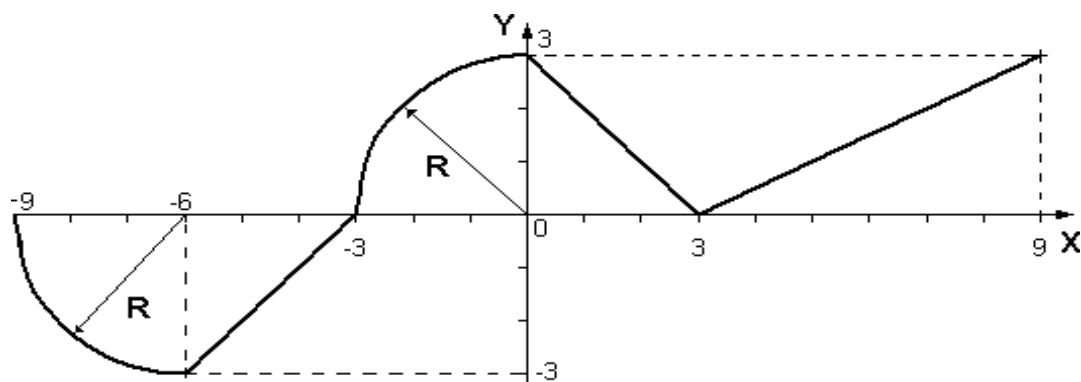
Exercise To practical work No. 2

"Branching computing processes."

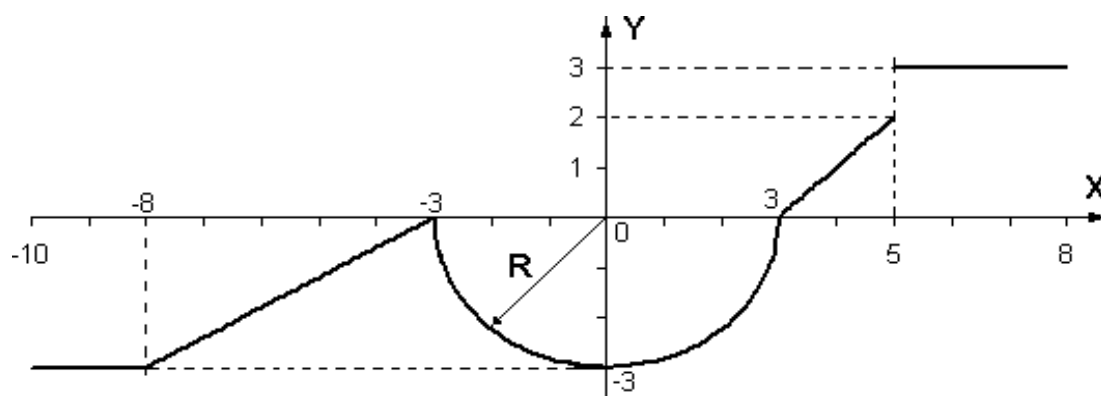
Exercise 1

Write a program that, given the input argument value, calculates the value of a function specified in the form of a graph. The parameters necessary for solving the problem should be obtained from the graph and define in the program, according to your option.

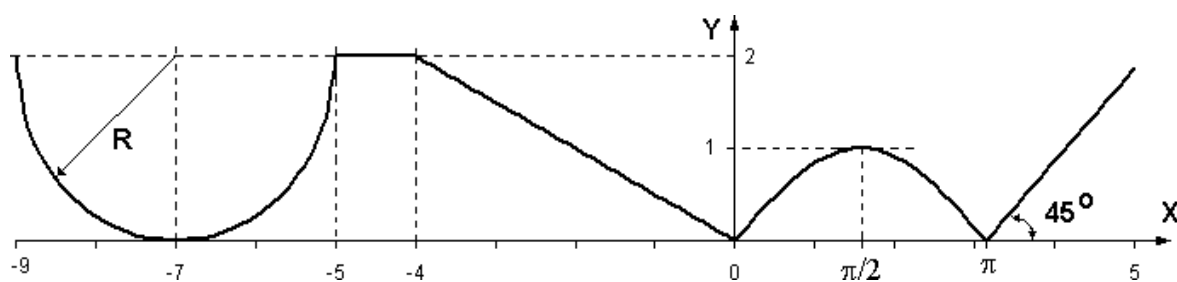
1)



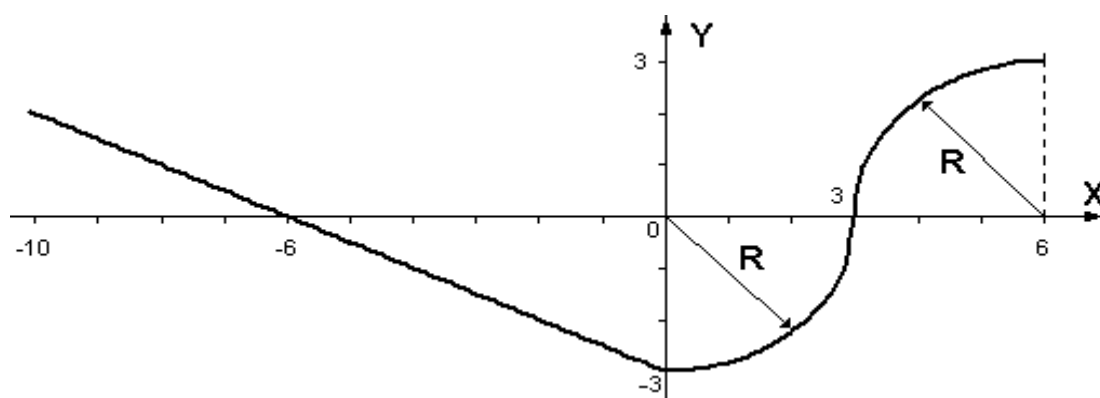
2)



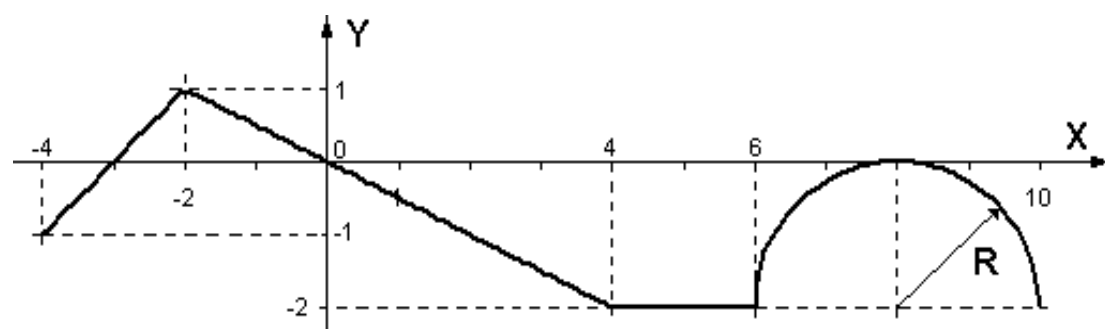
3)



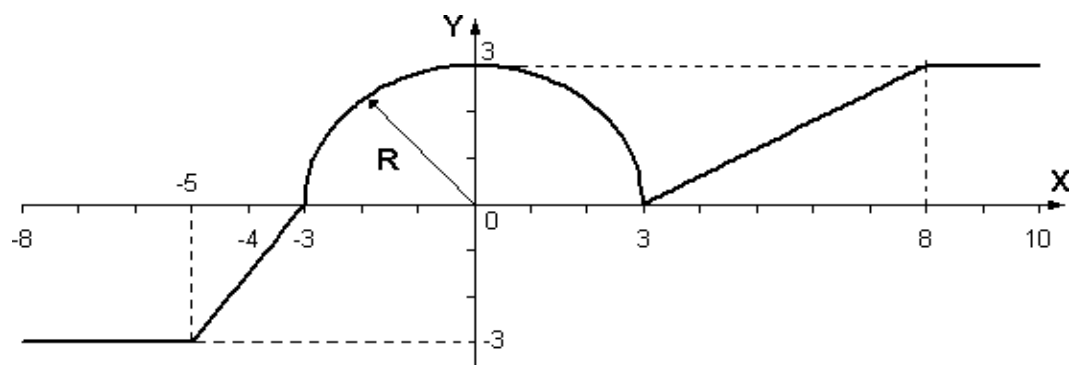
4)



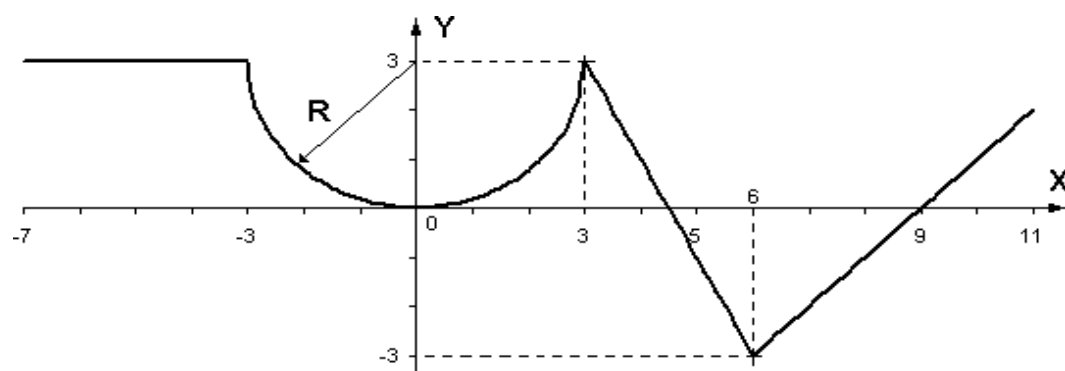
5)



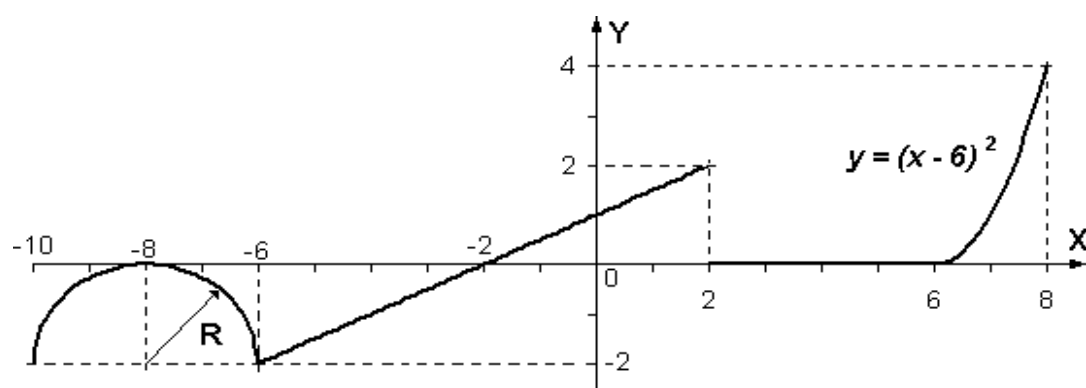
6)



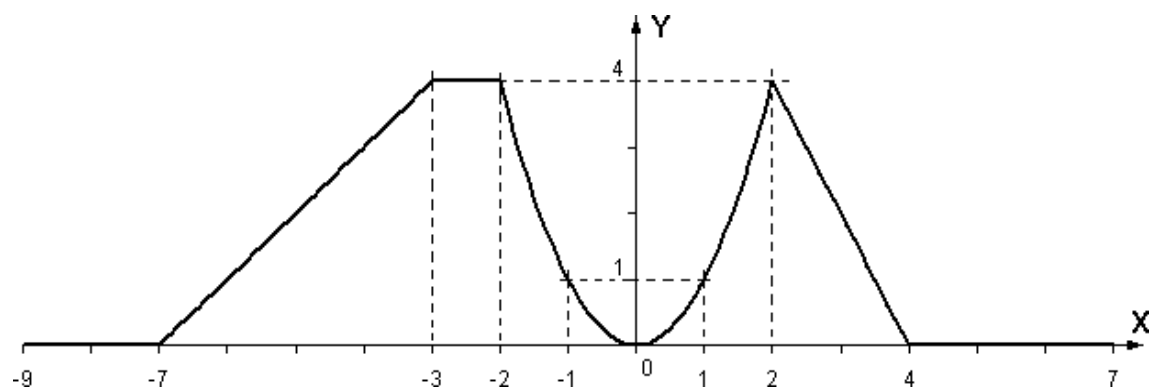
7)



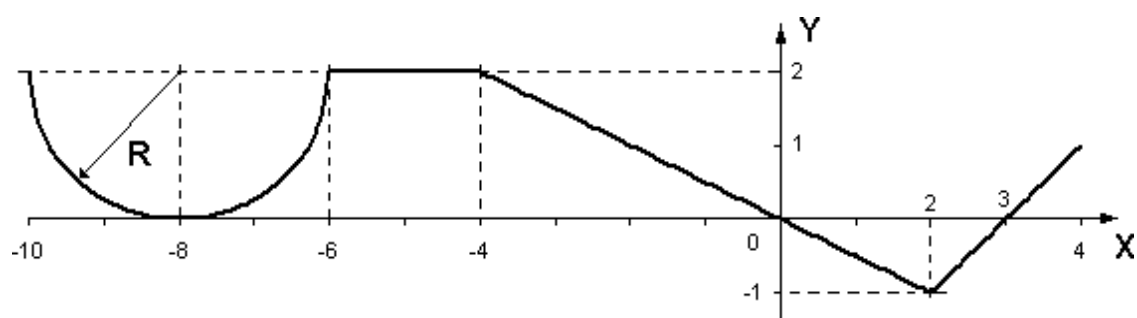
8)



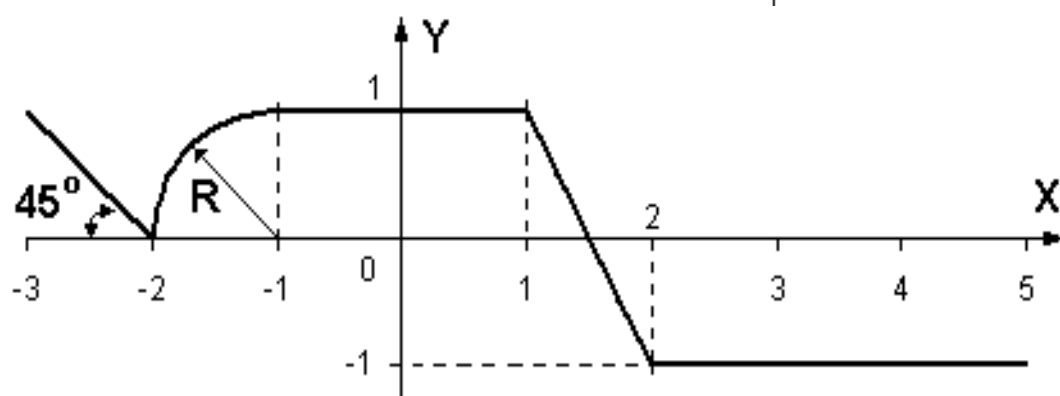
9)



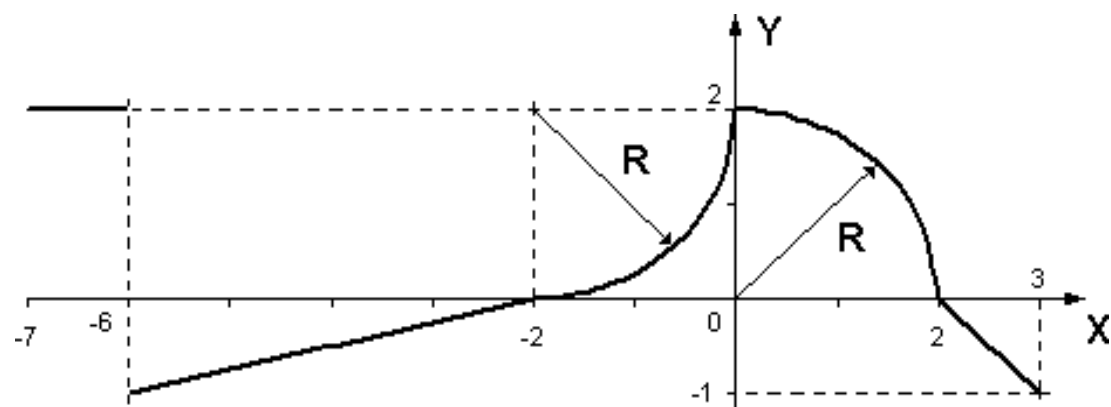
10)



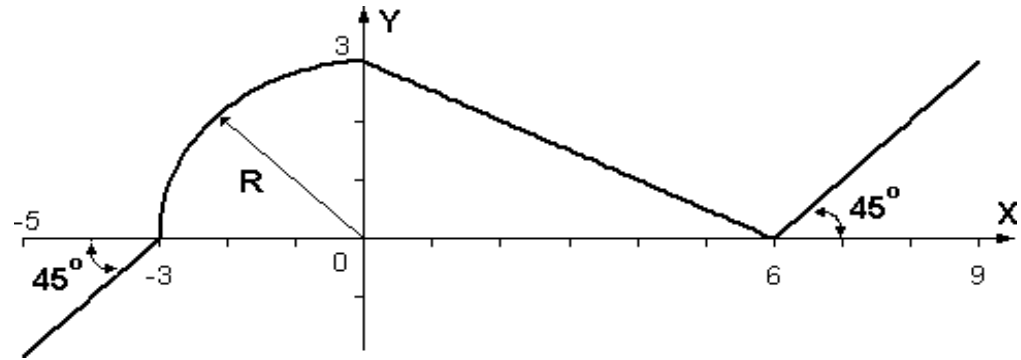
11)



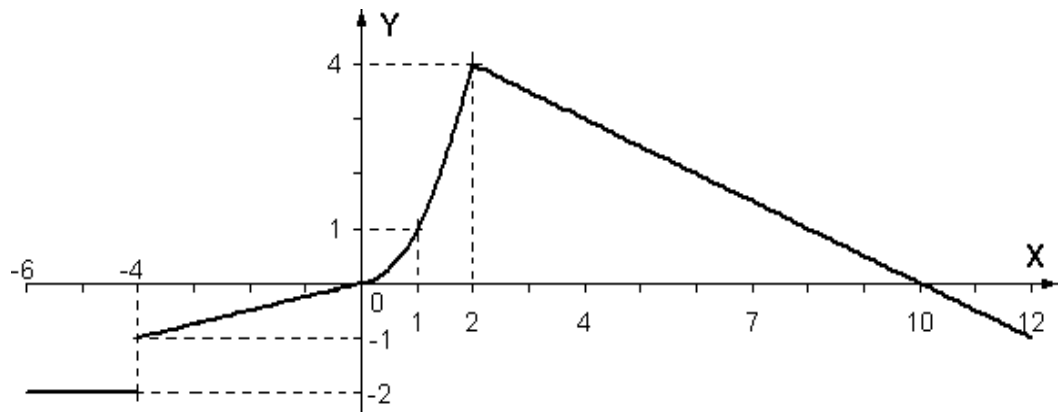
12)



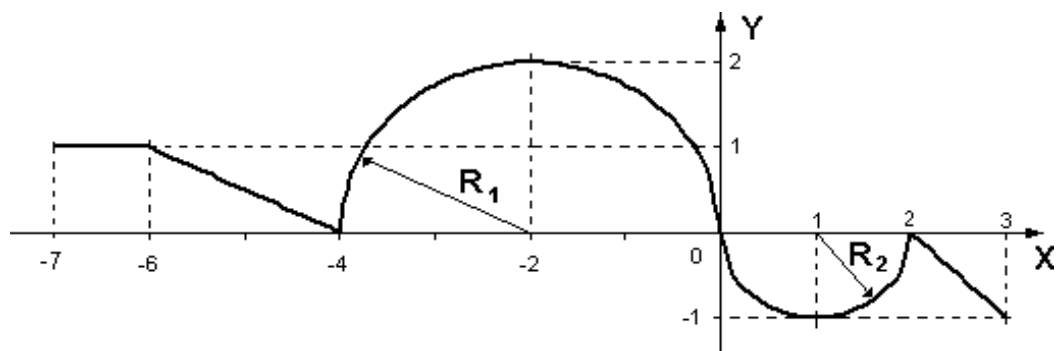
13)



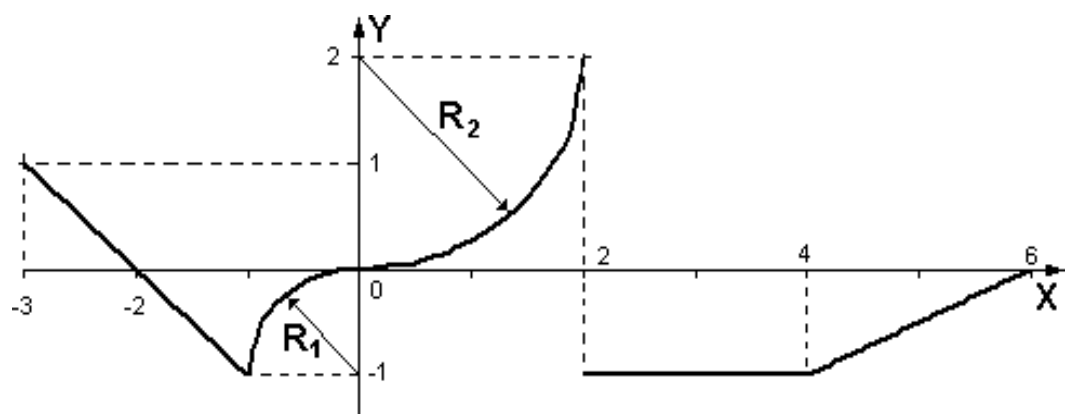
14)



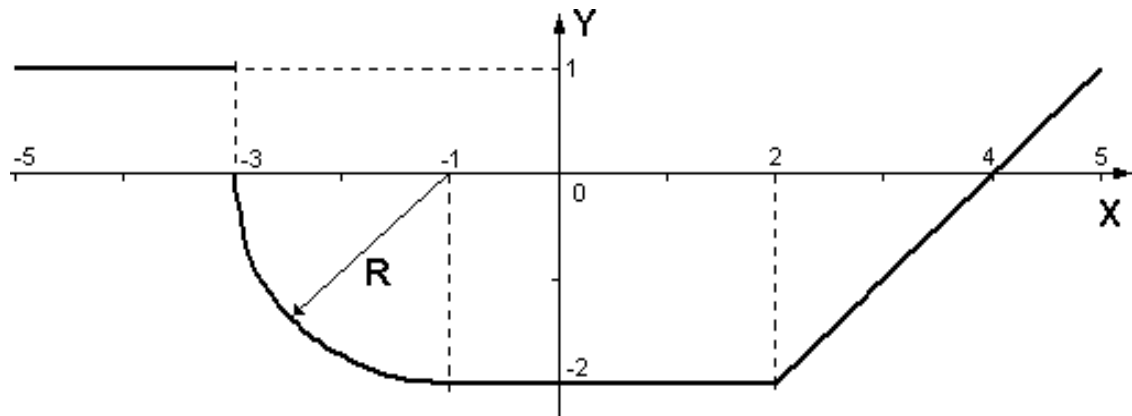
15)



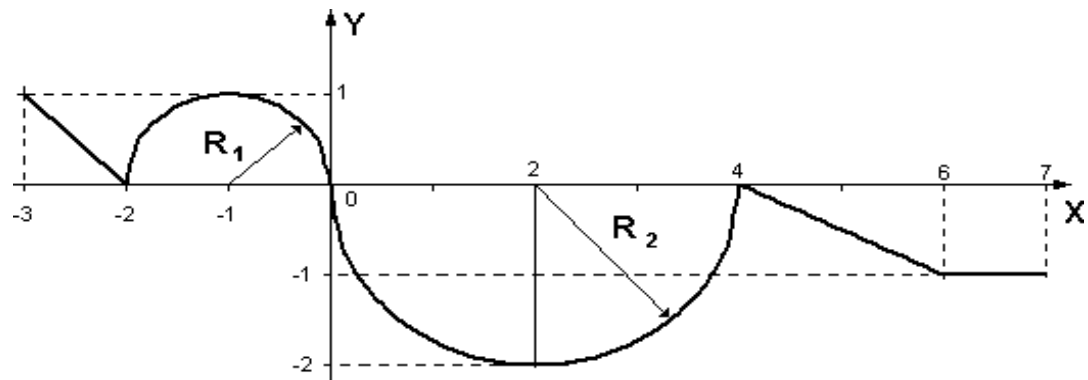
16)



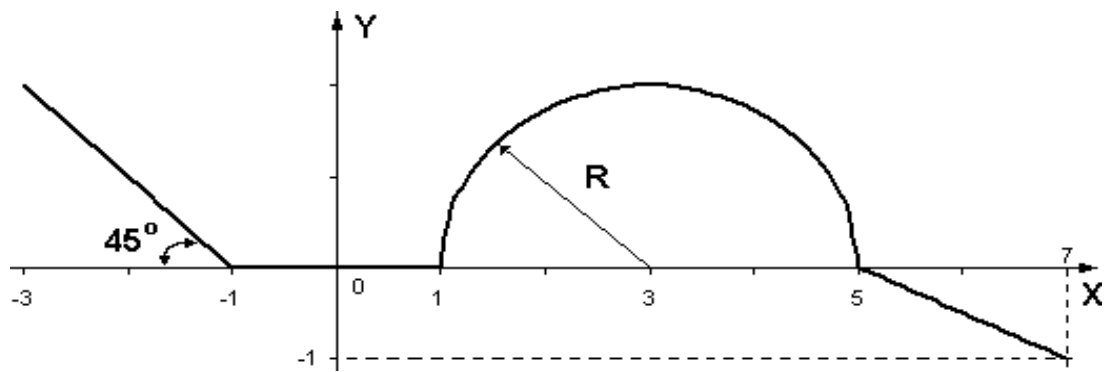
17)



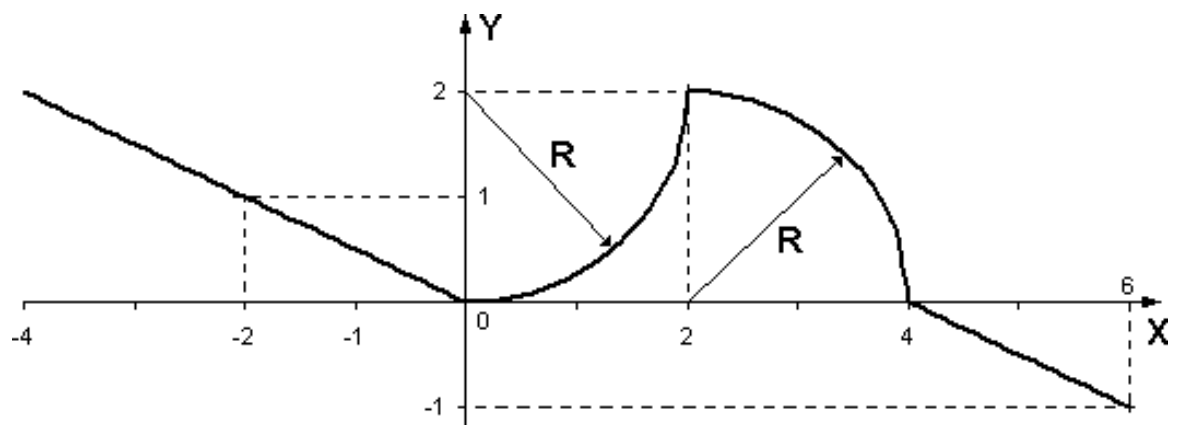
18)



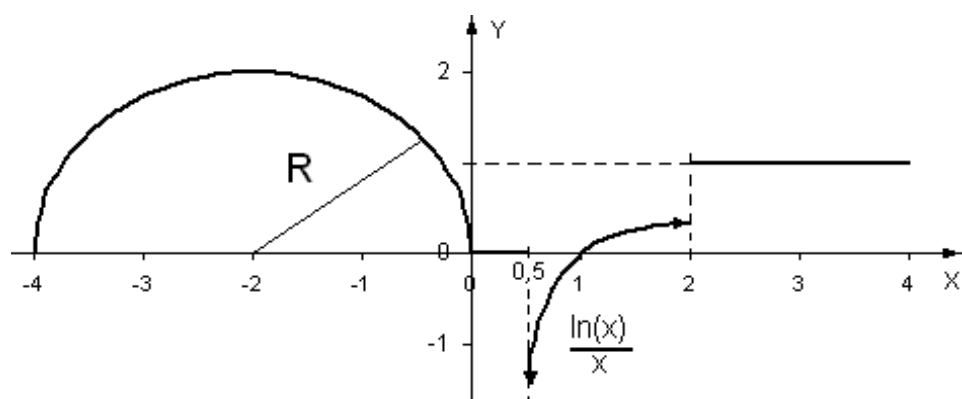
19)



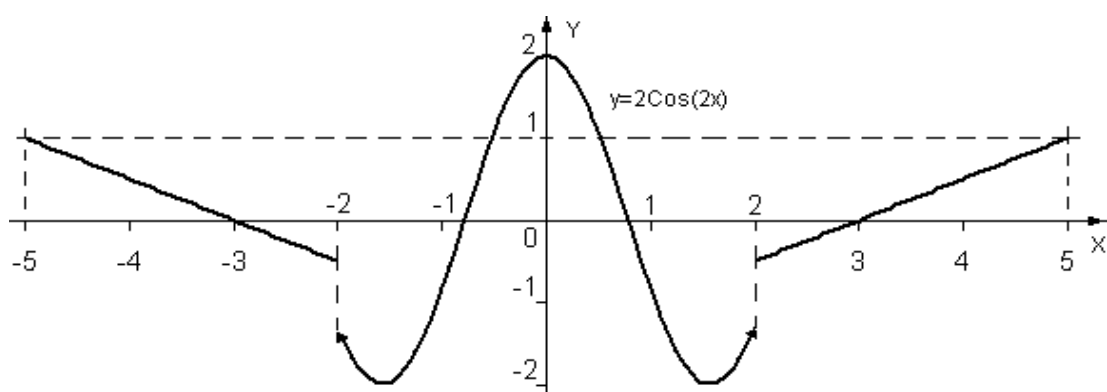
20)



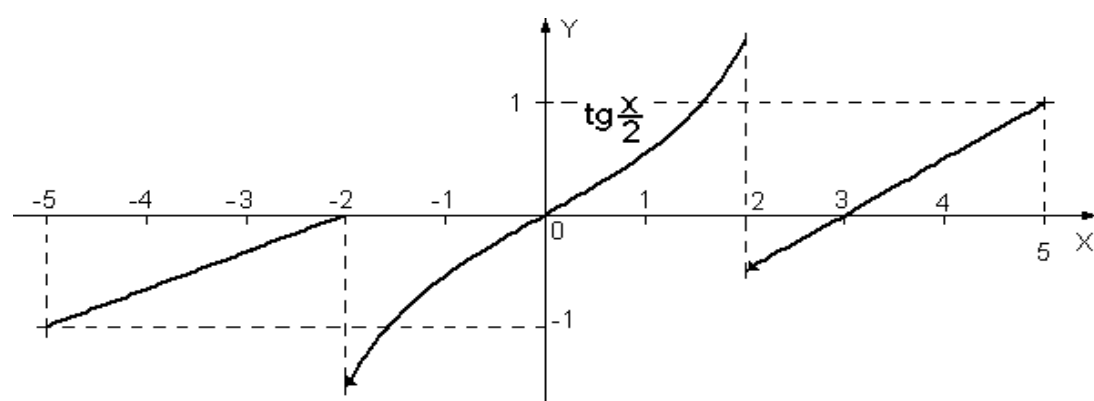
21)



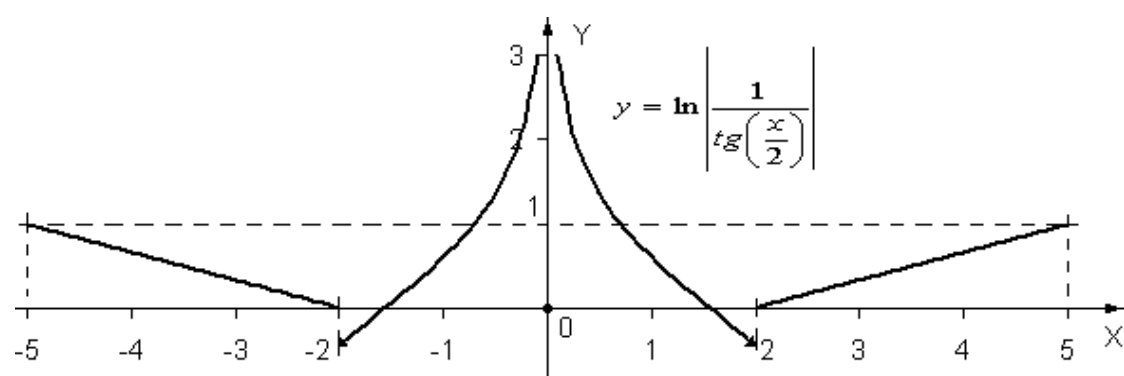
22)



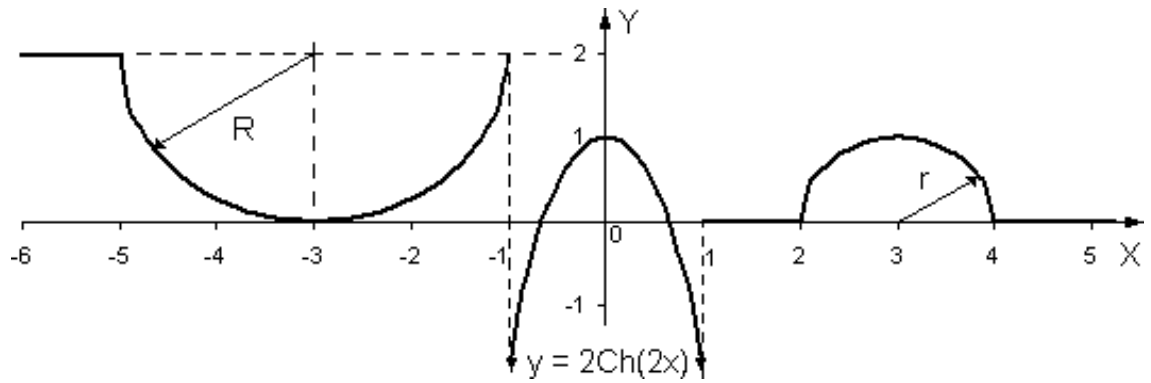
23)



24)

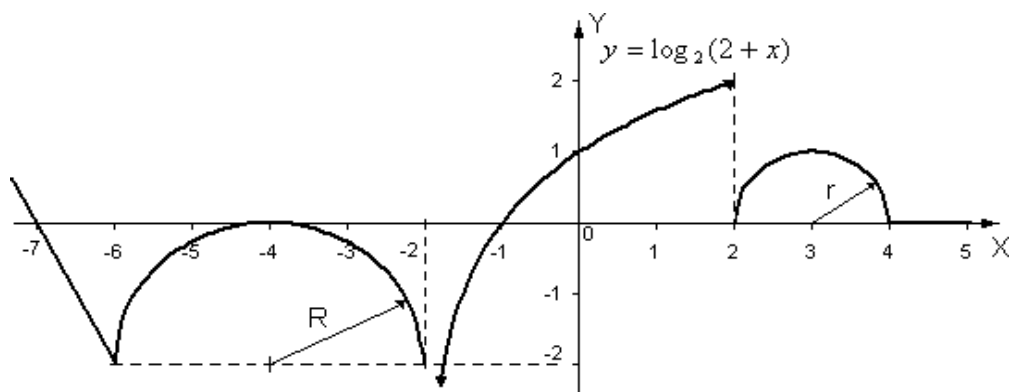


25)

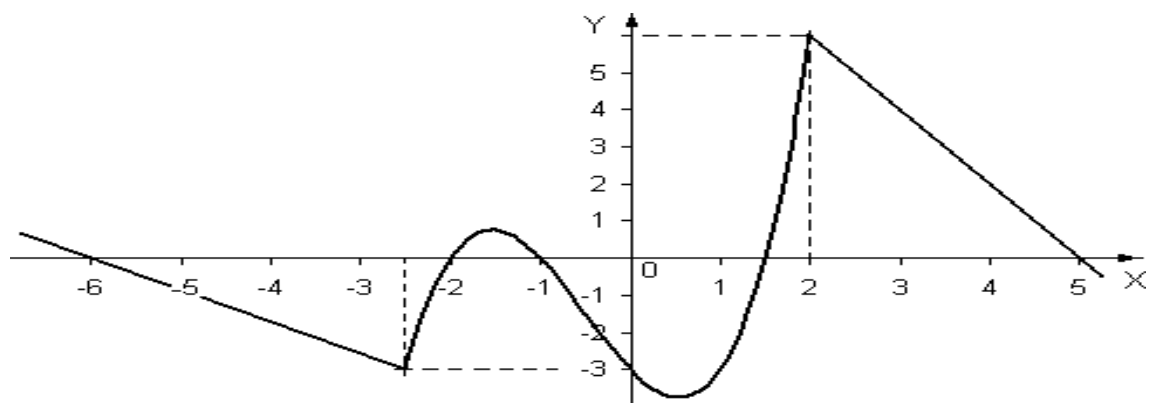


Hyperbolic cosine Maybe be described formula: $Ch(x) = \frac{1}{2} (e^x + e^{-x})$

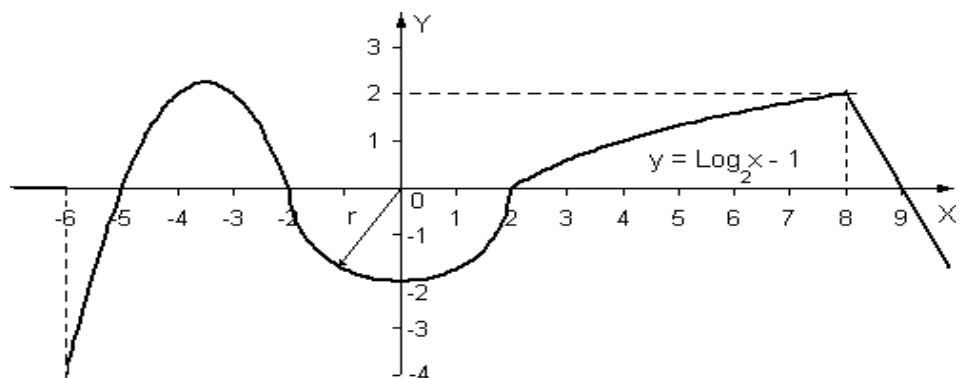
26)



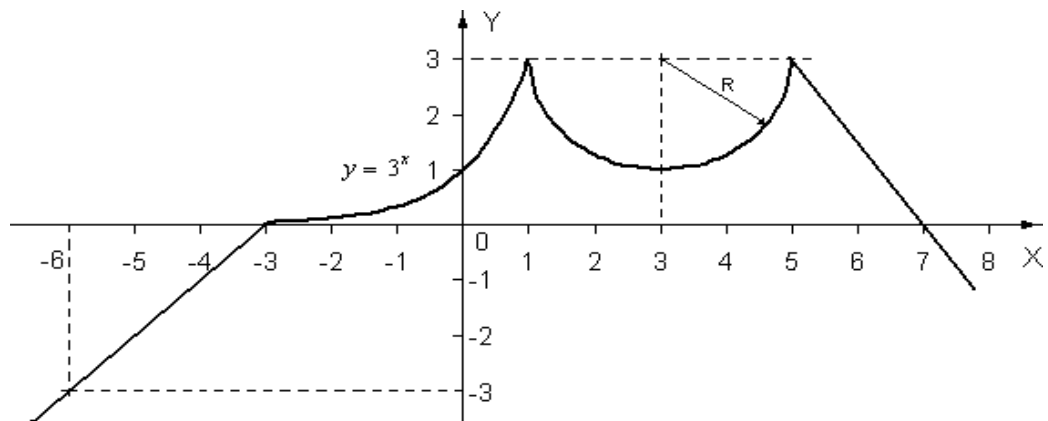
27)



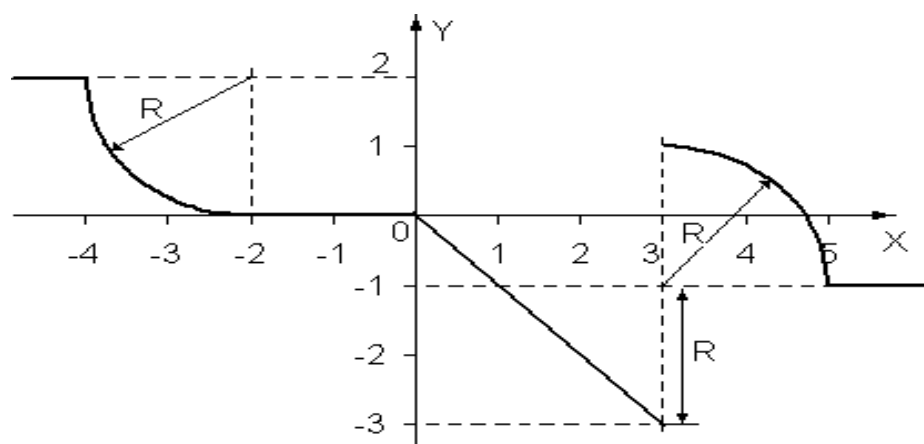
28)



29)



30)

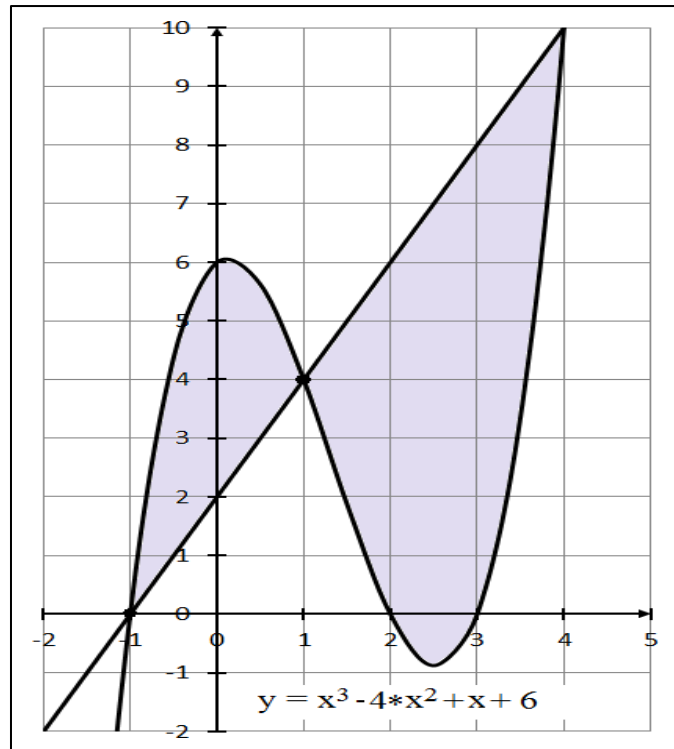


Additional requirement for task #30 : The program must be written so that solutions are obtained for different values of R entered from the keyboard. The center of the left quadrant of the circle changes in accordance with the entered radius, while the right quadrant remains constant. $X = 3$, $Y = -1$.

Practical Job No. 2: Branching Computing Processes. Task 2

Production tasks

Write a program that determines whether a point with given coordinates falls within a shaded area. Points on the boundary belong to the area. Obtain the necessary parameters from the figure. Display the program's output as a text message: Yes, No.



Theoretical introduction

For solutions tasks let's take advantage conditional operator:

```
if <Logical expression>:
    <Block - is being carried out, If
condition true> [ elif <Logical expression>:
    <Block - is being carried out, If condition true>
]
[ else:
    <Block - is being carried out, If All conditions
false>
]
```

A <block> is a set of instructions that are separated by an equal number of spaces (usually four).

It is necessary to correctly compose a logical expression, the parameters of which will be the value of the coordinates of the point (x , y) and the equations of the lines.

Exchange with the console is performed using standard

functions

input/output: `input()` And `print()` .

To solve the problem, we need to know the equation of the line (the equation of the curve is shown in the figure). From the figure, we obtain the coordinates of two points through which the straight line passes: (-1, 0) and (1, 4). Substituting the values of the selected points into the general equation $y = kx + b$ yields a system of equations. By solving this system, we find the values for the parameters k and b . As a result, the equation of the line becomes: $y = 2x + 2$.

A point with coordinates (x, y) entered by the user will fall within the shaded area on the interval $[-1, 1]$ if x is not less than -1 and not greater than 1. In addition, y must be no less than the value obtained for the straight line at point x and no greater than the value calculated for the cubic parabola at this point.

Description algorithm

1. Enter coordinates points (x, y) And bring values To type `float`.
2. Execute check on hit points V given region.
3. Withdraw result V view: "Dot x, y hits V region." And "Dot x, y Not hits V region."

Description input And weekend data

The input data are the coordinates of a point entered by the user. The data type and representation precision are not specified in the problem. We'll set the data type to `float`.

Weekend data - messages, V text view, O hit or missing points V given region.

Test examples

X	Y	Result
- 1	0	It hits
- 0.5	- 1	Doesn't hit
0	3	It hits
1	4	It hits
1	5	Doesn't hit
1.5	1	Doesn't hit
2	3	It hits
2.5	- 1	Doesn't hit
2.5	- 0.3	It hits
3.5	10	Doesn't hit

Listing programs

```
# -*- coding: cp1251 -*-
from math import *
flag = 0
print('Enter coordinates X And Y For
points:') x = float(input('X='))
y = float(input('Y='))
```

```

if (x < -1) or (x > 4):
    flag = 0          #False
if ((x>=-1) and (x<1) and (y>=2*x+2)
    and (y<=x**3-4*x**2+x+6) or
    (x>=1) and (x<=4) and (y>=x**3-4*x**2+x+6)
    and (y<=2*x+2)):
    flag = 1
else:
    flag = 0
print(" Point X={0: 6.2f} Y={1: 6.2f}"
      .format(x, y), end=" ")
if flag:
    print("hits", end=" ") else:
    print("not hits", end=" ")
print("to the region.")

```

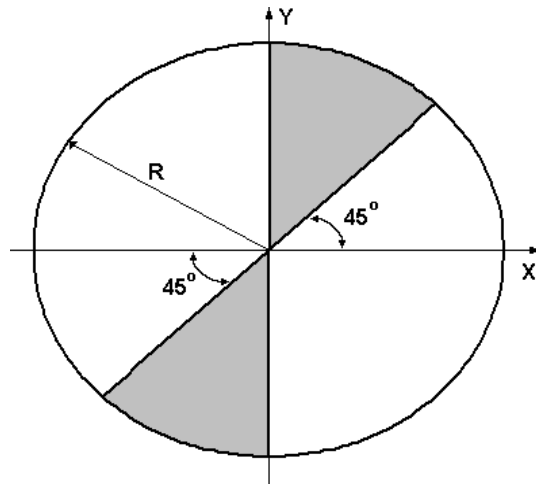
Comment To text programs: For transfer long lines V Python uses parentheses .

Exercise To practical work No. 2
"Branching computing processes."

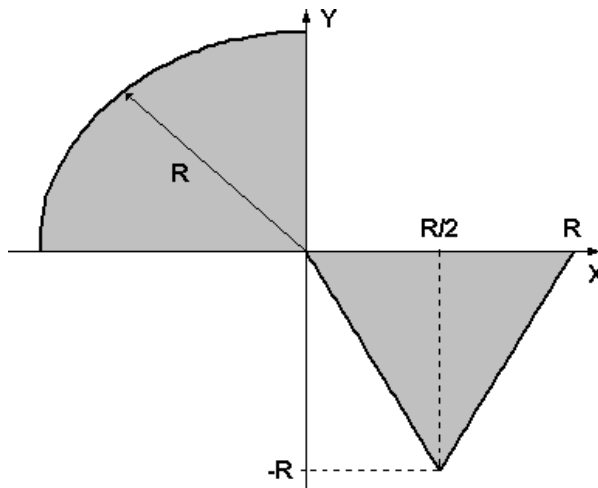
Exercise 2

Write a program that determines whether a point with given X, Y coordinates falls within the area shaded gray in the figure. Display the program's output as a text message. The R parameter is entered from the keyboard. Choose your solution.

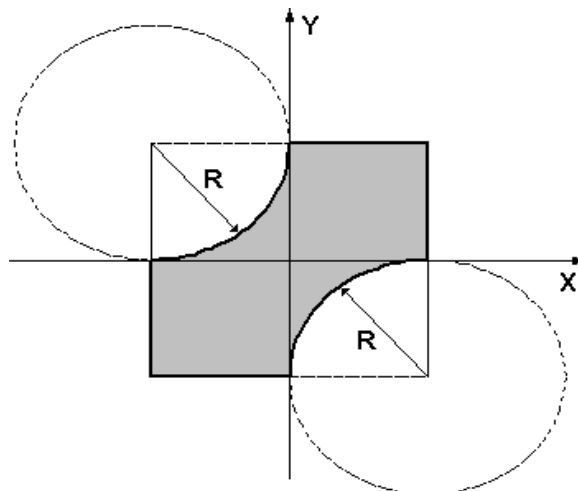
1)



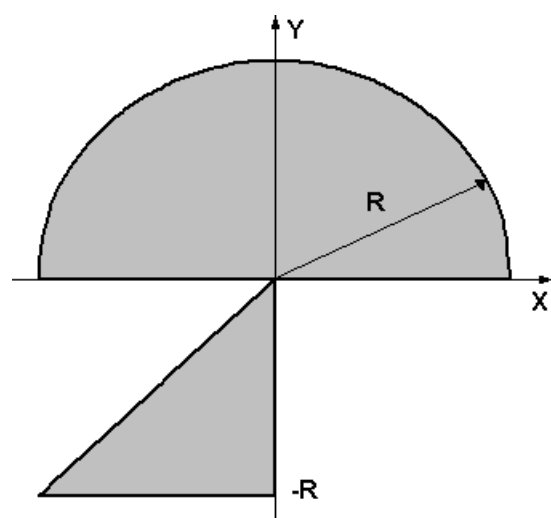
2)



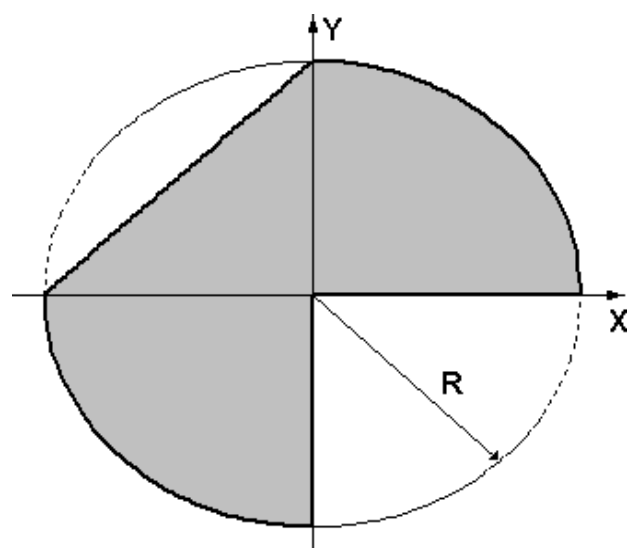
3)



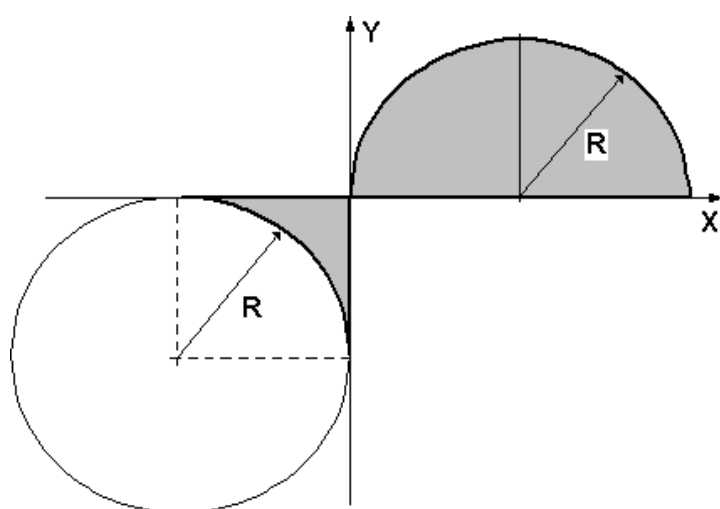
4)



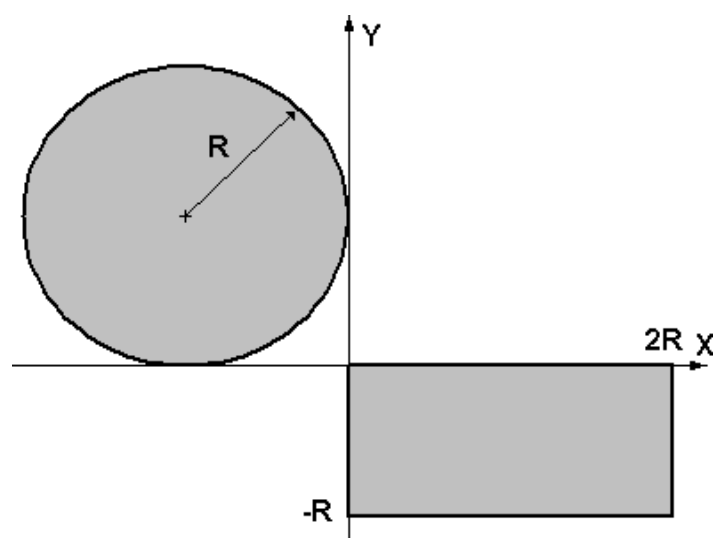
5)



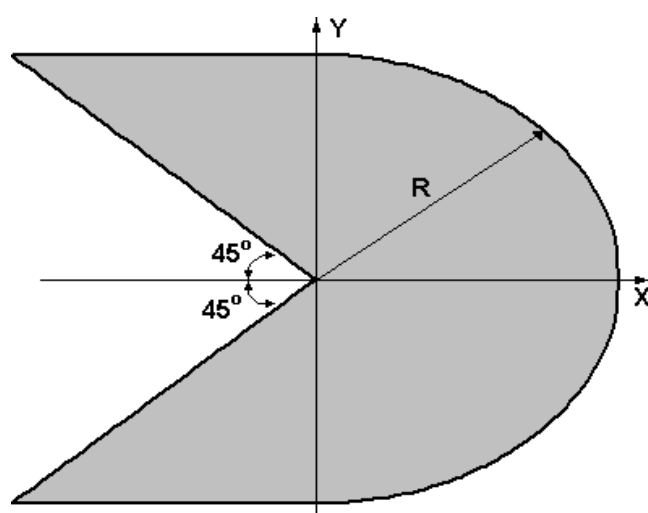
6)



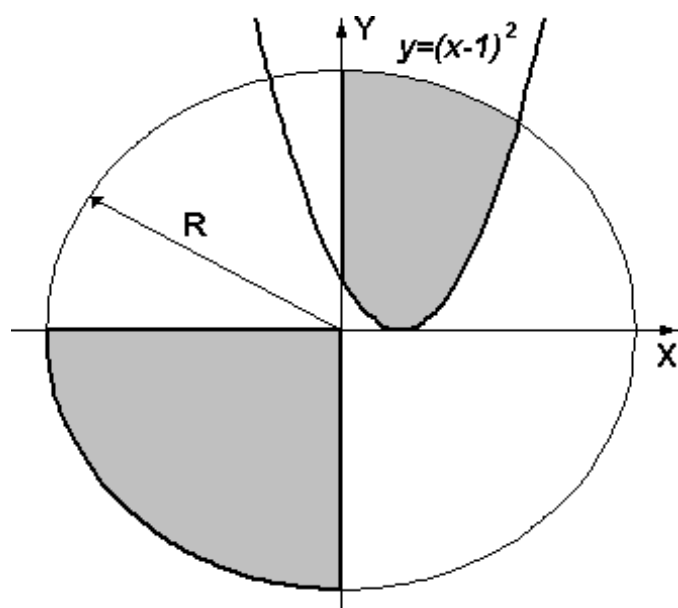
7)



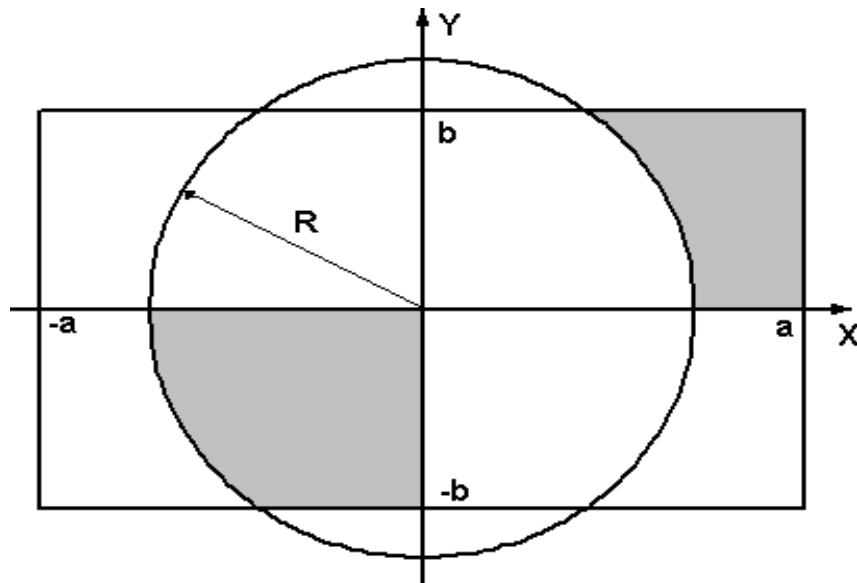
8)



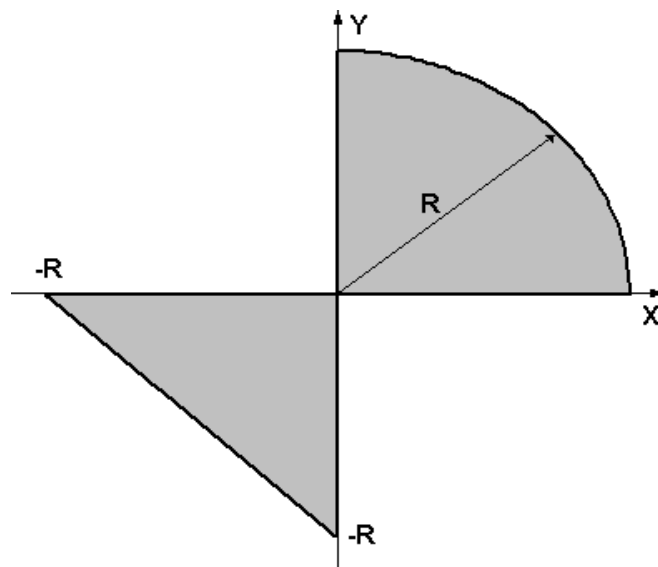
9)



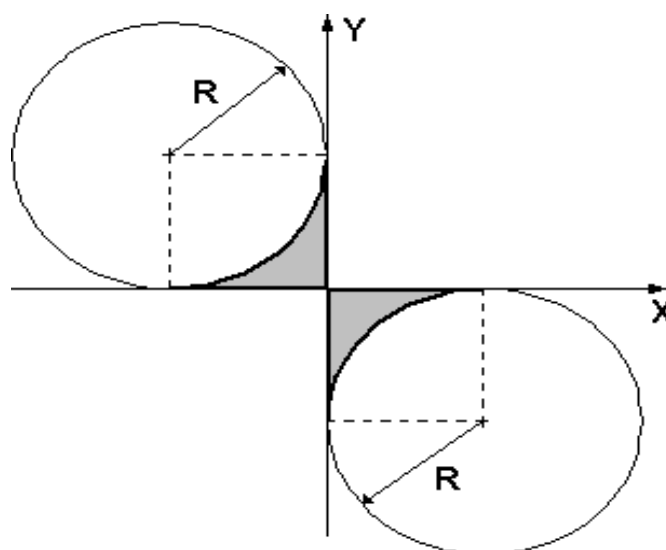
10)



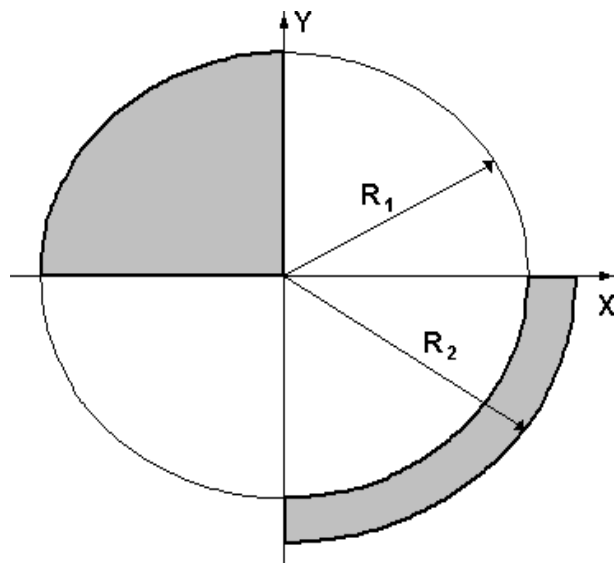
11)



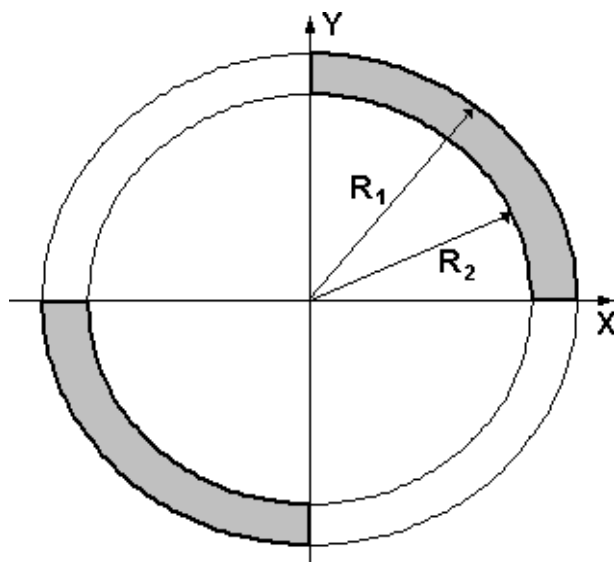
12)



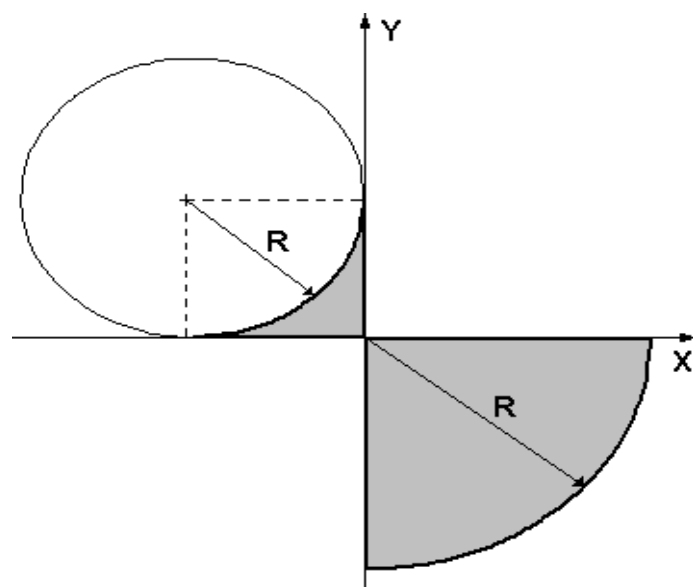
13)



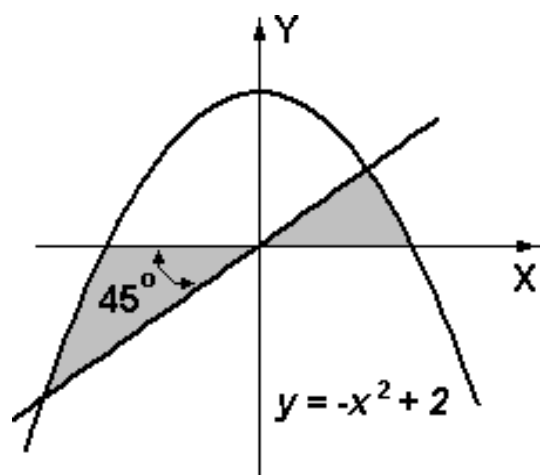
14)



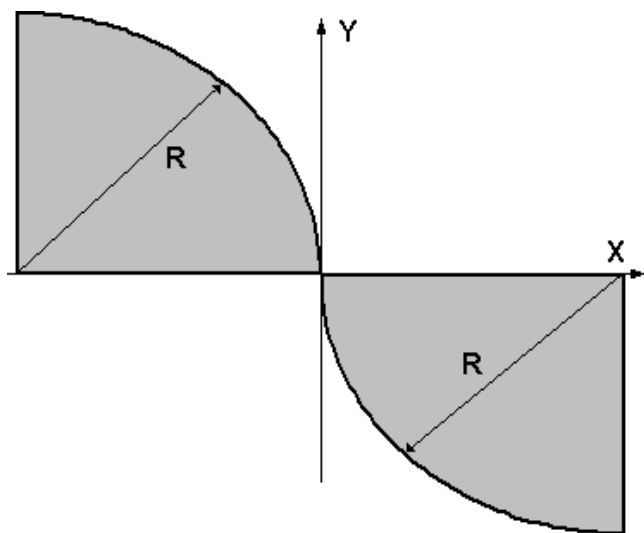
15)



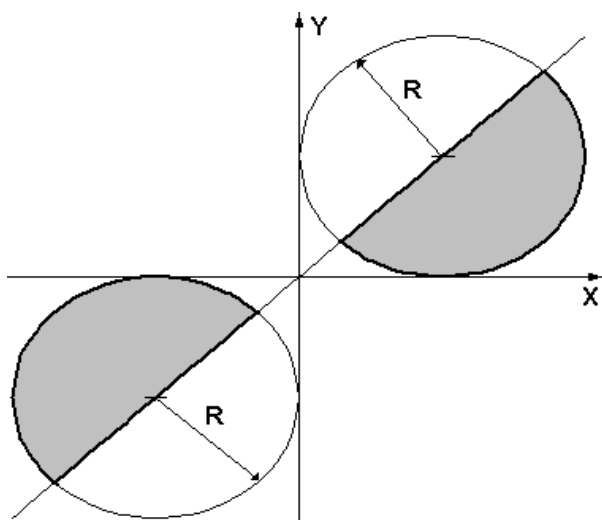
16)



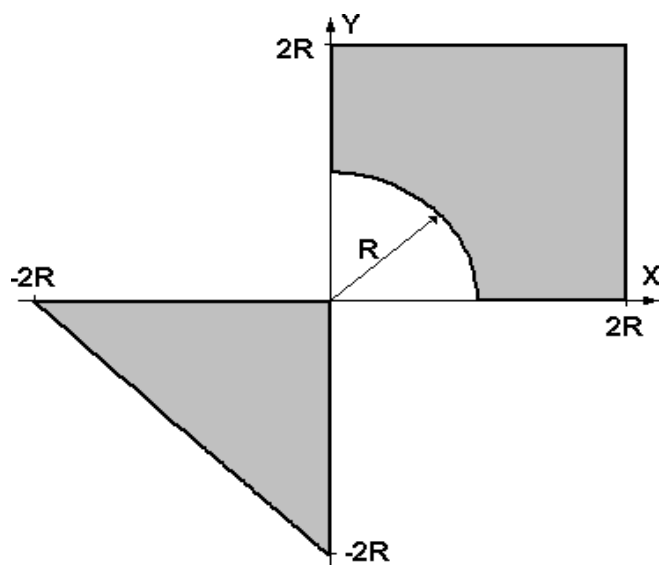
17)



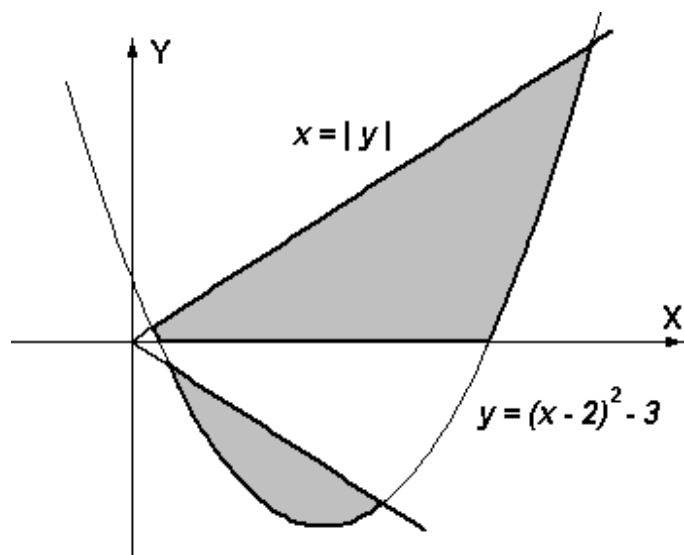
18)



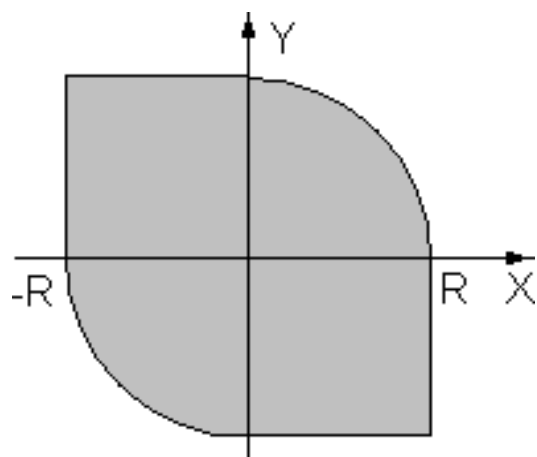
19)



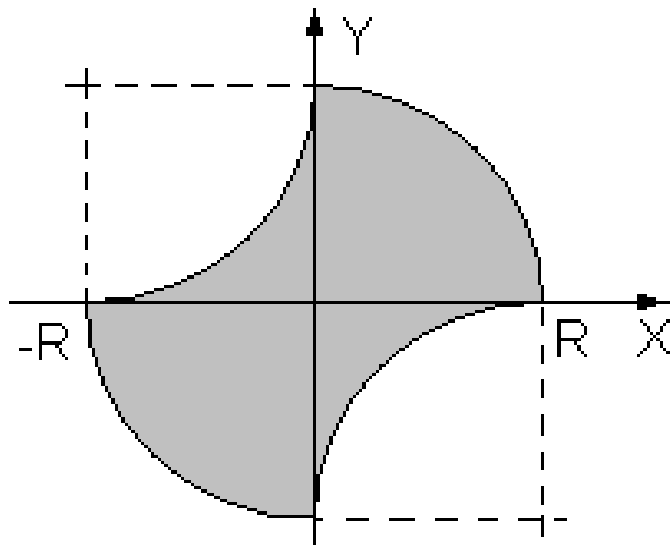
20)



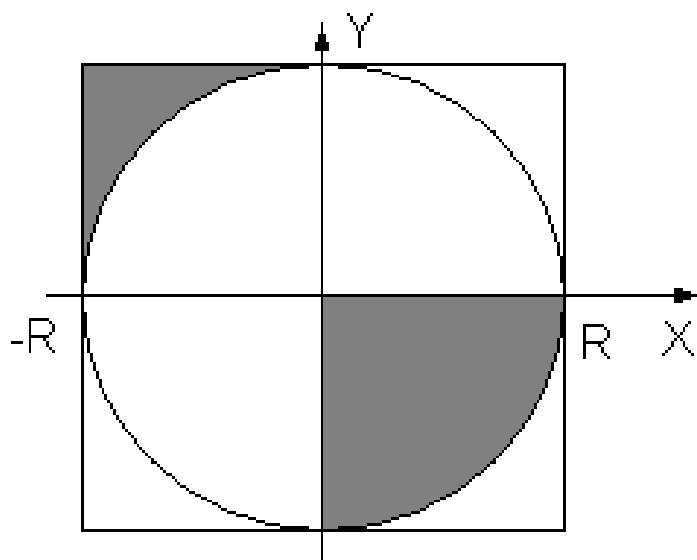
21)



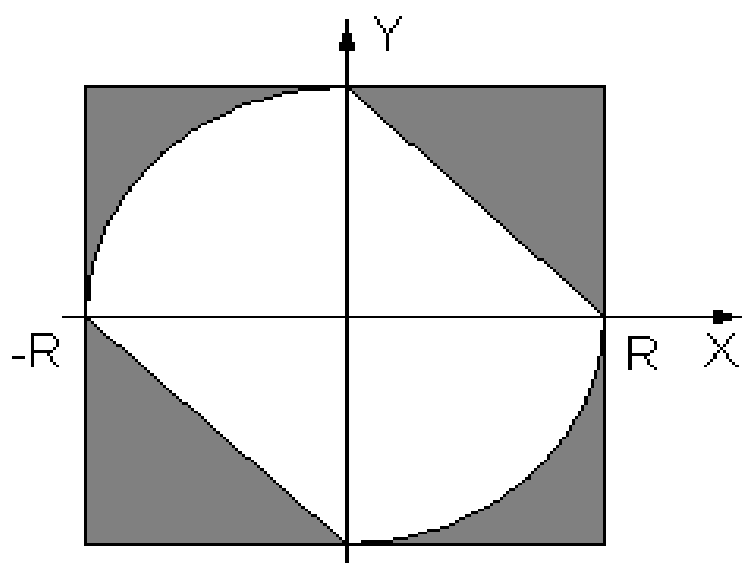
22)



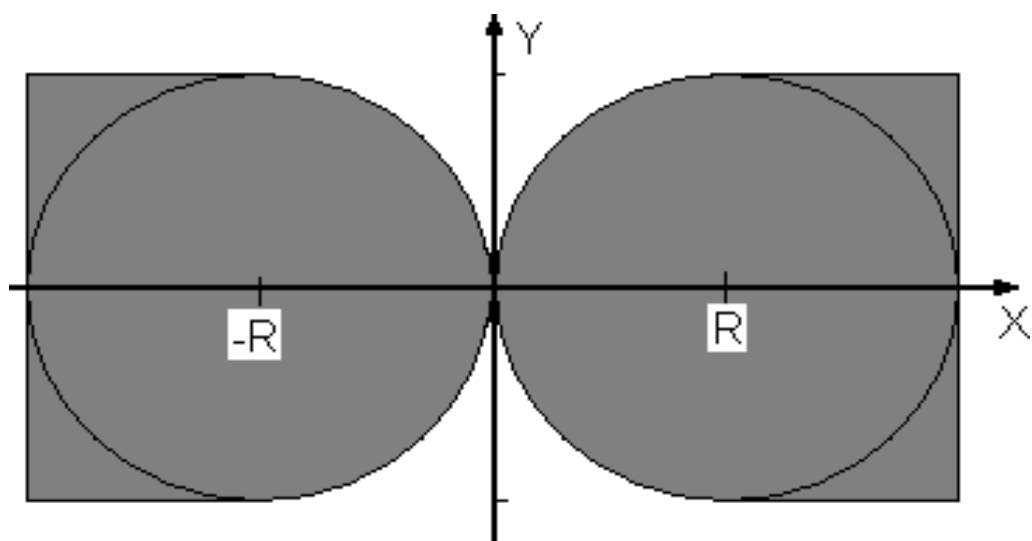
23)



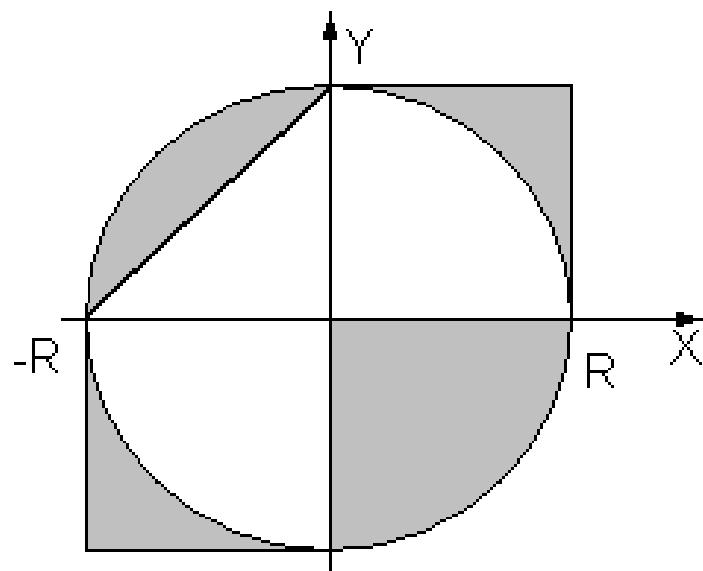
24)



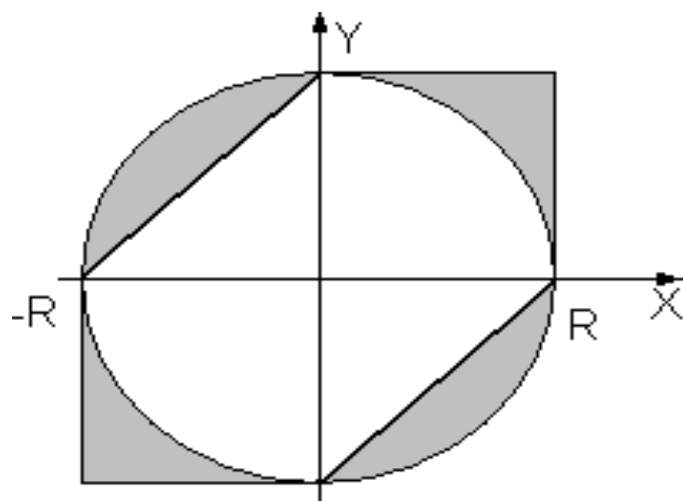
25)



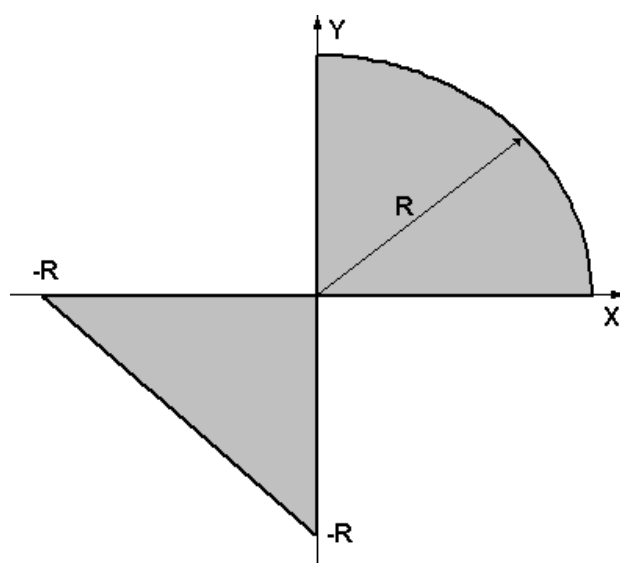
26)



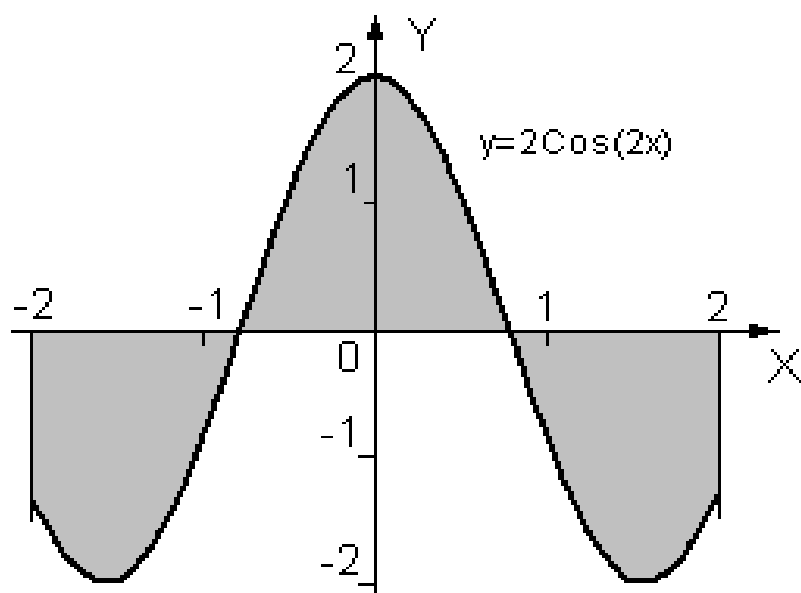
27)



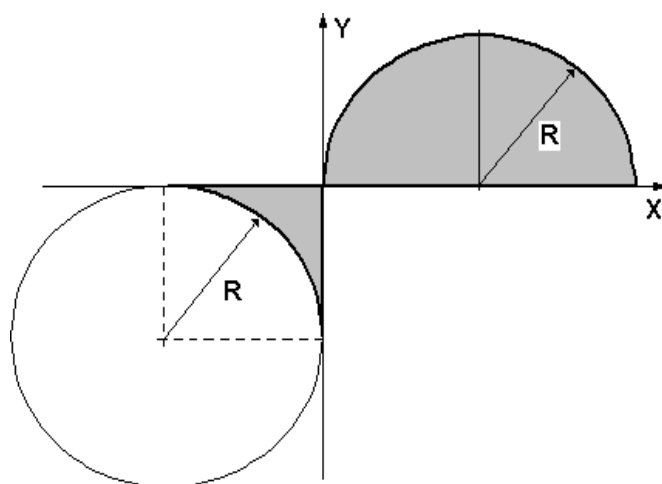
28)



29)



30)



Practical Job No. 3: "Organization cycles."

Exercise 1

Objective of the work:

To provide students with practical experience using basic structured programming constructs—loop operators. The assignment consists of three tasks.

Production tasks

Calculate and display as a table the values of a function defined graphically (see Practical Work #2, Task 1) on the interval from X_{start} to X_{end} with a step of dx . Specify the interval and step to check all branches of the program. Provide the table with a title and heading.

Theoretical introduction

To solve the problem, we used the program prepared in Practical Work No. 2, Task 1, and a loop operator with a precondition:

```
<Initial meaning>
while <Condition>:
    <Instructions>
    <Increment> [ else:
        <Block, performed, If Not was used break>
    ]
```

Standard `print()` and `input()` procedures are used for console communication (message output and initial data input). The program's output is written to a text file.

Description algorithm

1. Enter values variables X_{beg} , X_{end} , dx .
2. Assign current meaning X_t initial meaning: $X_t = X_{beginning}$.
3. Calculate meaning functions And withdraw V view lines tables.
4. Calculate new meaning argument $X_t = X_t + Dx$.
5. If meaning argument less X_{end} , That go to To point 3.
6. Finish drawing tables And work programs.

Description input And weekend data

In the previous work, a real data type was adopted. In this work, the data type is preserved. To simplify subsequent program control, the initial data is also written to the output text file.

Listing programs

```
# -*- coding: cp1251 -*-
from math import *
print('Введите Xbeg, Xend и Dx')
```

```

xb = float(input('Xbeg='))
xe = float(input('Xend='))
dx = float(input('Dx='))
print("Xbeg={0: 7.2f} Xend={1: 7.2f}"
      .format(xb, xe))
print("  Dx={0: 7.2f}".format(dx))
xt = xb
print("+-----+")
print("I    X    I    Y    I")
print("+-----+")
while xt <= xe:
    if xt < -5:
        y = 1
    elif xt >=-5 and xt<0:
        y = -(3/5)*xt-2
    elif xt >= 0 and xt<2:
        y = -sqrt(4-xt**2)
    elif xt >= 2 and xt<4:
        y = xt-2
    elif xt >= 4 and xt<8:
        y = 2+sqrt(4-(xt-6)**2)
    else: y = 2
    print("I{0: 7.2f} I{1: 7.2f} I"
          .format(xt, y))
    xt += dx
print("+-----+")

```

Result work programs

```

Xbeg= -10.00 Xend=  10:00
  Dx=    1.00
+-----+
I    X    I    Y    I
+-----+
I -10.00 And 1.00
AND
AND  -9.00 And    1.00 AND
AND  -8.00 And    1.00 AND
AND  -7.00 And    1.00 AND
AND  -6.00 And    1.00 AND
I  -5.00 To    1.00 I
I  -4.00 To    0.40 I
I  -3.00 I    -0.20 I
I  -2.00 I    -0.80 I
I  -1.00 I    -1.40 I
I   0.00 I    -2.00 I
I   1.00 I    -1.73 I
I   2.00 I     0.00 I
I   3.00 I     1.00 I

```

I 4.00 I 2.00 I

I	5.00	I	3.73	I
I	6.00	I	4.00	I
I	7.00	I	3.73	I
I	8.00	I	2.00	I
I	9.00	I	2.00	I
I	10.00	I	2.00	I
+-----+				

Tasks To practical work No. 3 "Organizing Cycles." Task 1

Calculate and display on the screen in the form of a table the values of a function defined graphically (see task 1 of practical work No. 2, p. 23), on the interval from X_{start} to X_{con} with a step of dx . Set the interval and step so that all program branches are checked. Provide the table with a title and header.

Practical Job No. 3: "Organization cycles."

Exercise 2

Production tasks

For ten shots, the coordinates of which are given by a random number generator, output text messages about hitting the target (see Practical Work No. 2, Task 2).

Theoretical introduction

To solve the problem, we used the program prepared in Practical Work No. 2, Task 2, and a loop operator with a parameter:

```
for < Current element > in < Sequence >:
    < Instructions inside cycle > [
else:
    < This code is being carried out, If V body cycle>
    <not was used break >]
```

Conclusion messages is being carried out standard function print().

For formations coordinates points is used module random number generator, which is connected by the instruction:

```
import random

or

from random import *
```

For formations random material numbers let's take advantage function

```
uniform(<Start>, <End>)
```

This function generates pseudo-random number V range from <Beginning> until <End>.

The right boundary is not included in the range of generated values (the range is open on the right). In our problem, the X values are generated in the range (-1, 4), and the Y values are generated in the range (-1, 10).

Description algorithm

1. Display the "header".
2. IN cycle from 1 to 10.
3. Form coordinates points X , Y .
4. Define hit points V given region. If if there is a hit, then the flag variable gets the value 1, otherwise – 0.
5. Display the coordinates of the point and leave the marker on the line the messages.
6. Withdraw result Yes or No V in accordance with the value of the flag variable .
7. Change parameter cycle And check condition completion.
8. If condition False , That go to To p. 3.
9. Finish work programs.

Description input And weekend data

The variable types used in the previous work remained unchanged. A new integer variable (`int`) was introduced to organize the loop.

Listing programs

```
# -*- coding: cp1251 -*-
from math import *
from random import *
flag = 0
print("      X      Y      Res")
print(" ----- ")
for n in range(10):
    x = uniform(-1, 4)
    y = uniform(-1, 10)
    if (x < -1) or (x > 4):
        flag = 0          #False
    if ((x>=-1) and (x< 1) and (y>=2*x+2)
        and (y<= x**3-4*x**2+x+6))
        or
        ((x>=1) and (x<=4) and (y>=x**3-4*x**2+x+6)
        and (y<= 2*x+2))):
        flag = 1
    else:
        flag = 0
    print("{0: 7.2f} {1: 7.2f}"
        .format(x, y), end=" ")
    if flag:
        print("Yes")
    else:
        print("No")
```

Result testing programs

	X	Y	Res
-	0.68	7.15	No
	3.53	3.44	No
-	0.05	4.02	Yes
-	0.24	0.01	No
	2.48	5.58	Yes
	0.41	4.77	Yes
	0.09	0.89	No
	0.68	0.98	No
-	0.03	5.46	Yes
-	0.73	0.73	Yes

Tasks To practical work No. 3 "Organizing Cycles." Task 2

For ten shots, the coordinates of which are given by a random number generator, output text messages about hitting the target (see Practical Work No. 2, Task 2).

Practical Job No. 3: "Organization cycles."

Exercise 3

Production tasks

Calculate and display on the screen in the form of a table the values of the integral sine function, given by power series, on the interval from X_{init} to X_{con} in increments of dx with an accuracy of ε - epsilon.

Table supply title And with a hat. Each one line tables must contain the value of the argument, the value of the function and the number of summed terms of the series.

$$Si(x) = \int_0^x \sin(x) dx = \sum_{n=0}^{\infty} (-1)^n \cdot \frac{x^{2 \cdot n + 1}}{(2 \cdot n + 1)! \cdot (2 \cdot n + 1)} = x - \frac{x^3}{3! \cdot 3} + \frac{x^5}{5! \cdot 5} - \dots + \left| x \right| < \infty$$

Theoretical introduction

When solving similar problems that require a general formula for calculating a series element, it's very useful to derive a recurrence formula. This formula simplifies the programming process and speeds up program execution, as the next result is based on the previous one. Particular attention should be paid to the convergence of the series. If the series diverges or converges weakly, the program may generate variable overflow messages or produce an incorrect result. Pay attention to the limits of acceptable argument values.

1. We will search recurrent ratio V view expressions:

$$a_{n+1} = k \cdot a_n. \text{ We will receive expression for } k:$$

$$k = \frac{(-1)^{n+1} \cdot x^{2 \cdot (n+1)+1} \cdot (2 \cdot n + 1)! \cdot (2 \cdot n + 1)}{(2 \cdot (n+1)+1)! \cdot (2 \cdot (n+1)+1) \cdot (-1)^n \cdot x^{2 \cdot n+1}} = - \frac{x^2 \cdot (2 \cdot n + 1)}{(2 \cdot n + 2) \cdot (2 \cdot n + 3)^2}$$

When performing the transformation, the factorial expression in the denominator was transformed to the form:

$$(2 \cdot (n+1)+1)! = (2 \cdot n + 3)! = (2 \cdot n + 1)! \cdot (2 \cdot n + 2) \cdot (2 \cdot n + 3)$$

2. For solutions tasks us will be required two cycle. First `While` loop (with precondition), will ensure a change in the value of variable X from X_{nach} to X_{kon} With step by step dx . Second cycle - This cycle With postcondition. He

needed For iterative calculations elements rows With $a_n < |\varepsilon|$.

IN language Python cycle With postcondition absent. For organizations For such a cycle, we use a cycle with a precondition in the following form:

```
while True:
    < body cycle >
    if not < condition > :
        break ;
```

Standard `input()` functions are used for exchange with the console (input/output) and `print()`.

Description algorithm

1. Enter values variables `Xnach` , `Xkon` , `dx` And accuracy parameter ε .
2. Withdraw hat tables.
3. Initiate `Xt` initial meaning (`Xbeginning`).
4. IN cycle By `Xt` .
5. Initialize a variable for calculating the sum of the series members and a variable that is responsible for the number of the series member (`n`).
6. IN cycle By `an` .
7. Calculate `k` , the element of the series `an` , the sum of the elements of the series and the number of the element.
8. If module element rows less ε , That interrupt cycle (`break`) by `an` , otherwise go to point 6.
9. Display the table rows: the value of `Xt` , the calculated value of the function, and the number of summed terms of the series
10. Calculate new meaning variable `Xt = Xt + dx` .
11. If meaning argument less `Xkon` , That go to To point 4.
12. Finish drawing tables, And work programs.

Description input And weekend data

Since the type of variables and the accuracy of representation are not limited by the condition of the problem, the input variables (`Xstart` , `Xend` , `dx` and the precision parameter ε) and the calculated values of the argument and function are represented by variables of real type (`float`). The number of terms of the series is counted by a variable of integer type (`int`).

Listing programs

```
# -*- coding: cp1251 -*-
from math import *
print('Введите Xbeg, Xend, Dx и Eps')
xb = float(input('Xbeg='))
xe = float(input('Xend='))
dx = float(input('Dx='))
eps = float(input('Eps='))
print("+-----+-----+-----+")
print("I      X      I      Y      I      N I")
print("+-----+-----+-----+")
xt = xb
while xt <= xe:
    an = xt
```

$$n = 0$$

```

y = an
while True:
    k=-(xt**2)*(2*n+1)/((2*n+2)*(2*n+3)**2)
    an = an*k
    y = y + an
    n = n + 1
    if abs(an) < eps:
        break
print("I{0: 7.2f} I{1: 7.3f} I{2: 4} I"
      .format(xt,y,n))
xt = xt + dx
print("+-----+-----+-----+")

```

The result works programs

```

Next=  -4.00  Xcon=    6.00
Dx=    2.00   Eps=    0.00003
+-----+-----+-----+
I   X       I   Y       I   N I
+-----+-----+-----+
I  -4.00 I -1,758 The  8 I
I  -2.00 I -1.605 I    5 I
I   0.00 I  0.000 I    1 I
I   2.00 I  1.605 I    5 I
I   4.00 I  1.758 I    8 I
I   6.00 I  1.425 I   10 I
+-----+-----+-----+

```

Exercise To practical work No. 3 "Organizing Cycles." Task 3

Calculate and display the function values on the monitor screen in the form of a table, given using the Taylor series, on the interval from X_{start} to X_{kon} with step dx and accuracy ε . Provide the table with a title and heading. Each row of the table should contain the value of the argument, the value of the function, and the number of summed terms of the series.

$$1. \ln \frac{x+1}{x-1} = 2 \cdot \sum_{n=0}^{\infty} \frac{1}{(2 \cdot n + 1) \cdot x^{2 \cdot n + 1}} = 2 \cdot \left(\frac{1}{x} - \frac{1}{3 \cdot x^3} + \frac{1}{5 \cdot x^5} - \dots \right), \quad |x| > 1$$

$$2. e^{-x} = \sum_{n=0}^{\infty} \frac{(-1)^n \cdot x^n}{n!} = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \frac{x^4}{4!} - \dots, \quad |x| < \infty$$

$$3. e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \dots, \quad |x| < \infty$$

$$4. \ln(1+x) = \sum_{n=0}^{\infty} \frac{(-1)^n \cdot x^{n+1}}{n+1} = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \dots, \quad -1 < x \leq 1$$

$$5. \ln \frac{1+x}{1-x} = 2 \cdot \sum_{n=0}^{\infty} \frac{x^{2 \cdot n + 1}}{2 \cdot n + 1} = 2 \cdot \left(x + \frac{x^3}{3} + \frac{x^5}{5} + \dots \right), \quad |x| < 1$$

$$6. \ln(1-x) = - \sum_{n=1}^{\infty} \frac{x^n}{n} = - \left(x + \frac{x^2}{2} + \frac{x^3}{3} + \dots \right), \quad -1 \leq x < 1$$

$$7. \operatorname{arctg}(x) = \frac{\pi}{2} + \sum_{n=0}^{\infty} \frac{(-1)^{n+1} \cdot x^{2 \cdot n + 1}}{2 \cdot n + 1} = \frac{\pi}{2} - x + \frac{x^3}{3} - \frac{x^5}{5} + \dots, \quad x \leq 1$$

$$8. \operatorname{arctg}(x) = \frac{\pi}{2} + \sum_{n=0}^{\infty} \frac{(-1)^{n+1}}{(2 \cdot n + 1) \cdot x^{2 \cdot n + 1}} = \frac{\pi}{2} - \frac{1}{x} + \frac{1}{3 \cdot x^3} - \frac{1}{5 \cdot x^5} + \dots, \quad x > 1$$

$$9. \operatorname{arctg}(x) = \sum_{n=0}^{\infty} \frac{(-1)^{n+1} \cdot x^{2 \cdot n + 1}}{(2 \cdot n + 1)} = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \dots, \quad |x| \leq 1$$

$$10. \operatorname{Arth}(x) = \sum_{n=0}^{\infty} \frac{x^{2 \cdot n + 1}}{2 \cdot n + 1} = x + \frac{x^3}{3} + \frac{x^5}{5} + \frac{x^7}{7} + \dots, \quad |x| < 1$$

$$11. \operatorname{Arcth}(x) = \sum_{n=0}^{\infty} \frac{1}{(2 \cdot n + 1) \cdot x^{2 \cdot n + 1}} = \frac{1}{x} + \frac{1}{3 \cdot x^3} + \frac{1}{5 \cdot x^5} + \dots, \quad |x| > 1$$

12.

$$\frac{1}{x+3} = \frac{1}{2} - \frac{(x+1)}{2^2} + \frac{(x+1)^2}{3} - \dots = \sum_{n=0}^{\infty} \frac{(-1)^n \cdot (x+1)^n}{2^{n+1}}$$

, $-3 < x < 1$

$$13. \text{ it } \sum_{n=0}^{\infty} \frac{(-1)^n \cdot x^{2n}}{n!} = 1 - x^2 + \frac{x^4}{2!} - \frac{x^6}{3!} + \frac{x^8}{4!} - \dots, \quad |x| < \infty$$

$$14. \cos(x) = \sum_{n=0}^{\infty} \frac{(-1)^n \cdot x^{2n}}{(2n)!} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \dots, \quad |x| < \infty$$

$$15. \frac{\sin(x)}{x} = \sum_{n=0}^{\infty} \frac{(-1)^n \cdot x^{2n}}{(2n+1)!} = 1 - \frac{x^2}{3!} + \frac{x^4}{5!} - \frac{x^6}{7!} + \dots, \quad |x| < \infty$$

$$16. \ln(x) = 2 \cdot \sum_{n=0}^{\infty} \frac{(x-1)^{2n+1}}{(2n+1) \cdot (x+1)^{2n+1}} = 2 \cdot \left(\frac{x-1}{x+1} + \frac{(x-1)^3}{3 \cdot (x+1)^3} + \frac{(x-1)^5}{5 \cdot (x+1)^5} + \dots \right), \quad x > 0$$

$$17. \ln(x) = \sum_{n=0}^{\infty} \frac{(x-1)^{n+1}}{(n+1) \cdot (x+1)^{n+1}} = \frac{x-1}{x} + \frac{(x-1)^2}{2 \cdot x^2} + \frac{(x-1)^3}{3 \cdot x^3} + \dots, \quad x > \frac{1}{2}$$

$$18. \ln(x) = \sum_{n=0}^{\infty} \frac{(-1)^n \cdot (x-1)^{n+1}}{(n+1)} = (x-1) - \frac{(x-1)^2}{2} + \frac{(x-1)^3}{3} - \dots, \quad 0 < x \leq 2$$

$$19. \arcsin(x) = x + \sum_{n=1}^{\infty} \frac{(-1)^{n-1} \cdot x^{2n+1}}{2 \cdot 4 \cdot \dots \cdot 2n \cdot (2n+1)} = x + \frac{x^3}{2 \cdot 3} + \frac{1 \cdot 3 \cdot x^5}{2 \cdot 4 \cdot 5} + \frac{1 \cdot 3 \cdot 5 \cdot x^7}{2 \cdot 4 \cdot 6 \cdot 7} + \dots, \quad |x| < 1$$

$$20. \arccos(x) = \frac{\pi}{2} - \left(x + \sum_{n=1}^{\infty} \frac{1 \cdot 3 \cdot \dots \cdot (2n-1) \cdot x^{2n+1}}{2 \cdot 4 \cdot \dots \cdot (2n) \cdot (2n+1)} \right) = \frac{\pi}{2} - \left(x + \frac{x^3}{2 \cdot 3} + \frac{1 \cdot 3 \cdot x^5}{2 \cdot 4 \cdot 5} + \frac{1 \cdot 3 \cdot 5 \cdot x^7}{2 \cdot 4 \cdot 6 \cdot 7} + \frac{1 \cdot 3 \cdot 5 \cdot 7 \cdot x^9}{2 \cdot 4 \cdot 6 \cdot 8 \cdot 9} + \dots \right), \quad |x| < 1$$

$$\mathbf{21.} \quad sh(x) = \frac{-}{2} \cdot (e^{-} - e^{-}) = \sum \frac{}{(2 \cdot n + 1)!} = x + \frac{}{3!} + \frac{}{5!} + \frac{}{7!} + \quad \left[x^2 < \infty \right]$$

$$22 \quad ch(x) = \frac{1}{2} \cdot (e^x + e^{-x}) = \sum_{n=0}^{\infty} \frac{x^{2 \cdot n}}{(2 \cdot n)!} = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \frac{x^6}{6!} + \dots \quad \left[\frac{x^2}{2} < \infty \right]$$

$$23 \quad \ln\left(\frac{x+1}{x}\right) = 2 \cdot \left[\frac{1}{2x+1} + \frac{1}{3 \cdot (2x+1)^3} + \frac{1}{5 \cdot (2x+1)^5} + \dots \right] \quad (2x+1)^2 > 1$$

$$24 \quad (1+x)^{\frac{1}{4}} = 1 + \frac{1}{4} \cdot x - \frac{1}{4} \cdot \frac{3}{8} \cdot x^2 + \frac{1}{4} \cdot \frac{3}{8} \cdot \frac{7}{12} \cdot x^3 - \frac{1}{4} \cdot \frac{3}{8} \cdot \frac{7}{12} \cdot \frac{11}{16} \cdot x^4 + \dots \quad |x| \leq 1$$

$$25 \quad (1-x)^{\frac{1}{2}} = 1 - \frac{1}{2} \cdot x + \frac{1}{2} \cdot \frac{1}{4} \cdot x^2 - \frac{1}{2} \cdot \frac{1}{4} \cdot \frac{3}{6} \cdot x^3 + \frac{1}{2} \cdot \frac{1}{4} \cdot \frac{3}{6} \cdot \frac{5}{8} \cdot x^4 - \dots \quad |x| \leq 1$$

$$26 \quad (1+x)^{-\frac{1}{3}} = 1 - \frac{1}{3} \cdot x + \frac{1}{3} \cdot \frac{1}{6} \cdot x^2 - \frac{1}{3} \cdot \frac{1}{6} \cdot \frac{7}{9} \cdot x^3 + \frac{1}{3} \cdot \frac{1}{6} \cdot \frac{7}{9} \cdot \frac{10}{12} \cdot x^4 - \dots \quad |x| \leq 1$$

$$27 \quad (1+x)^{-3} = \frac{1}{1 - \frac{1}{2}} (2 \cdot 3 \cdot x - 3 \cdot 4 \cdot x^2 + 4 \cdot 5 \cdot x^3 - 5 \cdot 6 \cdot x^4 + \dots) \quad |x| \leq 1$$

$$28 \quad (1-x)^{-4} = \frac{1}{1 \cdot 2 \cdot 3} (2 \cdot 3 \cdot 4 \cdot x + 3 \cdot 4 \cdot 5 \cdot x^2 + 4 \cdot 5 \cdot 6 \cdot x^3 + 5 \cdot 6 \cdot 7 \cdot x^4 + \dots) \quad |x| \leq 1$$

$$29 \quad (1+x)^{-5} = \frac{1}{1 \cdot 2 \cdot 3 \cdot 4} (2 \cdot 3 \cdot 4 \cdot 5 \cdot x - 3 \cdot 4 \cdot 5 \cdot 6 \cdot x^2 + 4 \cdot 5 \cdot 6 \cdot 7 \cdot x^3 - 5 \cdot 6 \cdot 7 \cdot 8 \cdot x^4 + \dots) \quad |x| \leq 1$$

$$30 \quad \ln(1+2x) = \ln 7 + \frac{2(x-3)}{7} - \frac{2^2(x-3)^2}{2 \cdot 7^2} + \dots (-1)^{n+1} \cdot \frac{2^n(x-3)^n}{n \cdot 7^n}, \quad -\frac{1}{2} < x \leq \frac{13}{2}$$

Practical Job No. 4: "One-dimensional arrays"

Objective of the work:

To give students practical skills in writing programs for processing one-dimensional arrays: finding maximums and minimums, sorting.

Note : There are different ways to organize arrays in Python, such as lists or solutions offered in the `array` , `numpy` , modules. These solutions may contain ready-made methods and functions that simplify programming, such as finding the maximum, minimum, or sorting. This guide suggests using algorithms search for solutions without using ready-made methods and functions.

Production tasks

Generate a one-dimensional list consisting of N real numbers generated by a random number generator. The number of list elements (N) is requested from the user, but does not exceed 30. The range of element values is from -5.0 to 5.0. Calculate:

1. First And second maximum By module elements list.
2. Amount elements, module which less units.
3. All elements, module which exceeds `Amax` reset.
3. Sort list, preserving order non-zero elements. Place zero-valued elements at the end of the list.

Theoretical introduction

For working with one-dimensional arrays and matrices (multidimensional arrays), Python has specialized modules and libraries (`array` , `numpy` , ...).

Array - This final named subsequence of the same type quantities.

To organize an array in Python, you can use data structures such as lists, tuples, sets, and ranges, which represent numbered collections of objects. Each element of the collection contains a reference to an object, which in turn can represent an object of any data type and have unlimited nesting .

In solving this problem, to store similar data (data arrays), it is proposed to use a data structure called **a list** .

A list is a sequence of elements numbered starting from **0 (zero)** . List elements can have different types. A list can be defined by listing the list elements in square brackets, for example, a list can be defined like this:

```
Preshe = [1, 3, 5, 7, 11, 13]
PColor = ['Red', 'Orange', 'Yellow', 'Green']
```

или так:

```
Trest = [1, 3, 'Old', 7, 'Or', 13]
```

В списке `Preshe` — 6 элементов, а именно:

```
Preshe[0] == 1 , Preshe[1] == 3 , Preshe[2] == 5 ,  
Leek[3] == 7 , Leek[4] == 11 , Leek[5] == 13 .
```

List `PColor` consists of from 4-x elements, each from which is a string, but here is a list of `Trest` consists of six elements, some of which are numbers and some of which are strings.

When accessing list elements, you can use either positive or negative indices. If the index is positive, the counting proceeds from zero to the maximum element, left to right. If the index is negative, the counting proceeds from right to left:

```
Preshe[-1] == 13, Preshe[-6] == 1 .
```

Quantity elements `V` list (length list), Can get using the `len` function , for example, `len(Preshe) == 6` .

Exists some ways work with list. Developers offer various options from special modules to libraries. The standard language solutions are sufficient for our example.

Let's consider one from ways creation list And his filling:

- Request the list size from the user. This size should not be less than 5 elements and not exceed 30;

- Create empty list (Not containing elements, lengths 0);

- Let's fill it up list random numbers, applying method `append` .

```
n = int(input("Elements V list (N<=30) N: "))  
if n > 30: n = 30
```

```
elif n < 5: n = 5
```

```
mas = [] # We create empty list
```

```
for i in range(n): # Initialization
```

```
    mas.append(uniform(-5, 5))
```

If you use, for example, the `array` module , the changes to the program will be minimal:

```
# creating a zero-length array #
```

```
With elements material type mas =
```

```
array('f')
```

```
for i in range(n): #
```

```
    Initialization
```

```
    mas.append(uniform(-5, 5))
```

The solution to the problem uses a `for` loop to generate a sequence of integers used to access array elements:

```
for <Current element> in <Sequence>:
```

```
    <Instructions inside cycle>
```

Standard input() functions are used for exchange

with the console (input/output) and print() .

After the list is formed, the next loop calculates the sum of elements whose absolute value does not exceed one. In this same loop, if an element in the array is found whose value exceeds the threshold, the element is reset to zero.

The search for the first and second maximal elements is based on the principle of a single-pass algorithm:

- Module element $A[i]$ compare with meaning V_{Max1} ;
- If $abs(A[i]) > Max1$, That $Max2 = Max1$
And
 $Max1 = abs(A[i])$;
- Otherwise module element $A[i]$ compare With $Max2$. If
 $abs(A[i]) > Max2$, That $Max2 = abs(A[i])$;

To compress an array according to a given principle (zero elements are placed at the end of the array while maintaining the order of non-zero elements), the following algorithm is used:

- Array is visible With using indexed variable i , the additional index j points to the first (zero) element of the array;
- When a non-null element is found, that element is copied into the element With index j And index j increments, pointing to the next vacant position;
- After array view all elements, starting from the element With index j are reset.

Description algorithm

1. Request quantity elements N And threshold meaning A_{max} .
2. Initialize the array with random data and print the initial state.
3. IN cycle from 0 up to $N-1$. Find the sum of elements whose modulus is less than 1 , and reset the elements whose value exceeds the set threshold A_{max} .
4. Initiate $Max1$ And $Max2$ module values zero array element .
5. IN cycle from 1 up to $N-1$. If the modulus of an array element is greater than $Max1$, then $Max1$ is stored in $Max2$, and the modulus of the array element is stored in $Max1$. Otherwise, if the modulus of the array element is greater than $Max2$, then the modulus of the array element is stored in $Max2$.
6. Initiate variable j .
7. IN cycle from 0 to $N-1$. If meaning element more zero, That copy it to the element with index j . Increase j by 1 .
8. IN cycle from j to $N-1$. All elements equate zero.
9. Withdraw received array And values $Max1$, $Max2$ And amounts.

Description input And weekend data

Since the type of the array elements is defined as real, the type of the variables used in the calculations is also defined as real (float).

Listing programs

```
# -*- coding: cp1251 -*-
from math import *
from random import *
n = int(input(" Elements V array (N<=30) N: "))
if n > 30: n = 30
elif n < 5: n = 5
amax = float(input("Threshold meaning A: "))
# Generating an array and output
print(" Initial state ") mas =
[]
for i in range(n):
    mas.append(uniform(-5, 5))
    print("{0: 7.3f}".format(mas[i]), end=" ")
print()
# Finding amounts
# Reset elements exceeded threshold asum =
0.0
for i in range(n): # from 0 to
    n-1 if abs(mas[i]) < 1.0:
        asum = asum + mas[i]
    if abs(mas[i]) > amax:
        mas[i] = 0.0
# Search maximum element max1
= abs(mas[0])
max2 = abs(mas[0])
for i in range(1,n):
    if max1 < abs(mas[i]):
        max2 = max1
        max1 = abs(mas[i])
    else:
        if max2 < abs(mas[i]):
            max2 = abs(mas[i])
j = 0
for i in range(s): # Sorting array if
    abs(abs[i]) > 0.00001:
        mass[j] =
        mas[i] j = j +
        1
for i in range(j,n): # from j to n-1
    mas[i] = 0.0
```

```

print(" Final state ")
for i in range(n): # Array after sorting
    print("{0: 7.3f}".format(mas[i]), end=" ")
print()
print("max1={0:7.3f} max2={1:7.3f} sum={2:7.3f}"
      .format(max1, max2, asum))

```

Result work programs

```

Elements V array(N<=30) N: 6
Threshold value A: 2.8 Initial
state
-2.655    2.194    1.034    2.804   -1.490   - 0.474
The final state
-2.655    2.194    1.034   -1.490   - 0.474    0.000
max1=  2.655 max2=  2.655 sum= - 0.474

```

Exercise To practical work No. 4

"One-dimensional arrays»

Option 1

IN one-dimensional array, consisting from n material elements, calculate:

1. Amount negative elements.
2. The product of the elements located between the maximum and minimum elements. Sort the array elements in ascending order.

Option 2

IN one-dimensional array, consisting from n material elements, calculate:

1. Amount positive elements.
2. The product of the elements located between the maximum and minimum absolute values. Sort the array elements in descending order.

Option 3

In a one-dimensional array consisting of n integer elements, calculate:

1. Work elements With even numbers.
2. The sum of the elements located between the first and last zero elements. Transform the array so that all positive elements appear first, followed by all negative elements (elements equal to zero are considered positive).

Option 4

IN one-dimensional array, consisting from p material elements, calculate:

1. Amount elements With odd numbers.
2. The sum of the elements located between the first and last negative elements. Compress the array by removing all elements whose absolute value does not exceed one. Fill the remaining elements at the end of the array with zeros.

Option 5

IN one-dimensional array, consisting from p material elements, calculate:

1. Maximum element array.
2. The sum of the elements located before the last positive element. Compress the array by removing all elements whose absolute value is in the interval $[a, b]$. Fill the vacated elements at the end of the array with zeros.

Option 6

IN one-dimensional array, consisting from p material elements, calculate:

1. Minimum element array.
2. The sum of the elements located between the first and last positive elements. Transform the array so that all elements equal to zero are listed first, followed by all others.

Option 7

In a one-dimensional array consisting of n integer elements, calculate:

1. Number maximum element array.
2. The product of the array elements located between the first and second zero elements. Transform the array so that The first half contained elements in odd positions, and the second half contained elements in even positions.

Option 8

IN one-dimensional array, consisting from p material elements, calculate:

1. Number minimum element.
2. The sum of the elements located between the first and second negative elements. Transform the array so that first were located All elements, module which Not exceeds one, and then all the rest.

Option 9

IN one-dimensional array, consisting from p material elements, calculate:

1. Maximum By module element.
2. The sum of the elements located between the first and second positive elements. Transform the array so that the elements equal to zero are located after all others.

Option 10

In a one-dimensional array consisting of n integer elements, calculate:

1. Minimum By module element.
2. The sum of the moduli of the elements located after the first element equal to zero.

Transform the array so that its first half contains elements, standing V even positions, A in second half - elements, standing V odd positions.

Option 11

IN one-dimensional array, consisting from p material elements, calculate:

1. Number minimum By module element.
2. The sum of the moduli of the elements located after the first negative element.

Compress the array by removing all elements whose values are in the interval $[a, b]$. Fill the remaining elements at the end of the array with zeros.

Option 12

IN one-dimensional array, consisting from p material elements, calculate:

1. Number maximum By module element.
2. The sum of the elements located after the first positive element.

Transform the array so that first all elements whose integer part lies in the interval $[a, b]$ are arranged, and then all the others.

Option 13

IN one-dimensional array, consisting from p material elements, calculate:

1. Quantity elements array, lying V range from A to IN .
2. The sum of the elements located after the maximum element. Sort the array elements in descending absolute value.

Option 14

IN one-dimensional array, consisting from p material elements, calculate:

1. Quantity elements array, equal zero.
2. Amount elements, located after minimum element. Sort the array elements in ascending order of absolute values.

Option 15

IN one-dimensional array, consisting from p material elements, calculate:

1. Quantity elements array, large *WITH*.
2. Work elements, located after maximum By module of the element.

Transform the array so that all negative elements are listed first, followed by all positive elements (elements equal to zero are considered positive).

Option 16

IN one-dimensional array, consisting from p material elements, calculate:

1. Quantity negative elements.
2. The sum of the moduli of the elements located after the element with the minimum modulus.

Replace all negative array elements their squares and sort the array elements in ascending order.

Option 17

IN one-dimensional array, consisting from p material elements, calculate:

1. Quantity positive elements.
2. Amount elements, located after last element, equal to zero.

Transform the array so that all elements whose integer part does not exceed one are listed first, and then all the others.

Option 18

IN one-dimensional array, consisting from p material elements, calculate:

1. Quantity elements array, smaller WITH.
2. Amount integer parts of array elements, located after the last negative element.

Transform the array so that all elements that differ from the maximum by no more than 20% are listed first, and then all the rest:

Option 19

IN one-dimensional array, consisting from p material elements, calculate:

1. Work negative elements.
2. The sum of positive elements located before the maximum element.

Change order following elements V array on back.

Option 20

IN one-dimensional array, consisting from p material elements, calculate:

1. Work positive elements.
2. The sum of the elements located before the minimum element. Sort the elements in even places and the elements in odd places in ascending order.

Option 21

IN one-dimensional array, consisting from p whole elements, calculate:

1. Maximum element array.
2. The sum of the remainders obtained from dividing the array elements by the maximum element.

Arrange the elements of the array in ascending order of the remainders obtained from dividing them by the maximum element.

Option 22

IN one-dimensional array, consisting from p material elements, calculate:

1. Average meaning, How attitude amounts everyone elements on their quantity

$$\bar{a} = \frac{1}{n} \cdot \sum_{i=1}^n a_i.$$

2. Amount squares differences between elements array And received

$$\text{average value } S = \sum (a_i - \bar{a})^2.$$

Sort the elements of an array in descending order of their difference with the mean value $(a_i - \bar{a})$: at the beginning arrange elements With positive maximum difference, and then the rest in descending order of difference.

Option 23

IN one-dimensional array, consisting from p material elements, calculate:

1. Second By size element.
2. Amount elements, located between maximum And second By size of elements.

Arrange the elements of the array in ascending order of the remainders obtained from dividing them by the second largest element.

Option 24

IN one-dimensional array, consisting from p material elements:

1. Calculate maximum And minimum elements (Max And Min).
2. Insert a new value into an element that is located midway between the maximum and minimum elements. To insert, perform the following: request a new value that must be in the range ($Min \leq a \leq Max$); move the array elements one position to the right from the insertion point (the last element is lost); assign a new value to the released element.

Sort by increasing absolute value of the difference between the array elements and the new value: $|a_i - a|$

Option 25

Using the *Random module* , form a one-dimensional array consisting of n real elements in which the elements take a random value, and the condition is met: $a_1 < a_2 < a_3 < \dots < a_n$.

1. For given numbers y , such, What $a_1 < y < a_n$, define such k , to $a_k < y < a_{k+1}$.
2. Calculate amount elements array, located to element ak . Sort the elements after element ak in descending order .

Option 26

Using the *Random module* , form a one-dimensional array consisting of n real elements in which the elements take a random value, and the condition is met: $a_1 > a_2 > a_3 > \dots > a_n$.

1. For given numbers y , such, What $a_1 < y < a_n$, define such k , to $a_k < y < a_{k+1}$.
2. Calculate amount elements array, located after element a_k . Sort the elements before a_k in descending order. .

Option 27

Using the *Random module* , form a one-dimensional array consisting of n real elements in which the elements randomly take a positive or negative sign and a value from -5 to 5. For a given number y such that $amin < y < amax$, calculate:

1. Amount elements array, values module which less y .
2. Work the rest elements.

Option 28

Using the *Random module* , create a one-dimensional array consisting of n real elements in which the elements randomly take a positive or negative sign and a value from -10 to 10. For a given number y such that $amin < y < amax$, calculate:

1. Work elements array, values module which greater than y .
2. Amount modules the rest elements.

Option 29

The coordinates of n points are specified as elements of a one-dimensional array. Odd elements are the ordinate values, and even elements are the abscissa values. Calculate the maximum distance between the closest points.

Output the coordinates of all points that fall within a circle with center coordinates $X0, Y0$ and radius R . Sort the points' positions in ascending order of their ordinates.

Option 30

The coordinates of n points are specified as elements of a one-dimensional array. Odd elements are the ordinate values, and even elements are the abscissa values. Calculate the minimum distance between the closest points.

Output the coordinates of all points that fall within the ring with center coordinates $X0, Y0$, inner radius $R1$ and outer radius $R2$.

Organize position points By increase their abscissa.

Practical Job No. 5: "Two-dimensional arrays and functions"

Objective of the work:

To give students practical skills in writing programs for processing two-dimensional arrays using functions.

Production tasks

Design two-dimensional array material numbers.

1. Initiate elements random numbers V range $(-10 \text{ :- } 10)$.
2. Calculate average And dispersion (D) By to everyone elements array.
3. Replace, on average meaning, those elements array, deviation which from average exceeds σ , Where $\sigma = \sqrt{D}$.
4. Points tasks design V view functions.

Theoretical introduction

Organization two-dimensional arrays

In Python, you can implement an array using nested lists. The following program listing demonstrates how to create a two-dimensional list and initialize its elements.

Note the line highlighted in bold. The array element, previously assigned a numeric value, is initialized with a string value. Python performs this operation without issuing a warning (list elements can be of different types). This is done here for illustration purposes only.

Listing programs

```
from random import *
n=int(input("Enter size sq. matrices (N): "))
A=[] # Empty list
B=[]
for i in range(n):
    A.append([0]*n) # Nested list
                    # initiated 0
    B.append([3]*n) # Nested list
                    # initiated 3
print(A) # Conclusion
print(B) list
print() # Just "Enter"
for Row in range(n): # By lines And
    for Col in range(n): # By columns
        A[Row][Col]=randint(0.99) # random
                                   # number
for Row in range(n): # Conclusion
    result
    for Col in range(n):
```

```

        print("{0:02}".format(A[Row][Col]),
              end=" ")
    print()
print()
#
A[1][2] = 'AB'          # A this is a "
surprise " for i in range(n): # Conclusion
result
    for j in range(n):
        if ((i==1) and (j==2)):
            print(A[1][2], end=" ")
        else:
            print("{0:02}".format(A[i][j]),
                  end=" ")
print()

```

Result work programs

```

Enter size matrices (NxN): 4 [[0,
0, 0], [0, 0, 0], [0, 0, 0]]
[[3, 3, 3], [3, 3, 3], [3, 3, 3]]

```

```

58 75 20
17 99 45
26 49 73

```

```

58 75 20
17 99 AB
26 49 73

```

Let's note next moments, concerning such implementation code:

- a list element can be an object of any type, this reduces the efficiency of the code;
- in the case of processing arrays (a set of identical elements) More efficient code should be used.

Arrays are frequently used in scientific computing. Examples include time series of meteorological observations and matrices used to solve systems of equations.

As already noted, Python is an object-oriented language. In this language, arrays are objects with their own attributes and methods.

So, For receipt dimensions array Can use the `ndim` attribute:

```
dim = Mymas.ndim
```

Quantity elements By each from coordinates returns attribute

shape :

```
a, b = Mymas.shape
```

Attribute `size` returns quantity elements V array:

```
n = Mymas.size
```

Function `len()` allows get quantity elements V line:

```
m = len(Mymas)
```

In order to improve the efficiency of the code and implement additional methods and functions used in array processing, Python has developed various modules. For example, big The ability to work with multidimensional arrays is provided by the **NumPy** (**numpy**) module, which is proposed to be used to solve the problems in our tutorial.

The following program listing demonstrates how to generate matrices and print matrices to the screen using the **numpy module** .

Listing programs

```
import numpy as np

n=int(input("Enter size matrices (NxN): ")) #
Zero matrix
Z=np.zeros((n, n), int) # Row, Colum, type
print(Z)
print()
# Diagonal single matrix
E=np.eye(n)
print(E)
print()
# Matrix with random set values of A =
20*np.random.random(size=(n,n)) - 10 for
Row in range(n):
    for Col in range(n):
        print("{0:>5.2f}"
                .format(A[Row][Col]), end=" ")
    print()
print()
# Initialization
matrices for Row in
range(n):
    for Col in range(n):
        A[Row][Col]=Row*10+Col
for Row in range(n):
    for Col in range(n):
        print("{0:>05.2f}"
                .format(A[Row][Col]), end=" ")
    print()
```

Result work programs

Enter size matrices (NxN): 4

```
[[0 0 0 0]
 [0 0 0 0]
 [0 0 0 0]
 [0 0 0 0]]
```

```
[[ 1.  0.  0.  0.]
 [ 0.  1.  0.  0.]
 [ 0.  0.  1.  0.]
 [ 0.  0.  0.  1.]]
```

```
-0.09 -1.68  5.82  0.47
-5.72 -5.65 -5.96 -2.20
 6.20  1.72 -7.72 -5.56
 8.60  0.18  8.39  4.11
```

```
00.00 01.00 02.00 03.00
10.00 11.00 12.00 13.00
20.00 21.00 22.00 23.00
30.00 31.00 32.00 33.00
```

When solving problems, automate the process of generating values for two-dimensional arrays rather than entering them manually. Avoid using ready-made methods and functions that obscure the solution algorithm, such as finding the maximum or minimum.

To solve our task, see the problem statement above, initialization elements matrices we will do it function, which generates pseudo-random numbers of real type from the NumPy module.

Some functions of the NumPy module generate a pseudorandom number in the range $[0, 1)$ (including zero and excluding 1), for example, `random()`, `randf()`, `sample()`. When generating pseudorandom numbers in the range $[a, b)$, where $b > a$, the following expression should be used:

$$y = a + (b - a) * \text{Rand},$$

Where Rand - pseudo-random number, generated one from functions listed above. An example is presented in the program above.

To calculate the mean, variance, and standard deviation, we use the following formulas:

$$\text{Average: } A = \frac{\sum_{i=1}^n \sum_{j=1}^n a_{ij}}{n^2}; \quad \text{Dispersion: } D = \frac{\sum_{i=1}^n \sum_{j=1}^n (a_{ij} - A)^2}{n^2 - 1}; \quad \sigma = \sqrt{D}.$$

We will formulate the matrix initialization, calculation of the mean and variance, and matrix correction as functions.

Functions

In a program, a frequently repeated fragment of code can be represented as a separate, named structure. In computer science terms, such a structure is usually called a subroutine.

There are two types of subroutines: procedures and functions. Procedures and functions can return their results through parameters. Furthermore, functions always return their results through their own names and, unlike procedures, can be used in expressions.

Python uses only the concept of "function." A function object is created with the `def` statement, followed by the function name and, optionally, parameters, which are specified in parentheses. The function header ends with a colon. The function body is formed by a block of statements that follow and are indented four spaces by default. If there are no statements in the function body, for example during testing, a `pass` statement is written, which does nothing. The function exits after the last statement of the function body is executed. This statement can, but is not required, be a `return` statement. In this case, the function, through its name, returns the object specified after the `return` statement. Such a function can be used on the right-hand side of expressions. If the `return` statement is not used, the function returns the value `None`, which can also be analyzed in a conditional expression.

Functions can be nested within other functions, but this is not recommended, see, for example, M. Lutz, *Learning Python*, 2011.

Note : Variables used in declaring a function are usually called parameters, and variables used when calling a procedure are called arguments.

There are several styles for formatting programs with subroutines. In some, the subroutine description precedes the program description itself; in others, the subroutine description follows the program description (with only the subroutine header preceding the program); and in still others, subroutines are placed in separate modules that are connected to the program using special instructions: `uses` (Pascal), `include` (C), `import` (Python). In all styles, measures are taken to ensure that by the time a subroutine is called in a program, its name (the name of the subroutine) is known.

For this purpose, in Python, it is recommended to declare all functions in the program's head. Function declarations follow the following structure:

```
def <FunctionName> ([Parameters]) :  
    ['''DocumentationString''']  
    <Function_body>  
    [return <Return_Result>]
```

Here <Function_Name> – This is the name that will be used when calling a function in a program. Parameters may appear in parentheses after the function name. Essentially, a parameter declaration is a declaration of variables used in the function body to execute various instructions involving them. Parameters, on the other hand, are variables through which a function receives data from the program or returns new data to the program.

Name functions should be unique And correspond requirements for variable naming.

Comment O sequences descriptions functions in Python:

When calling a function (f2()) from another function (f1()), the function f2() itself can be declared after f1() . The only important thing here is that the call to f1() is made after the declaration of f2() . This is due to the fact that in Python, the byte code is generated first (this is the interpreter) and only then the Python virtual machine operates. Thus, by the time the code is executed (the call to f1()), the placement of f2() is known and the code can be processed without problems.

Example:

```
def f1(y):
    f2(y) # Leading link

def f2(x): # Summoned print(x)
    function

f1(56) # Call functions
```

Description algorithm

Let's consider algorithm solutions ours tasks.

1. Request the size of the array N. Since the shape of the array is not defined, we will solve the problem for a square matrix (NxN).
2. Make an array - initialize it with a set of pseudo-random real numbers (function MakeMatr()).
3. Withdraw received matrix (function PrintMatr ()
4. Calculate average meaning And dispersion (function MidlDisp()).
5. Execute adjustment elements (function CorrectMatr ()).
6. Withdraw corrected matrix

Description input And weekend data

The program requests the array size. The program's output is visualized on the monitor screen. The matrix element type is set to float , as specified.

Description subroutines

Function MakeMatr(n, a, b)

The function initializes a square matrix with pseudorandom numbers in the range $[a, b)$. The matrix size is $n \times n$. elements.

Returns object type array .

Function MidlDisp(Matr)

The function calculates the average value of the elements of the Matr array. and dispersion.

The function returns a tuple object where the first element is the mean and the second element is the variance.

Function CorrectMatr (Matr)

The function adjusts the Matr array by replacing the element values with the average if the absolute value in the array exceeds the standard deviation.

Function returns meaning type array .

Function PrintMatr (Matr)

This function outputs array on screen monitor.

Listing programs

```
import numpy as np
from math import *

def MakeMatr(n, a, b):
    """ Initialization of a square matrix of
    size NxN pseudo-random values in the
    range [a,b) """
    Matr = (b-a)*np.random.random(size=(n,n)) + a
    return Matr

def MidlDisp(Matr):
    """ Calculation average values
    Calculating the variance """
    sum = 0
    (nRow, nCol) = Matr.shape # Размеры
    Measurement for Row in range(nRow):
        for Col in range ( nCol )
            : sum +=
                Matter[Row][Col]
    Midl = sum / Matrix size # Size
    Disp =
    for Row in range(nRow):
        for Col in range(nCol):
            Disp += (Matr[Row][Col] - Midl)**2
    return (Midl, Disp / (Matr.size - 1)) # Коптеж

def CorrectMatr(Matr, avr, sigma):
```

```

    """ Replacing " bounced " values """ (nRow,
nCol) = Matr.shape # Dimensions matrices
for Row in range(nRow):
    for Col in range(nCol):
        if abs(abs(Mat[Row][Col]) - avr) > sigma:
            Matr[Row][Col] = avr
return(Mat)

def PrintMat(Mat):
    """ Печать матрицы """
    (nRow, nCol) = Mat.shape
    for Row in range(nRow):
        for Col in range(nCol):
            print("{0: 7.3f}"
                .format(Mat[Row][Col]), end=" ")
        print()
    print()
n=int(input(" Enter size matrices (NxN): "))
MyMat=MakeMat(n, -5, 5) # Make matrix
PrintMat(MyMat)
(mMidl, mDisp) = MidlDisp(MyMat) # Average And
dispersion print("Midl = {0:7.3f}".format(mMidl))
(print("Disp = {0:7.3f} Sig = {1:7.3f}"
    .format(mDisp, sqrt(mDisp))))
# Adjustment
NMat = CorrectMat(MyMat, mMidl, sqrt(mDisp))
PrintMat(NMat)

```

Result work programs

Enter size matrices (NxN): 7

-3.775	-3.782	-1.127	4.921	-3.295	1.567	-0.084
-1.091	-4.529	3.789	-4.245	3.748	4.199	-0.593
-4.977	4.779	1.184	-0.448	-1.605	3.030	-4.608
0.076	-1.509	-3.713	-3.791	-4.139	4.570	3.234
2.997	-3.789	1.384	-2.880	-2.572	-2.550	-3.275
-2.468	-1.418	-0.306	2.019	4.532	2.518	1.570
-2.188	4,122	-0.271	2,883	-3.256	4,610	-1.648

Means -0.249

= Disp 9,902 Say = 3,147

=

-0.249	-0.249	-1.127	- 0.249	-0.249	1,567	-0.084
-1.091	-0.249	-0.249	-0.249	-0.249	-0.249	-0.593
-0.249	-0.249	1.184	-0.448	-1.605	-0.249	-0.249
0.076	-1.509	-0.249	-0.249	-0.249	-0.249	-0.249
-0.249	-0.249	1.384	-2.880	-2.572	-2.550	-0.249
-2.468	-1.418	-0.306	2.019	-0.249	2.518	1.570

-2.188 -0.249 -0.271 2.883 -0.249 -0.249 -1.648

Exercise To practical work No. 5

"Two-dimensional arrays And functions»

The dimensions of two-dimensional arrays should be requested from user. All necessary data must be passed to functions as parameters. All variables used only within the function must be declared as local. The use of global variables in functions is not allowed. Provide output of both the original array and the array obtained as a result of the program's operation, where this is possible according to the conditions of the problem.

Points tasks design V view functions.

Option 1

Dana integer rectangular matrix. Define:

1. Quantity lines, Not containing no one zero element.
2. Maximum meaning from numbers, encountered V given matrix more than once.

Option 2

Dana integer rectangular matrix.

1. Define quantity columns, Not containing no one zero element.
2. By rearranging the rows of a given matrix, arrange them in accordance with growth of characteristics.

NOTE: Characteristics lines integer matrices let's call the sum of its positive even elements.

Option 3

Dana integer rectangular matrix. Define:

1. Quantity columns, containing Although would one null element.
2. Number lines, V which located the most long series identical elements.

Option 4

Dana integer square matrix. Define:

1. The product of the elements in those rows that do not contain negative elements.
2. Maximum among sums elements diagonals, parallel main diagonal of the matrix.

Option 5

Dana integer square matrix. Define:

1. Amount elements V those columns, which Not contain negative elements.
2. Minimum among sums modules elements diagonals, parallel to the secondary diagonal of the matrix.

Option 6

Dana integer rectangular matrix. Define:

1. The sum of the elements in those rows that contain at least one negative element.
2. Numbers lines And columns everyone saddle points matrices.

NOTE: Matrix **A** has a saddle point a_{ij} if a_{ij} is the minimum element in the i -th row and the maximum in the j -th column.

Option 7

For a given matrix of size 8×8 , find k such that the elements of the k -th row of the matrix coincide with the elements of the k -th column.

Find the sum of the elements in those rows that contain at least one negative element.

Option 8

By rearranging the columns of a given matrix, arrange them in accordance with the growth of characteristics.

Find amount elements V those columns, which contain Although would one negative element.

NOTE: Characteristics column integer matrix we will call the sum of the moduli of its negative odd elements.

Option 9

Neighbors element a_{ij} V matrix Let '**s**' call it such elements a_{kl} , for which the following conditions are met : $i-1 \leq k \leq i+1$, $j-1 \leq m \leq j+1$, $(k, m) \neq (i, j)$.

These are all the elements that surround the central element horizontally, vertically, and diagonally. The leftmost elements have no neighbors to the left, and the rightmost elements have no neighbors to the right.

$$\begin{pmatrix} \begin{matrix} \mathbf{a}_{1,1} & \mathbf{a}_{1,2} \\ \mathbf{a}_{2,1} & \mathbf{a}_{2,2} \end{matrix} & \dots & a_{1,j-1} & a_{1,j} & a_{1,j+1} & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{i-1,j-1} & a_{i-1,j} & a_{i-1,j+1} & \dots \\ \begin{matrix} a_{i,j-1} & \mathbf{a}_{i,j} & \mathbf{a}_{i,j+1} \\ \mathbf{a}_{i+1,j-1} & \mathbf{a}_{i+1,j} & \mathbf{a}_{i+1,j+1} \end{matrix} & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \end{pmatrix}$$

In this matrix, the neighbors of the corner element are a_{11} and element a_{ij} highlighted in bold.

The matrix smoothing operation produces a new matrix of the same size, each element of which is obtained as the arithmetic mean existing neighbors of the

corresponding element of the original matrix.

Construct the smoothing result of a given 10×10 real matrix.

In a smoothed matrix, find the sum of the moduli of the elements located below the main diagonal.

Option 10

Element matrices it's called local minimum, If He is strictly smaller than all its neighbors. The concept of neighbors is given in Option 9.

Calculate the number of local minima of a given 10 x 10 matrix .

Find the sum of the moduli of the elements located above the main diagonal.

Option 11

The coefficients of a system of linear equations are given as a rectangular matrix. Using permissible transformations, convert the system to triangular form.

Find the number of rows whose element mean is less than a given value.

Option 12

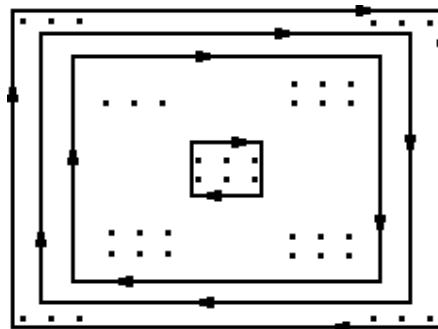
To implement cyclic shift of elements rectangular matrix by n elements to the right or down (depending on the mode entered), n can be greater than the number of elements in a row or column.

NOTE: Cyclic shift of elements means rearranging the elements of a row or column according to the following rule:

1, 2, 3, ...n	-> n, 1, 2, 3, ...,n-1	-> n-1, n, 1, ...n-2	-> n-2, n-1, n, 1, 2, ...n-3
Initial state	right shift by 1 element	by 2 elements	3 .

Option 13

Perform a cyclic shift of the elements of a matrix of size $m * n$ (m lines x n columns). The shift is performed to the right by k elements in the following manner: the elements of the first row are shifted to the last column from top to bottom, from there to the last row from right to left, from there to the first column from bottom to top, from there to the first row; for the remaining elements, the process is similar.



Option 14

Compact a given matrix by removing rows from it and columns filled with zeros.
Find the number of the first row containing at least one positive element.

Option 15

Given an integer rectangular matrix, determine the number of the first column containing at least one zero element.

By rearranging the rows of a given matrix, arrange them in descending order of characteristics.

NOTE: The characteristic of a row of an integer matrix is the sum of its negative even elements.

Option 16

Arrange the rows of an integer rectangular matrix in ascending order of the number of identical elements in each row.

Find the number of the first column that does not contain any negative elements.

Option 17

By rearranging the elements of a square real matrix, ensure that its largest element is in the upper left corner (1,1), the next largest is in position (2, 2), the next largest is in position (3, 3), and so on, thus filling the entire main diagonal.

Find the number of the first row that does not contain any positive elements.

Option 18

Dana integer rectangular matrix. Define:

1. Quantity lines, containing Although would one null element.
2. Number column, V which located the most long series identical elements.

Option 19

Dana integer rectangular matrix. Define:

1. Quantity negative elements V those lines, which contain at least one zero element.
2. Numbers lines And columns everyone saddle points matrices.

NOTE:

Matrix **A** has a saddle point a_{ij} if a_{ij} is the minimum element in the i -th row and the maximum in the j -th column.

Option 20

Write a program that swaps the columns of a square matrix containing the largest and smallest elements and calculates the sum of the elements of the main diagonal.

Option 21

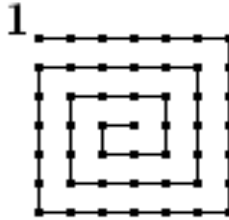
Dana integer square matrix. Define:

1. Amount elements V those lines, which Not contain negative elements.
2. Minimum among sums elements diagonals, parallel main diagonal of the matrix.

Option 22

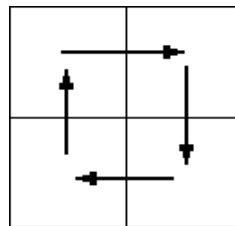
Write a program that generates a square matrix whose elements are natural numbers arranged in ascending order. from 1 up to n^2 (n - order matrices) according to scheme, given in the picture.

Calculate the sum of the elements located on the main diagonal of the resulting matrix.



Option 23

A square matrix of order $2n$ consists of 4 blocks. Write a program that generates a new matrix by rearranging the blocks of the original matrix according to the diagram and prints the sum of the elements of each block.



Option 24

Given a square matrix with integer elements, write a program that converts the columns of the given matrix into rows of a new matrix.

For each value of a matrix element, count the number of elements that take the same value. Count only those values that are present in the matrix.

Option 25

Given two matrices of the same order $m * n$ (m rows n columns). Write program additions, subtractions And transposition matrices.

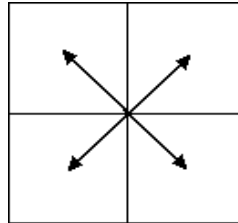
1. Addition And subtraction: $c_{ij} = a_{ij} \pm b_{ij}$

2. Transposition $b_{ij} = a_{ji}$

Option 26

Square matrix order $2 \times n$ consists of from 4-x blocks. Write a program that:

- forms new matrix, rearranging blocks original matrices according to the diagram;
- prints work elements each block.



Option 27

Write program, which fills matrix dimensions 4×13 numbers from 1 to 52 randomly: repetition of numbers is not allowed. Received result withdraw on screen monitor.

Option 28

The work two matrices **A** on **B** it's called such matrix **C** , For

$$\text{which : } c_{ik} = a_{i1} \cdot b_{1k} + a_{i2} \cdot b_{2k} + \dots + a_{in} \cdot b_{nk} = \sum_{j=1}^n a_{ij} \cdot b_{jk} .$$

That is, the element c_{ik} matrix **C** is equal to the sum of the products of the elements of the i-th row of matrix **A** by the corresponding elements of the k-th column of matrix **B**.

Write program calculations works two matrices.

Program should By given dimensions matrices report about the possibility of obtaining such a work.

Option 29

The exponential of a (square) matrix **A** can be defined as follows:

$$\text{by } \mathbf{A} = \mathbf{I} + \mathbf{A} + \frac{1}{2!} \mathbf{A}^2 + \frac{1}{3!} \mathbf{A}^3 + \dots ,$$

Where **I** - single matrix, at which elements main diagonals equal

$$\text{unit, A rest - zero: } and = \begin{cases} 1 & m = n \\ 0 & m \neq n \end{cases}, \quad \mathbf{A}^2 = \mathbf{A} \times \mathbf{A}.$$

The sum of matrices **A** and **B** of the same size is a matrix **C** of the same size With elements $c_{ik} = a_{ik} + b_{ik}$ at everyone i And k. Like this in this way, Addition of matrices of the same size is performed element by element.

The product of a matrix by a real number λ will be a matrix in which all elements are multiplied by λ : $\mathbf{C} = \lambda \cdot \mathbf{A} = |\lambda \cdot a_{ik}|$.

The work two matrices **A** on **B** it's called such matrix **C** , For

which :

$$c_{ik} = a_{i1} \cdot b_{1k} + a_{i2} \cdot b_{2k} + \dots + a_{in} \cdot b_{nk} = \sum_{j=1}^n a_{ij} \cdot b_{jk} .$$

That is, the element c_{ik} of the matrix **C** is equal to the sum of the products of the elements of the i-th row matrix **A** into the corresponding elements of the k-th column of matrix **B**.

Write a program that calculates the exponential of a matrix by summing the first n elements of the series.

Example performances single matrices, matrices 2x2 matrix dimensions and calculation of the square of the matrix:

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}, \quad A^2 = \begin{pmatrix} 1 \cdot 1 + 2 \cdot 2 & 1 \cdot 2 + 2 \cdot 4 \\ 3 \cdot 1 + 4 \cdot 3 & 3 \cdot 2 + 4 \cdot 4 \end{pmatrix} = \begin{pmatrix} 5 & 10 \\ 15 & 22 \end{pmatrix}$$

Вариант 30

There is kit symbols 0..9 And A..Z (total 36 characters). Write a program that:

- randomly fills the elements of a two-dimensional 6x6 matrix . Repeated characters are not allowed;
- outputs a sentence that contains no more than 10 words, composed according to the following principle:
 - random way is being asked number, which is column or row number;
 - elements lines (column) I am forming word.

Practical Job No. 6: Files

Objective of the work:

To give students practical skills in writing programs that perform operations with text files – reading, writing.

Production tasks

In the previous tasks, the data required for the program was entered from the keyboard, and the result was displayed on the monitor screen. Obviously, What And at debugging programs, And at input big volume This approach is labor-intensive when entering data into a program and is therefore unsuitable. The most suitable solution is to input data from a file and output the results to a file. This way, the input data is prepared once and with the required diversity, while the results can be analyzed repeatedly.

Let's write several programs that will read input data from a pre-prepared text file and output the result. into a text file. For this purpose, we'll use the programs written in the previous tasks.

Theoretical introduction

File type data introduced V languages programming For work with external devices – files on disk, input/output ports, printers, etc. File types are divided into text and binary .

A text file contains data of the string type (`str`), while a binary file contains data of the `bytes` type .

File access can be sequential or direct. With sequential access, each subsequent element can be An element is read only after a similar operation has been performed on the previous element. With direct access, a read (write) operation can be performed on any element with a given address.

Text files are sequential access files. They are designed to store string-type information. Input and output of information are accompanied by data type conversion. When output to a text file, data is converted from its internal representation to characters, and when input, the reverse occurs. transformation.

For example, if some integer variable in memory stores the value 731 ₁₀ in two bytes , then:

- V binary presentation This will: 0000 0010 1101 1011 ₂;
- V text: 37 33 31 _{string} .

Binary files are directly accessible files. They store information in binary form. When writing or reading a binary file, the information does not undergo any additional transformation.

General approach To work With files, For everyone languages programming is based on the following concepts:

- the operating system has a file subsystem that ensures reliable work with files in multi-user and multi-tasking modes when working with various external media: hard drives, flash drives, DVDs, ...;
- The operating system provides a set of functions (API functions) that allow you to work with files at a logical level. The programmer is not responsible for managing external devices, monitoring file records in folders, or monitoring recording quality. etc.

To organize work with files, when programming in a high-level language, four steps are typically performed:

- A file object is created. This is done using subroutines that associate the file name specified by the user with a variable that stores a reference to a structure created specifically by the operating system. This structure contains information about the file, the data buffer through which the exchange between the program and the file will take place, and the current state of the data exchange process;
- the exchange mode in which the work with the file will take place is set: will be whether This mode readings, records, additions or Which or combined mode, such as reading and writing. This step is implemented either after creation of a file object, or during the execution of the first step;
- data is being written or read. The process of exchanging data between a program and a file consists of exchanging data between the program and a data buffer managed by the file subsystem. When writing data to the buffer, the file subsystem monitors the writing process and, when the buffer is filled to a certain level, writes the data to the file, and the buffer is cleared, allowing the program to continue writing. When reading data from the buffer, the file subsystem monitors the amount of data in buffer and, if necessary, downloads fresh data from the file;
- The file close operation is performed. The remaining data in the buffer is written to the file, and the file is closed.

A file close operation is mandatory if the file was opened for writing. If the file is not closed, the resources allocated by the operating system will be closed when the program terminates. In this case, the file may be empty (most common) or may not contain all the information that was attempted to be written to it. It depends on the buffer size, which in modern OSs can be quite large.

Files opened for reading also need to be closed. There are two reasons for this:

- a file opened for reading is locked and other applications cannot access it;

– Both the program and the operating system have a limit on the number of open files.

And although, in your opinion, access to the file from other applications is unlikely, and the number of open files is quite large, you should not tempt fate, only to receive an error message and then spend a long time and painfully searching for its cause.

In an object-oriented programming language, such as Python, some of the steps described can be implemented in a hidden form.

For example, when creating a file object, the first and second steps are performed in one statement:

```
fh = open( < File name > [, mode = <mod>])
```

Where fh - variable, storing link on file object, <File_name>

– absolute or relative path and file name, mode=<mod> – the mode in which the file is opened: writing, reading, adding, ...

The Python language supports the context manager protocol. This protocol ensures that a file is closed correctly, regardless of whether an exception (error) occurs within a code block. For example, the following code opens a file for writing, writes strings to the file, closes the file, then reopens it, displays text to the screen, and closes the file. Note that the code does not explicitly close the file:

```
with open(r"lab6.txt", "w", encoding="cp1251") as fh:
    fh.write("Mercury\n") # Write lines to file
    fh.write("Venus\n")
    fh.write("Earth\n")
# IN this place file fh closed automatically
with open(r"lab.6.txt", "r", encoding="cp1251") as fh:
    print(fh.read())
# IN this place file fh So it's closed
```

When creating a file object, you must specify the path and name of an existing or future file. The file path can be specified either relative to the current working directory or absolutely. A relative path is defined as a path relative to the current working directory. folders, A under current working folder understood folder, V which the user is located V moment launch file (launched file Maybe be in another folder).

To properly understand how a path can be formed, one should refer to literature, for example [1], and conduct independent experiments.

Below are modifications of the programs written for Labs #1 and #5. These modifications provide methods for working with files.

Addition To practical work No. 1

IN this work We studied record expressions on language Python.
Let's make a contribution following change V ours program:

- let's prepare text file With original data;
- Using the instructions for working with a text file, we will read the recorded lines and perform calculations;
- result calculations let's write it down V text file V view tables.

Reminder : When writing a program, you can spread a long instruction across multiple lines. To do this, use the backslash character, immediately followed by Enter or, for example, parentheses or square brackets (preferred).

Computed expressions we'll arrange it V view functions:

```
def f1(a,x):
    y = (tan(x**2/2-1)**2+(2*cos(x- p/6))
        / (1/2+sin(a)**2))
    return y

def f2(x):
    y = pow(2, log(3- cos(pi/4+2*x), 3+sin(x))
        / (1+tan(2*x/p)**2))
    return y
```

Textual file

Let's install it next format text file:

- two lines - This cap, V which indicated appointment columns. These We will provide the lines with a comment symbol, which is used in Python: ' # ';
- two column - This data, For which will be be carried out calculations.

Example:

```
# a      x
#-----
- 2      - 2
0        - 2
...
```

We use simplified diagram work With text file.

Create file object :

```
fh = open(<File_name> [, mode = <mod>]) ,
```

Where fh - variable, storing link on being created file object,

<file_name> is the absolute or relative path and file name, mode=<mod> is the mode in which the file is opened. <mod> is replaced by one of the characters, see Table 6.1.

Table 6.1. - Modes work With files

mod	Mode	Note
'r'	reading a file (read)	the file must exist if the file does not exist, then an exception is raised : FileNotFoundError
'w'	writing to a file (write)	If file does not exist, That He is being created
'a'	adding to file (append)	If the file does not exist, it is created, recording is being carried out V end file
'r+'	reading and recording	the file must exist if the file does not exist, then an exception is raised : FileNotFoundError
'w+'	reading and recording	If the file does not exist, it is created, existing file is being rewritten
'a+'	reading and recording	If the file does not exist, it is created, recording is being carried out V end file
'x'	create file For records	if the file exists, then it is executed exception: FileExistsError
'x+'	create file For readings and records	if the file exists, then it is executed exception: FileExistsError

The mode can be followed by a modifier that determines the file opening mode: `t` - text or `b` - binary.

For solutions ours tasks us necessary open two file. One the file will contain information for calculating expressions and will be opened for reading, and the second one – for displaying calculation results – will be opened for writing:

```
fi = open("lab1_pb_in.txt", mode = "rt")
fo = open("lab1_pb_ou.txt", mode = "wt")
```

Read from file:

Reading a file Can organize in different ways, for example, to read line by line (`readline()` method) or read the entire file into a buffer (`readlines()` method) and then process the lines. Note that the lines are terminated by the end character lines (`'\n'`). The `readline()` method reads a line, including the end-of-line character.

When reading line by line, iteration (reading the next line) can be performed using a `for` loop . Let's look at two examples:

```
1) while True:
    line = fi.readline() # reading if
    lines not line:      # line empty
        break           # end file And processing
    elif line=="\n":     # end of line
```

```
        continue          # let's continue reading  
lines (b, c) = line.split() # split line
```

```

2)      for line in fi:          # For everyone lines
          file if line=="\n":    # end of line
              continue          # continue reading lines
          (b, c) = line.split("\t") # divide line

```

In the first example, the iterations are performed in a while loop when the line reading instruction is executed. The read value is stored in the `line` variable, and a check is made to ensure that the value received is not empty. If instructions `fi.readline()` returns an empty string, which means that the end of the file (EOF) has been read and further processing can be stopped (`break`). In addition, it checks whether the string contains information. If the string does not contain information, it will contain only the end-of-line character. In this case, processing should continue by reading the next line (`continue`).

Note: If a string contains no information (no characters that can be visualized), then it contains a line break character. If a string is empty, then it contains NO characters.

In the second In this example, iterations (reading lines) are performed using a for loop. Lines are read one by one into the `line` variable. In this case, there is no need to check for end-of-file (EOF).

Further processing of the read information is performed in accordance with the data recording format.

In our example, we placed two lines at the beginning of the file describing the data that follows, and then arranged the data itself into rows in two columns. The column separator can be a tab character or multiple spaces.

At reading such file let's do it next way:

- Let's read two lines without processing (skip these lines). These lines are a reminder that can be used when preparing the file in a text editor;
- V cycle we read line, And split her For receipt data. The format for recording the splitting (division) method is as follows:

```
str.split(sep=None, maxsplit=- 1) ,
```

Where `str` - line symbols, `sep` - separator, A `maxsplit` - the number of groups into which the string is divided.

If the separator `sep` is specified (`sep` stands for separate) and the number of groups `maxsplit`, then the string will be split into `maxsplit + 1` parts. If `maxsplit` If not specified or equal to -1, the number of parts into which the string will be divided is unlimited. Example of splitting a string:

```
'1,2,3'.split(',',maxsplit=1) # ['1', '2,3']
```

We can do not specify parameters in the `split()` method , because When splitting, a list of two values will be formed (there are two columns in a row), and we have chosen spaces or a tab character as the separator.

IN left parts instructions We we will indicate two variables, which will take the values obtained by splitting the string.

```
b, c = line.split()
```

The values obtained during splitting will be of string type and for further use it is necessary to convert them to real type (`float`).

Record to file:

To write to a file we will use the `write(<Data>)` method . Here, the data is a string, including a format string containing formatting characters. When writing a string to a file, the `write()` method Not Adds a line break character. To ensure that subsequent data is written on a new line, the current line must end with a line break character (`'\n'`). An example of this can be found in the program listing below.

Below are two versions of the program and their output. The second version uses a NumPy module function to load data from a file, and the data is loaded into a two-dimensional array (the row highlighted in bold).

Listing programs

First option:

```
# -*- coding: cp1251 -*-
from math import *

def f1(a,x):
    y = (tan(x**2/2-1)**2+(2*cos(x- p/6))
        /(1/2+sin(a)**2))
    return y

do f2(x):
    y = pow(2, log(3- cos(pi/4+2*x),3+sin(x))
        /(1+tan(2*x/p)**2))
    return y

fi = open("lab1_pb_in.txt", "rt") # read fo file
fo = open("lab1_pb_ou.txt", "wt") # write V file
line = fi.readline()             # Skip lines line =
fi.readline()                    # header V file #
Output hat tables V file
fo.write("+=====+\n")
fo.write("I    A I    X I    F1    I    F2    I\n")
fo.write("+=====+\n")
```

```

for line in fi:      # For everyone lines
    file if line=="\n":
        continue
    (b, c) = line.split()    # split
    a = float(b)            # bring To substance type
    x = float(c)
# Conclusion V file
fo.write("I {0: .1f} I {1: .1f} I {2: 5.4f} I"
        .format(a, x, f1(a, x)))
fo.write("{0: 6.4f} I\n".format(f2(x)))
fo.write("+-----+-----+-----+-----+
        "-----+\n")
fi.close()    # закроем файлы
fo.close()

```

Второй вариант:

```

# -*- coding: cp1251 -*-
from math import *
import numpy as np
def f1(a,x):
    y = (tan(x**2/2-1)**2+(2*cos(x- p/6))
        /(1/2+sin(a)**2))
    return y
def f2(x):
    y = pow(2, log(3- cos(pi/4+2*x),3+sin(x))
        /(1+tan(2*x/p)**2))
    return y
fi = open(r"lab1_pb_in.txt", "rt") # read fo file
fo = open(r"lab1_pb_ou.txt", "wt") # write V file #
Output hat tables V file
fo.write("+=====+=====+=====+=====+\n")
fo.write("I    A    I    X    I    F1    I    F2    I\n")
fo.write("+=====+=====+=====+=====+\n")
# Let's load it data V array
ax = np.loadtxt(fi,dtype=np.float, ndmin = 1)
nRow, nCol = ax.shape # Lines And columns V array
for Row in range(nRow):    # for everyone lines
    a = ax[Row][0]
    x = ax[Row][1]
    fo.write("I {0: .2f} I {1: .2f} I {2: 6.4f} I"
            .format(a, x, f1(a, x)))
    fo.write("{0: 6.4f} I\n".format(f2(x)))
    fo.write("+-----+-----+-----+-----+
            "-----+\n")

```

```

fi.close()    # закроем файлы
fo.close()

```

Note : Please note how the long lines are wrapped. In the function `f1()` The entire expression is enclosed in parentheses. In the `f2()` function parentheses are used in the `pow()` function , and in the `fo.write()` method for writing data to a file –method parentheses.

Result work programs (text file lab1_pb_ou.txt)

```

+=====+=====+=====+=====+
I      A      I      X      I      F1      I      F2      I
+=====+=====+=====+=====+
I -2.00 I -2.00 I   1.1970 I 1.1184 I
+-----+-----+-----+-----+
I  0.00 I -2.00 I -0.8347 I 1.1184 I
+-----+-----+-----+-----+
I  0.00 I  0.00 I  5.8896 I 1.6880 I
+-----+-----+-----+-----+
I  2.00 I  0.00 I  3.7309 I 1.6880 I
+-----+-----+-----+-----+
I  1.50 I  0.50 I  2.7712 I 1.7955 I
+-----+-----+-----+-----+
I  4.00 I  3.00 I -1.3266 I 1.0517 I
+-----+-----+-----+-----+

```

Addition To practical work No. 5

We'll do this in two steps. In the first step, we'll create an array using the `random()` function. and save it in text format. We'll create the array, write it to a file, and read it from a file using NumPy module functions.

To initialize the array, calculate the mean and variance, and adjust the array, we will use the previously written functions.

We will make some small changes to the array generation function so that we can obtain not only square, But and a rectangular array.

Listing programs

```

import numpy as np
from math import *

def MakeMatr(n, m, a = -5, b = 5):
    """ Initialization of a rectangular
    matrix of size nxm pseudo-random values
    in the range [a, b) """
    Matr = a+(b-a)*np.random.random(size = (n,m))
    return Matr

```

```

def MidlDisp(Matr):
    """ Calculation average values
    Calculating the variance """
    sum = 0
    (nRow, nCol) = Matr.shape # Dimensions matrices
    for Row in range(nRow):
        for Col in range(nCol):
            sum += Matr[Row][Col]
    Midl = sum / Matr.size # Среднее
    Disp = 0
    for Row in range(nRow):
        for Col in range(nCol):
            Disp += (Matr[Row][Col] - Midl)**2
    return (Midl, Disp / (Matr.size - 1))

def MakeCorrect(Matr, avr, sigma):
    """ Replacing " bounced " values """ (nRow,
    nCol) = Matr.shape # Dimensions matrices
    for Row in range(nRow):
        for Col in range(nCol):
            if abs(abs(Matr[Row][Col]) - avr) > sigma:
                Matr[Row][Col] = avr
    return(Matr)

n,m=input(" Enter size matrices (N M):").split() n
= int(n)
m = int(m)
tstMatr=MakeMatr(n, m, -7, 7) # we're cooking
matrix str_o = 'Resalt after initiation:'
fh_i = open("lab5_pd_in.txt", "wb") # For entries
np.savetxt(fh_i, tstMatr, fmt = ' %8.4f',
            header = str_o) # we'll unload her V
file fh_i.close() # close file

# We are doing it exercise
# Let's load it data V array
fh_i = open("lab5_pd_in.txt", "rt")
MyMatr = np.loadtxt(fh_i, dtype=np.float, ndmin=1)
# Open file For results work programs fh_o =
open(r"lab5_pd_ou.txt", "wb")
# Save the original str_o
array = 'Before correction:'
np.savetxt(fh_o, MyMatr, fmt = ' %8.4f',
            header = str_o)

# We calculate average And
dispersion mMidl, mDisp =
MidlDisp(MyMatr)

```

```
# let's prepare lines For records
```

```

str_o = '\nAfter correction:'
fstr=" ".join((' \nMidl= {0:5.4f} Disp= {1:6.4f}',
               'Sigm= {2:6.4f}'))
str_o1 = fstr.format(mMidl, mDisp, sqrt(mDisp))
# We are making adjustments array
NMatr = MakeCorrect(MyMatr, mMidl, sqrt(mDisp))
# Save result work V file V format: # Header
# Adjusted array # final
part
np.savetxt(fh_o, NMatr,fmt = ' %8.4f',
           header = str_o, footer = str_o1)
fh_o.close() # let's close the
file

```

Note : In this example, the `.join()` method The format string `fstr` is compiled and used to save the calculation results. in the file, see the following instructions. The space in double quotes before the method is a separator used when concatenating strings. You can also use a combination of characters instead of a space.

Result work programs (text file lab1_pd_ou.txt)

```

# Before correction:
3.3232    -1.7425    -5.2076
6.2726    -0.8202     6.1146
4.6182     0.2746    -2.9510
2.6626    -3.9889    -5.1610
#
# After correction:
3.3232    -1.7425     0.2829
0.2829    -0.8202     0.2829
0.2829     0.2746    -2.9510
2.6626    -3.9889     0.2829
#
# Midl= 0.2829 Disp= 17.9449    Sigm= 4.2361

```

Important : When viewing the file in Notepad, the entries will appear as a single line. Open the files in WordPad. In this case, you will see a different representation of the data. If you can find the reason for this ambiguous representation, contact your teacher for clarification.

Exercise To practical work No. 6 Files

Make adjustments to the programs written for laboratory assignments No. 1, No. 4 and No. 5, so that data input and output of work results are carried out using files.

Practical Job No. 7: Graphics Programming. Turtle Module

Objective of the work:

Introduce student on practice With writing programs For generating graphical images using Python modules.

Production tasks

Using program code from practical work No. 2 For Tasks

1 And Tasks 2, A So same one from graphic modules Python, write scripts that illustrate the code's operation graphically:

- Task 1 – Solve the inverse problem – plot the graph of the function using the program code written for this task;
- Task 2: Construct a graphical representation of the result of generating points with random coordinates. Use up to 10,000 trials. Estimate the area of the shaded portion of the figure using the Monte Carlo method and compare it with the actual area. Express the estimate as a percentage of the actual area.

Theoretical introduction

Python has several modules for working with graphics. Two graphics modules are part of the standard Python libraries:

- `turtle` (`turtle`) is a simple graphics package that can also be used to create a simple user graphical interface – Graphical User Interface (GUI);
- `tkinter` – designed specifically for creating graphical user interfaces (GUIs). The IDLE interface is built using `tkinter` . This module includes a Canvas widget for drawing graphical images.

Another way to write graphical applications is to use cross-platform graphical libraries. One such cross-platform graphical library is Qt. This library is used with languages such as C++, Java, Ruby, Delphi, Lazarus, and others. It also has a Python binding, PyQt.

Computer graphics is a vast and complex field of study. It encompasses the technical means of displaying images, including raster and vector displays, plotters, and printers. It also includes knowledge of the methods used to create color images, which are perceived either as the glow of individual display dots or as reflections. For example, from sheet papers: additive color RGB model or subtractive - CMY. A vast field of mathematical and physical knowledge helps decide questions movements, turn object,

formation of the second and first plans of the image (scene), taking into account reflexes (secondary reflections) and much more.

This tutorial examines the functionality of graphical modules that allow you to draw function graphs.

At construction graphs you will be required knowledge of functions and methods of the graphical system that allow:

- create a window in which the graph will be built (size, coordinate system);
- build graphic primitives (dot, line, ...);
- set the width and color of lines, background color, fill the image or part of it with a specified color;
- control the marker (pointer) used for drawing;
- apply text.

Module turtle

`turtle` module is part of the standard Python library. This module was originally marketed as a tool for teaching computer graphics to children.

The model of this module is as follows. There is a rectangular surface, by which crawls turtle. The turtle can move a set distance forward, backward, at an angle, or to specified coordinates. A turtle can be cloned, creating a group of turtles. Each turtle lives its own life. To draw, the turtle uses a colored pen (pencil), which can be raised or lowered. If the pen is lowered, it leaves a mark. The color and thickness of the line can be changed.

The turtle understands commands that can be used to draw a circle of a given radius and color, an arc with a given angle, and fill. The shape can be colored, retrieved, or modified. The turtle's shape can be customized and used as a stamp, leaving a drawing on the canvas.

The `turtle` module also implements interactive methods for interacting with the turtle. Events related to mouse clicks or keyboard key presses and releases can be handled by custom functions bound to these events.

ATTENTION :

1. The Python language and its modules are constantly evolving and being modified. When using modules, please monitor their updates and the availability of updated documentation.

2. WITH electronic version benefits is spreading file `Turtle_3-6-1.pdf`, in which the list of functions and methods of the `turtle` module is provided. in Russian.

our acquaintance with the functions of the turtle module by solving a specific problem: we will plot the graph of the function that was specified in graphical form in practical work No. 2 Task 1 (we will solve the inverse problem).

Solution first tasks

In the first step, we will format the program written in Practical Work #2, Task 1 as a function, and then add the graphical part.

When plotting a function graph, we iterate over the values of the function argument within the range specified by the graph. Since the process of obtaining the argument is programmatic, situations are possible, in general, where the function value cannot be calculated for a given argument, and the program terminates with an error. An error can be caused, for example, by dividing by zero or taking the square root of a negative number.

To avoid such situations, it is necessary to investigate the function and the points at which it cannot be calculated should be excluded using conditional operators. We will assume that the function returns the value None for such points.

Our function doesn't have any keypoints, but the function listing below includes some additional code. This code is provided as an example and is commented out, but you can test it later.

```
def MyFun(x):
#     if (x >= 5) and (x <= 7):
#         return(None)
    if x < -5:
        y = 1
    elif x >=-5 and x<0:
        y = -(3/5)*x-2
    elif x >= 0 and x<2:
        y = -sqrt(4-x**2)
    elif x >= 2 and x<4:
        y = x-2
    elif x >= 4 and x<8:
        y = 2+sqrt(4-(x-6)**2)
    else: y = 2
    return(y)
```

Comment: Parameters, necessary for solutions tasks follows obtain from the graph and determine in the program, for example, the radius of the arc.

Graphic Part

At construction graphics we necessary execute row conditions and conduct preparatory work. Before, how go to To writing

To understand the graphical part of the program, you need to understand the graphical system. It is necessary to answer the question: "How does it work?"

Note : Monitor here refers to an external device that connects To video card systemic block. Display – A device used to form an image. A display is a component of a monitor. The image is composed of square elements called display pixels , and each pixel consists of three rectangular elements that allow for the formation of the pixel's color in RGB format. One of the technical characteristics of a display is its resolution: the number of dots horizontally and vertically. For example, 1680x1050.

A graphic image can be created on a monitor display, printer, or plotter. Each of these devices has specific functional capabilities. In order to create an invariant graphic systems (device-independent graphics) use certain positions. Let's consider some of them for a flat drawing.

1. In a multitasking system, each application can have its own graphical window, which can occupy the entire display surface or part of it. This window is called the main window. The window's dimensions and position are specified as arguments to the `setup()` function :

```
turtle.setup(Dw, Dh, Xc, Yc),
```

Where `Dw` And `Dh` dimensions windows horizontally and vertically, respectively, and `Xc` and `Yc` are the coordinates of the window on the display. If the coordinate values are positive, they determine the position of the upper-left corner. The main window, and if negative, the lower-right corner. The origin of the coordinate system will be at the center of the window. The unit of measurement is pixels. The window can be resized by dragging its borders. The direction of the coordinate system axes depends on the mode; see the `mode()` function. For more information, see the `turtle` module documentation . Please also note the configuration file, which can define parameters for the main window and other features.

2. For drawing, the Python graphics system provides the user with a canvas. The canvas's dimensions can be set using the `screensize()` function :

```
turtle.screensize(Cw, Ch, bg) ,
```

Where `Cw` , `Ch` - dimensions canvas By horizontals And verticals, A `bg` - color background.

Note: If you use the `screensize()` function to change only the background color;

```
turtle.screensize(bg="yellow") ,
```

That dimensions canvas will be installed How dimensions by default.

Another way to determine the canvas dimensions, and at the same time the position of the world coordinate system, is to use the `setworldcoordinates()` function :

```
turtle.setworldcoordinates(Xlc, Ylc, Xrc, Yrc) ,
```

where `Xrc`, `Yrc` – the coordinates of the lower-left and upper-right corners of the canvas, respectively. The graphics system is set to "world" mode. The `bgcolor()` function can be used to set the background color . If the canvas dimensions are larger than the main window, scrollbars—canvas widgets—appear.

Both the main window and canvas are initialized when the `turtle` module is imported . Initialization values are taken from the configuration file, and if one is missing, default values are used. You can also use the module's functionality to change the size and position of the main window, as well as canvas parameters, as desired.

While the main window dimensions and coordinates are specified in pixels, the canvas dimensions are user-defined units. For example, suppose you're drawing geometric shapes on a sheet of paper and specifying the dimensions in millimeters. If your sheet is A4, the canvas dimensions could be defined as 210x297, and the coordinates of a point on the canvas could be specified as (34.7; 12.0). If the unit of measurement is centimeters, the canvas dimensions could be specified as 21x29.7, and the coordinates of the same point would be (3.47; 1.2). And Finally, if you are drawing the solar system, you might as well define the dimensions of the canvas in astronomical units and indicate the positions of the bodies being depicted in the same units and their fractions.

What are the relationships between the concepts introduced: main window, canvas, and world coordinate system? Figure 7.1 explains this.

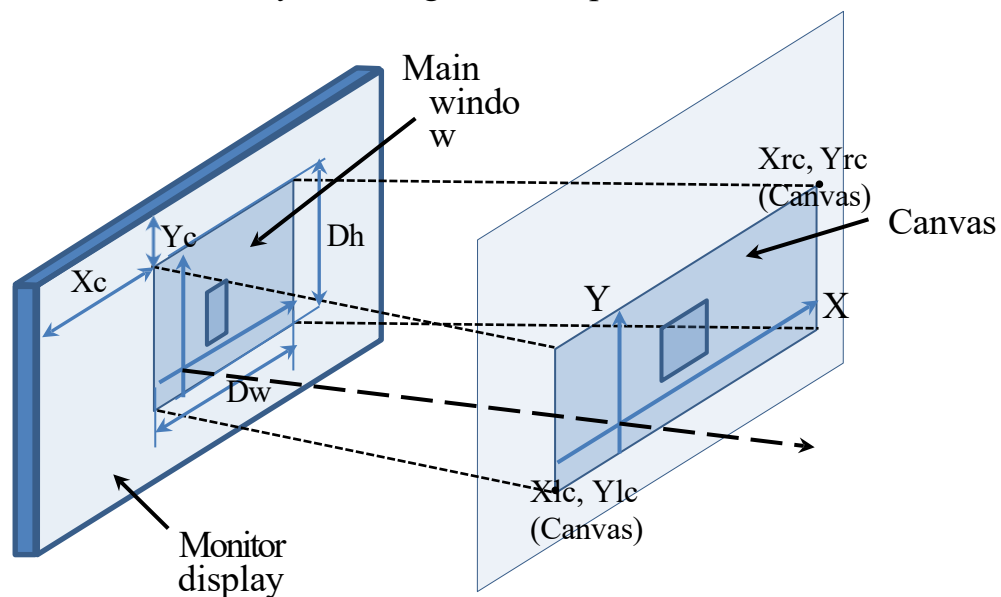


Fig.7.1. - Display, The main thing window, canvas And world coordinate system

The canvas is placed in the projection plane ($z=0$). Coordinates X_{lc} And Y_{lc} determine position left lower angle canvas, X_{rc} , Y_{rc} - right. Canvas dimensions make up: $Cw = X_{rc} - X_{lc}$ And $Ch = Y_{rc} - Y_{lc}$. The graphics system projects the canvas onto the main window so that the corner points of the main window and the canvas coincide, and a correspondence is established between units of measurement in the world coordinate system and pixels. main window - scale. Scales set along the X axes and Y may be different.

The following point is clear from the figure: if the ratio of the canvas width to its height and the ratio of the main window width to its height (the aspect ratio) are not the same, distortion occurs. The angles and relationships between image elements are distorted. As shown in Figure 7.1, a square is transformed into a rectangle because the canvas width is greater than the main window width, even though their heights are equal. This effect can be used, for example, to resize an image or intentionally distort it. Incidentally, a similar effect can be observed with graphs in Excel.

Note: In the figure, the coordinates of the lower left corner of the canvas are negative, and the origin of the coordinate system is on the canvas. If the coordinates of the corner are defined as positive numbers, the origin will be outside the canvas. Shapes placed on the canvas may extend beyond the canvas dimensions and, therefore, appear outside the main window. To detect such "escaped" shapes, you can resize them by dragging their borders with the mouse.

Before plotting a graph, you need to decide on the ranges over which the function's argument and value will vary. We'll use the following notations: X_{min} , X_{max} . – the range of variation of the argument, and Y_{min} , Y_{max} – the range of function values. Then the canvas parameters can be set as follows:

```
turtle.setworldcoordinates(Xmin, Ymin, Xmax, Ymax)
```

It's also worth keeping in mind that the window border overlaps part of the image, so the canvas parameters and corresponding main window parameters may need to be set with some leeway. This is a minor issue and can be resolved through debugging.

If we assume that the width and height of the main window into which the canvas will be projected are Dw and Dh , respectively, then to avoid distortion of the corners and elements of the graph, the following condition must be satisfied:

$$k = Dw / Dh = (X_{max} - X_{min}) / (Y_{max} - Y_{min})$$

$$k = \frac{Dw}{Dh} = \frac{Xmax - Xmin}{Ymax - Ymin}$$

or, for example, $Dh = Dw / k$ $Dh = \frac{Dw}{k}$

Now Can go to To decision our tasks.

First step :

Let's define the boundaries of the area in which we will draw our graph: boundaries changes argument (x) And borders changes values functions (y). In general, when we don't know the function's behavior over a given interval of its argument, we can pre-calculate the function and find its minimum and maximum values. These values will help define the boundaries of the plotting domain. Furthermore, such calculations can be performed once, placing the results in either an array or a list, which can then be used to plot the graph.

Let's return to Practical Work #2, Task 1, and accept the notations for the variables that we will use in our program.

a) Importing modules. We'll need math functions and functions from the turtle module :

```
from math import sqrt
import turtle as tr
```

b) Dimensions By horizontals And verticals make up (-10, 10) And (-2, 4) conventional units, respectively. Let's increase these dimensions by 20% (10% on each side) so that the graph fits within the main window (the percentage choice is subjective). We get:

```
aX = [-12, 12] # left and right
aY = [-3, 5]   # lower And top
```

c) Let's define the horizontal size of the main window (Dx) as 800 pixels. Then, with the given graph parameters, the height of the main window, to avoid distortion, will be:

```
Dx = 800
Dy = Dx / ((aX[1] - aX[0]) / (aY[1] - aY[0]))
# we create The main thing
window tr.setup(Dx, Dy)
```

G) Let's install it number points For receipt high-quality drawing:

```
# number points drawing
Nmax = 1000
```

d) Set the world coordinate system

```
tr.setworldcoordinates(aX[0], aY[0], aX[1], aY[1])
```

The second step

turtle module functions such as `up()` , `down()` , `goto()` , `write()` , `begin_fill()` , and `end_fill()` .

Drawing axes, tags, shooter And inscriptions To axes, designed V view functions. Algorithm drawing axes simple:

- raise feather;
- go to To left (lower) border;
- lower feather;
- go to To right (upper) border.

Algorithm application divisions And inscriptions:

- raise feather;
- V cycle from left (lower) borders To right (upper) border, with a given step;
- go to To point on axes;
- lower feather;
- move to a point below (to the right) of the axis. Distance from the axis (length of the dash) is selected empirically;
- raise feather;
- go to To point application inscriptions (below / to the right axes): we determine empirically;
- we are deducing number, preliminary having transformed his V line;
- If cycle Not ended, That go to To next iterations.

Algorithm drawing arrows:

- we are preparing two list with coordinates characteristic points drawing a And b . The first one point With coordinates (0, 0) Not we indicate, see Fig.8.2;
- V cycle we are building polygon And we register his How new turtle shape ;
- we appoint the turtle a new shape - our polygon and we pull out the arrow;
- we are installing turtle V That place, Where should be arrow, rotate it at the desired angle and create a stamp:
- we inscribe axis.

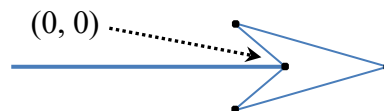


Fig.7.2.– Characteristic points arrows, used V program

Third step

Let's move on To drawing functions. On this step let's choose another color and line thickness. By the size of the argument's range of values and by the number of points `Nmax` we determine the increment along the X axis :

$$dx = (aX[1] - aX[0]) / Nmax .$$

Algorithm drawing next:

- get initial coordinate, calculate meaning functions;
- install turtle V initial position;
- V cycle while , from initial values argument to final values;
- get current meaning argument, calculate meaning functions;
- If function Not calculated (None), feather raise, A otherwise go to to the point and lower the pen;
- continue the loop until a value greater than the value for the right-hand side of the argument is obtained.

Note : It should be noted attention for that, what's in body the loop checks for the validity of the values for Y . Since we have assumed that for singular points the function returns the value None , then when detecting In such a situation, the drawing process is skipped (the pen is raised).

That's all. Below is the program listing and the output, see Fig. 7.3.
P.S.: Delete symbols comments V functions And take a look on result.

Listing programs

```
# -*- coding: cp1251 -*-
from math import sqrt
import turtle as tr
#
def Fun1(x):
    """
    Curve from lab. 2 Exercise
    1 """
    #if (x >= 5) and (x <= 7):
    #    return(None)
    if x < -5:
        y = 1
    elif x >= -5 and x < 0:
        y = -(3/5) * x - 2
    elif x >= 0 and x < 2:
        y = -sqrt(4 - x**2)
    elif x >= 2 and x < 4:
        y = x - 2
    elif x >= 4 and x < 8:
        y = 2 + sqrt(4 - (x - 6)**2)
    else: y = 2
    return(y)

def Axis(txy, ax = 'X'):
    """
```

```

Drawing axes.
txy - list: [Xmin, Xmax]
         or   [Ymin, Ymax]
ax - 'X' or 'Y'
"""
fluent = txy[0]
b = txy[1]
tr.up()
if (sure ==
    'X'): pb =
    [fluent, 0]
    pe = [b, 0]
else:
    pb = [0, a]
    pe = [0, b]
tr.goto(pb)
tr.down()
tr.goto(pe)

def Mark(txy, ax = 'X'):
    """
    Marking axes.
    txy - list: [Xmin, Xmax]
            or   [Ymin, Ymax]
    ax - 'X' or 'Y'
    """
    fluent = txy[0]
    b = txy[1]
    tr.up()
    for t in range(a, b):
        if (ax == 'X'):
            pb = [t, 0]
            pe = [t, 0.2]
            pw = [t, -0.5]
        else:
            pb = [0, t]
            pe = [0.2, t]
            pw = [0.2, t]
        tr.goto(pb)
        tr.down()
        tr.goto(pe)
        tr.up()
        tr.goto(pw)
        tr.write(str(t))

```

```
def Arrow(txy, ax = 'X'):
```

```

"""
Drawing arrows.
txy - list: [Xmin, [Xmax]
           or [Ymin, Ymax]
ax - 'X' or 'Y'
"""
# Parameters polygon
a = [0.1, 0, - 0.1]
b = [-0.1, 0.3, - 0.1]
tr.up()
tr.goto(0, 0)          # to the beginning
tr.begin_poly()        # let's begin recording
vertices for i in range(2): # for all
vertices
    tr.goto(a[i], b[i]) # polygon
tr.end_poly()          # we're stopping entry
p = tr.get_poly()      # link on polygon
# register a new turtle shape
tr.register_shape("myArrow",p)
tr.resizemode("myArrow")
tr.shapesize(1,2,1) # stretch (example) if
(ax == 'X'):          # for the X-axis
    tr.tiltangle(0)    # angle for the
    shape tr.goto(txy[1]+0.2,0) # To place
    arrows pw = [int(txy[1]),-1.0] #
    inscription
else:                  # For axes Y
    tr.tiltangle(90)   # angle for the
    shape tr.goto(0,txy[1]+0.2) # To place
    arrows pw = [0.2, int(txy[1])] #
    inscription
tr.stamp()            # leave stamp - arrow #
label axis
tr.goto(pw)            # To place
inscriptions tr.write(ax,
font=("Arial",14,"bold"))
#
def main():
# Initial parameters
# *****
# Borders graphics: [Xmin, [Xmax] And [Ymin,
Ymax] aX = [-12, 12] # left and right
aY = [-3, 5]        # lower And top #
Main window
Dx = 800

```

```
Dy = Dx / ((aX[1] - aX[0]) / (aY[1] - aY[0]))
tr.setup(Dx, Dy)
tr.reset()
# Number points drawing
```

```

    Nmax = 1000
#
# Setting the world coordinate system
    tr.setworldcoordinates(aX[0],aY[0],
                           aX[1],aY[1])
#
    tr.title("Lab_8_2_1")      # title
    tr.width(2)                # line thickness
    tr.color("blue", "blue")  # color And
    filling
#
    *****
    ***** tr.ht()           # invisible
    tr.tracer(0,0)            # no delays
#
# X - axis, tags, arrow
    Axis(aX, 'X')
    Mark(aX, 'X')
    Arrow(aX, 'X')
# Y - axis, tags, arrow
    Axis(aY, 'Y')
    Mark(aY, 'Y')
    Arrow(aY, 'Y')
#
# Function
    tr.color("green")          # line color
    tr.width(3)                # And
    thickness dx = (aX[1]-aX[0])/Nmax #
    step
# V start
    x = aX[0]
    y = Fun1(x)
    if (y is None):
        tr.up()
        tr.goto(x, 0)
    else:
        tr.goto(x, y)
        tr.down()
# пусчем
    while x <= aX[1]:
        x = x + dx
        y = Fun1(x)
        if (y is None):
            tr.up()
            continue

```

```
else:  
    tr.goto(x, y)  
    tr.down()
```

```
#
if __name__ == "__main__":
    main()

#
# Comment at work V IDLE #
tr.mainloop()
```

Result work programs

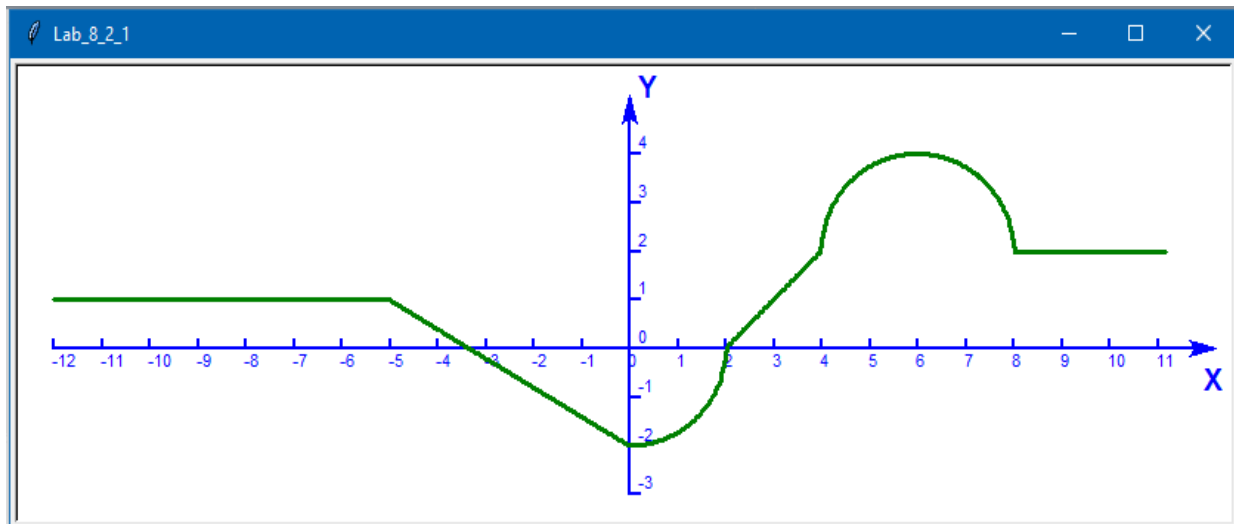


Fig.7.3.– Schedule functions

Second task

To solve the second problem (Practical Work No. 2, Task 2), we will need to understand what the Monte Carlo method and a random number generator are.

Let us remember that the task required writing a program that determines Based on the user-entered coordinates of a point, it is determined whether this point falls within the shaded area.

Theoretical introduction

The Monte Carlo method is a numerical method for solving mathematical problems tasks at help modeling random quantities (method of statistical testing).

One of the simplest mechanisms for generating random variables is roulette, a staple of casinos. The European city famous for its casinos is Monte Carlo, the capital of the Principality of Monaco. The method's name derives from this city.

One of the problems solved by the Monte Carlo method is calculating the area or volume of a complex figure. The essence of the method is as follows. If a figure is depicted on a plane, then another figure can be described around it, whose area we can calculate precisely. For example, a circle or a regular polygon. By generating N random points, the coordinates of which will be uniformly distributed over the surface described figures, We Can count number points, which got there V

figure With unknown area – N_f . Knowing square describing figure S , you can calculate the area of the desired figure with a certain accuracy:

$$S_f = S \cdot \frac{N_f}{N}$$

The accuracy will be higher the more points are generated and the more evenly they are scattered throughout the entire circumscribing figure. Accuracy

proportional $\sqrt{\frac{D}{N}}$, Where D - some constant, N - number tests (number points).

Please note that to increase accuracy by an order of magnitude (10 times), the number of tests will need to be increased by 100 times. The method's application became possible with the development of computing technology.

More information O method Monte Carlo Can find V Internet.

Solution second tasks

For our task we need a generator that generates random numbers with a probability uniformly distributed over a given interval For this goals we use function `uniform()` module `random`:

```
from random import uniform
```

Using the listing of the program written in Practical Work #2, Task 2 and the knowledge gained from solving the previous problem, it is easy for us to write a program in which the coordinates of the points will be generated randomly, and the result of the hit will be displayed graphically.

Definition accurate area figures

In the problems described in Practical Work 2 Task 2, the area of most shaded figures is found simply.

Since our figure is formed by the intersection of a cubic parabolas and straight lines, the exact determination of area is possible using integral calculus.

IN in general case square figures, educated two lines, Can

find from expressions: $S_f = \int_a^b (f_2(x) - f_1(x)) \cdot dx$

Where $f_2(x)$ and $f_1(x)$ are the functions that bound the figure, a and b – the boundaries of the figure on the left and right, respectively. In this case, $f_2(x) \geq f_1(x)$. The equations for our lines are as follows:

$$f_1(x) = 2x + 2 \quad \text{And} \quad f_2(x) = x^3 - 4x^2 + x + 6$$

Let's split our figure into two regions. One region will be in the argument range $[-1, 1]$, and the second – $[1, 4]$. Then we get that the area is equal to the sum of two integrals:

$$S_f = \int_{-1}^1 (f_2(x) - f_1(x)) \cdot dx + \int_1^4 (f_1(x) - f_2(x)) \cdot dx$$

We won't provide a detailed solution. The calculation yields: $S_f = 253/12 \approx$

$$S_f = \frac{253}{12} \approx 21.0833$$

21.0833 For calculations area figures method Monte Carlo let's describe

near

it is a rectangle with a base of 7 and a height of 13 conventional units (range a $x = [-2, 5]$ and a $y = [-2, 11]$). The area of the rectangle is:

$$S = [5 - (-2)] * [11 - (-2)] = 91.$$

IN program No any features. It follows turn Please note that drawing a large number of dots takes a lot of time.

To speed up calculations, you can avoid drawing points that do not fall within the shaded area, i.e., leave only the first part of the conditional operator, which is located in the body of the loop for generating point coordinates.

Listing programs

```
# -*- coding: cp1251 -*-
import turtle as tr
from random import uniform
#
def fun2_2(x,y):
    if (x < -1) or (x > 4):
        flag = 0          #False
    if ((x>=-1) and (x<1) and (y>=2*x+2)
        and (y<=x**3-4*x**2+x+6) or (x>=1)
        and (x<=4) and (y>=x**3-4*x**2+x+6)
        and (y<=2*x+2)):
        flag = 1
    else:
        flag = 0
    return(flag)

# Инициализация turtle
# *****
# Graph boundaries
aX = [-2, 5]          # left and right
aY = [-2, 11] # lower And upper Dx =
300
Dy = Dx/((aX[1] - aX[0])/(aY[1] - aY[0]))
tr.setup(Dx, Dy, 200, 200)
tr.reset()
# number points drawing
```

```
Nmax = 10000  
#
```

```

# Installation world systems coordinates
tr.setworldcoordinates(aX[0], aY[0], aX[1], aY[1])
#
tr.title("Lab_8_2_2")      # title
tr.width(2)                # thickness
lines

tr.ht()                    # invisible
tr.tracer(0,0) # 0 - delay between updates #
*****

#
# drawing tr.up()
functions
mfun = 0                   # points hit V
                           # shaded area for n in
range(Nmax):
# we generate coordinates
    points x =
    uniform(aX[0],aX[1])
    y = uniform(aY[0],aY[1])
    tr.goto(x,y)
    if fun2_2(x,y) != 0:    # got
        hit tr.dot(3,"green")
        mfun += 1
#     else:                  # Not got
hit #                        tr.dot(3,
"#ffccff")
#
tr.color("blue", "blue") # color And
filling #
# Let's draw
axes # X-
axis tr.up()
tr.goto(aX[0], 0)
tr.down()
tr.goto(aX[1], 0)
# Y -axis
tr.up()
tr.goto(0, aY[1])
tr.down()
tr.goto(0, aY[0])
#
# coordinate tags #
And inscriptions on
axes X tr.up()

```

```
for x in range(aX[0], aX[1]):  
    tr.goto(x, 0.1)  
    tr.down()
```

```

        tr.goto(x, 0)
        tr.up()
        tr.sety(-0.4)
        coords = str(x)
        tr.write(coords)

#
# на оси Y
for y in range(aY[0], aY[1]):
    tr.goto(0, y)
    tr.down()
    tr.goto(0.1, y)
    tr.up()
    tr.setx(0.2)
    coords = str(y)
    tr.write(coords)

#
# Let's draw
arrows # on the
X axis
poli = [0, 0.1, 0, -0.1, 0]
Arrbeg = int(aX[1])
Xpoli = [Arrbeg, Arrbeg - 0.1, Arrbeg+0.3,
        Arrbeg - 0.1, Arrbeg]
tr.goto(Xpoli[0], poli[0])
tr.begin_fill()
tr.down()
for i in range(1,5):
    tr.goto(Xpoli[i], poli[i])
tr.end_fill() # we're pouring
arrow # Let's label the X axis
tr.up()
tr.goto(Arrbeg, -0.7)
tr.write("X", font=("Arial",14,"bold"))
#
# на оси Y
Arrbeg = int(aY[1])
Ypoli = [Arrbeg, Arrbeg - 0.1, Arrbeg + 0.3,
        Arrbeg - 0.1, Arrbeg]
tr.up()
tr.goto(poli[0], Ypoli[0])
tr.begin_fill()
tr.down()
for i in range(1,5):
    tr.goto(poli[i], Ypoli[i])
tr.end_fill()

```

Надпишем ось Y

```

tr.up()
tr.goto(0.2, Arrbeg)
tr.write("Y", font=("Arial",14,"bold"))
Sf = (aX[1] - aX[0]) * (aY[1] - aY[0]) * mfun/Nmax
tr.goto(1, 9)
fstr = "N = {0:8d}\nNf = {1:8d}\nSf = {2:8.2f}"
meseg = fstr.format(Nmax, mfun, Sf)
tr.write(meseg, font=("Arial",12,"bold"))
print(meseg)
# Comment at work V IDLE #
tr.done()

```

Result work programs

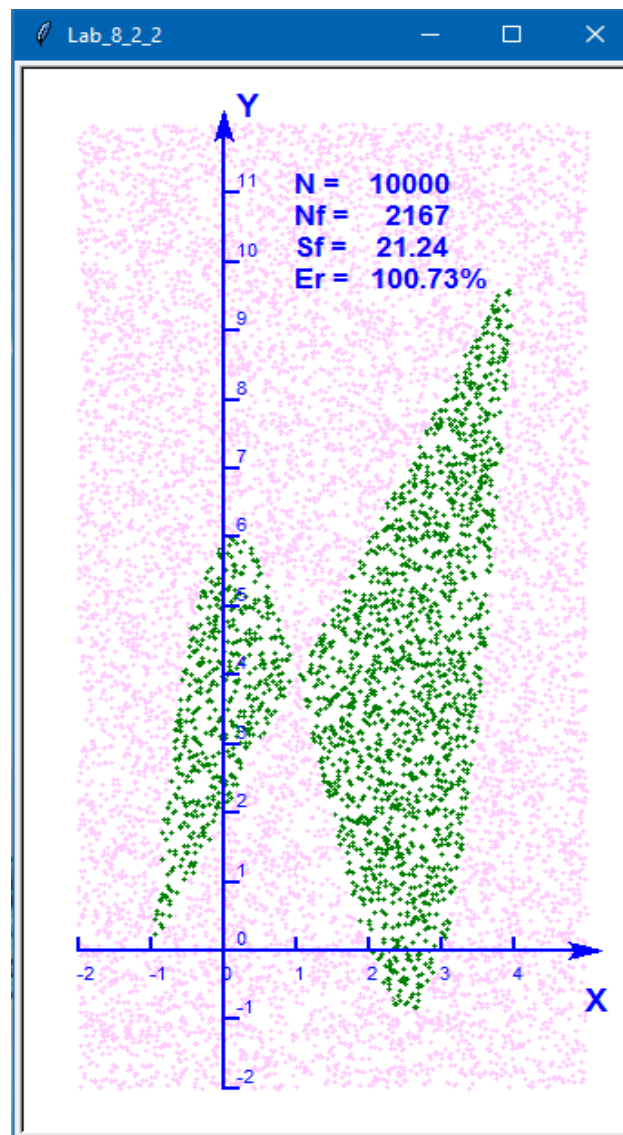


Fig.8.4. - Result program operation

List recommended literature

1. Prokhorenko N.A., V.A. Dronov, Python 3 and PyQt 5. Application Development , BHV-Petersburg, 2017

2. Angel E., Interactive computer graphics. Introductory OpenGL-based course, 2nd edition, Williams Publishing House, 2001
3. Khakhaev I.A., Workshop By algorithmization And programming in Python, Alt Linux, 2011
4. Ermakov CM., Method Monte Carlo V computing Mathematics (introductory course), St. Petersburg, 2009
5. Novozhilov B.V., Method Monte Carlo. M.: Knowledge, 1966.

Assignment for Practical Work No. 7

"Programming" graphics". Module turtle

Complete laboratory work No. 3 of Task 1 in graphic form. execution Tasks 2, 3.

1 decide reverse task - draw
schedule functions.

In Task 2, use the Monte Carlo method to determine the area of the shaded portion of the figure. Obtain a point image of the figure using up to 10,000 trials. Evaluate the accuracy of the area calculation. V percent By relation To real area. Real The area can be obtained using geometric methods and, if necessary, using integral calculus.

In Task 3, draw a graph of a function whose values are calculated through a series with an accuracy of 10^{-3} .

Graph images should occupy most of the screen, be accompanied by a title, contain axis names and gradations, and be scaled based on the source data. For all acceptable source data values, the image should fit entirely on the screen. The program should not rely on specific screen resolutions.

Practical Job No. 8: GUI, module Tkinter»

Objective of the work:

IN this work students get acquainted:

- with the principles of organizing event-driven programming;
- With basics graphical interface organization user (GUI) and widgets of the Tkinter module ^{*)} ;
- with ways constructions graphs functions methods this module.

Note : Students, when solving the problems of this practical work, Not are limited choice Tkinter module . Welcome choice other modules or libraries, such as PyQt.

Production tasks

Display the graphs of two functions on an interval in graphical mode. from x_{nach} to x_{kon} with a step of dx . The first function is defined using a Taylor series and must be calculated with an accuracy of ε . The value of the parameter b , which determines the shift of the function along the ordinate axis, is entered from the keyboard. The graphs should be smooth and have different colors.

$$y(x) = 2 \cdot \left(1 - \frac{2^2 \cdot x^2}{3!} + \frac{2^4 \cdot x^4}{5!} - \dots + (-1)^n \cdot \frac{2^{2 \cdot n} \cdot x^{2 \cdot n}}{(2 \cdot n + 1)!} + \dots \right) \quad -\infty < x < \infty$$

$$z(x) = \frac{\sin(2 \cdot x)}{x} + b$$

Theoretical introduction

To do this, we need to become familiar with Python modules that can be used to plot a function and organize the user interface. In addition, we need *познакомиться* With concept "event" And principles organizing programs that use these events to build GUIs.

Event-driven programming

An event in an operating system is any user action that occurs when interacting with the software interface: pressing keyboard keys, single and double-clicking mouse buttons, and moving the mouse within a window. In addition to these events, the system also includes events triggered by both the operating system itself and other applications, such as the system timer, information exchange between running processes.

^{*)} Throughout the tutorial, the module name is capitalized, like a proper noun. In a script, the module name should be capitalized (e.g., tkinter) for Python 3. In Python 2, the module name is capitalized (e.g., Tkinter).

All events, emerging V computing The system is processed by the operating system. These events are converted into a message—a record containing the necessary information about the event.

For example, this could be a key code and its state transition—whether the key was pressed or released. When a mouse event occurs, the message will contain information about which button was pressed, whether it was a single or double click, and the marker coordinates. mouse at the moment of click.

Messages are received into a common queue, from where they are distributed among application queues.

Each application has its own message processing loop, in which a message is retrieved from the application queue. The operating system then calls a subroutine designed to handle this message.

Figure 8.1 shows the structure of an event-driven program.

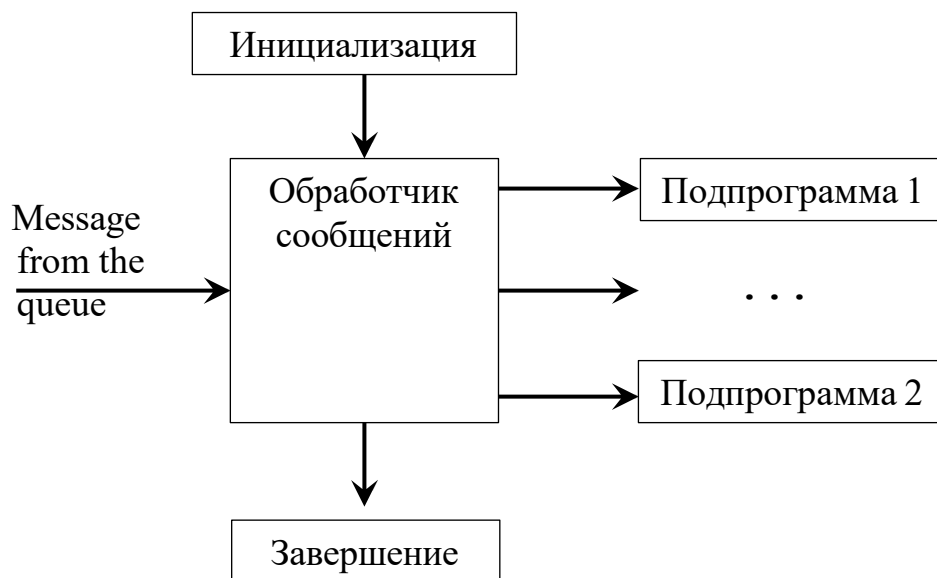


Fig.8.1. - Structure program controlled events

The following conclusion can be made: the program consists of the main part, which performs initialization and termination applications, cycle processing messages, and set message handlers (subroutines).

Module Tkinter

Python has many modules and libraries for creating graphical user interfaces (GUIs). Here's a short list: Tkinter, PyQt5, PyGTK, wxPython, Pythonwin, and others.

The standard module installed with Python by default is `Tkinter`. This module has a clear and understandable syntax and can be used in simple GUI programs.

Next, we will consider the solution to our problem, posed at the beginning of this practical work, within the framework of the capabilities of the Tkinter module.

To use the module's methods and functions, you must import the module. This is accomplished with the following line:

```
from tkinter import *
```

Another form of import is with the assignment of an alias (alias is a calque from English, alias is a nickname):

```
import tkinter as tk
```

Widgets

In Tkinter, the visual components used to build the user graphical interface are called widgets (*widget* , from English *window gadget*).

Виджет – **элемент интерфейса** – a graphical user interface (GUI) primitive that has a standard appearance and performs standard actions. Sometimes you may encounter another name for a control (English: *control*). Here's a short list of widgets we'll use to solve our problem:

- Button – used to launch a command or, for example, to select the next action. For example, the "Draw" button will start the graph drawing process, and the "Exit" button will terminate the script;
- single-line text field (*Entry*) - this widget is designed to display or enter only one, usually small, line, for example, the name of an item or a last name;
- Label – typically used to inform the user. For example, a label could be a description of a single-line text field such as Entry: the purpose of the field;
- canvas, canvas, linen (Canvas) – an area for drawing graphic images;
- Message boxes are dialog boxes that display messages, warnings, or errors when interacting with the user. There are also dialog boxes for working with files, such as file selection boxes when opening or saving a file.

All widgets have two sets of configuration parameters. One set of parameters is common to all widgets: size, position on the form.

The second set of parameters is determined by the widget's individual characteristics: the state of the Spinbox widget's counter, or the index of the value selected by the user in the Combobox widget. Explore these widgets (Spinbox, Combobox) for yourself.

Some widgets, such as Entry and Button, receive focus, allowing them to be linked to the keyboard. Focus can be given to a widget by clicking it or using the Tab key.

Widget-related events (mouse clicks, keyboard presses) can be bound to a method (function) that handles the event. For example, when a user clicks on a canvas, a method can be called that retrieves the mouse cursor coordinates at the time of the click.

Important : Widgets described in the script must be processed by the packager that places them on the form.

There are three method packaging widgets:

Method pack

This method specifies how the widget will fill the form space: where it will be placed, and how it will change when the form is resized.

Method place

This method allows the programmer to control the widget's position by specifying coordinates and dimensions. The position coordinates are relative to the upper-left corner of the window to which the widget is attached. If the window is nested and its coordinates change, the widget's relative coordinates remain unchanged.

Method grid

The most commonly used wrapper. In this method, the window is divided into cells using a grid . The programmer specifies the cell in which the widget will be placed using the column and column number. The number of rows and columns across which the widget will be placed can also be specified.

IN ours script we will use this packer.

Please note that it is not recommended to mix wrappers when creating a single form. More information about widgets and wrapping methods can be found in relevant literature or online .

Interface programs

Let's divide our script into two parts. The head section will contain descriptions of the functions that will perform a certain set of operations, and the second section will contain descriptions of the widgets that will organize the interface for user interaction.

Us will be required next widgets:

- `Canvas` - canvas For graphic performances functions;
- `Entry` - a field for entering data, for example, shifting a graph along the ordinate axis or displaying data about the position of the mouse cursor;
- `Button` - button for setting an action: displaying an updated graph after changing the values specified through the `Entry` field; terminating the program;
- `Label` - label, For descriptions `Entry` fields .

Let's clarify technical exercise To interface script.

The canvas will occupy the main top portion of the form. Input fields, labels describing these fields, and buttons will be placed at the bottom of the form. In the script, we'll describe the following fields:

- fields For conclusion coordinates pointer mice V moment clique;
- Fields for defining the coordinate area boundaries: abscissas and ordinates. Specifying these fields will allow you to view the graph's shape in different coordinate areas;
- fields for specifying the offset of the function and the step that will be used when applying markings on the coordinate axes.
- buttons For conclusion graphics And completion work script.

The window layout is presented as a drawing, see Fig. 8.2

Полотно								
X:		Xmin		Xmax		Шаг меток		Рисовать
Y:		Ymin		Ymax		Смещение		Выход

Fig.8.2. - Layout windows programs

Before we begin describing our layout in the form of a script, let's briefly familiarize ourselves with the parameters of the widgets used.

Canvas - canvas

The Canvas widget methods allow you to create graphic primitives that are used to form a drawing:

- line – create_line(). To draw a line, you must specify the start and end coordinates of the line ends, its thickness, color, type (solid or dotted), and the type of ends (the presence or absence of arrows);
- Closed contour – create_polygon(). To draw a polygonal line, you specify a list of coordinates through which the contour will pass, the parameters described for the line, and the fill color of the shape. You can also specify the method

conducting lines between dots. IN this case coordinates points

will play a weighting role (the line will not pass through the points), but will influence the shape of the resulting figure;

- polygon – `create_rectangle`. This primitive allows you to draw rectangles;
- Oval – `create_oval`. To draw an ellipse or circle, specify the coordinates of the vertices of the rectangle (square) in which it will be inscribed. this are determined coordinates top left angle And lower right, fill color. Just like for a line, you can set the outline width, color, etc.;
- circle, arc – `create_arc()`. This primitive is used to draw arcs, sectors, or segments (defined by the `style` parameter). It is similar to the oval, but has more parameters;
- text – `create_text()`. This primitive Used to display text information in a graphic field representing the canvas. The text can be multi-line. The anchor coordinates are specified as parameters. text fields (anchor), relatively whom text is leveling out In this field, you can set the background and character color, font type, size, ...

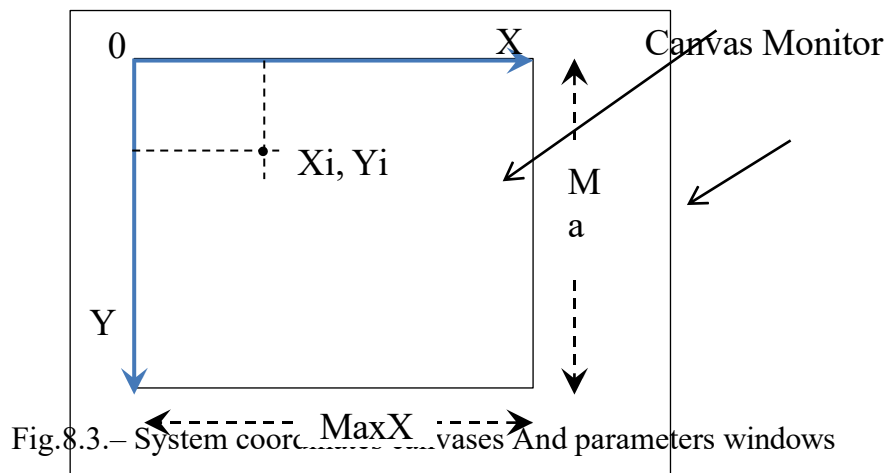
An important feature of the canvas is worth noting. All drawn primitives return an identifier that can be assigned to a variable, and then its value can be used, for example, to remove the primitive's image from the canvas.

System coordinates canvases

Let's look at the coordinate system that is implemented on the canvas .

Note that this coordinate system uses pixels as the unit of measurement. The Tkinter module does not provide a method for creating a custom coordinate system, as is implemented in the Turtle module (see Lab #7).

Organization systems coordinates on canvas shown on Fig.8.3.



This figure shows that the x-axis is directed from left to right, The ordinate axis runs from top to bottom. Note that the canvas dimensions (MaxX, MaxY) and point coordinates (Xi, Yi) are specified in pixels.

Given this feature of the module, as well as the fact that the user must enter or receive coordinates in their own coordinate system, we will have to convert the data entered by the user into pixels for drawing the graph lines and return the data converted to the user's coordinate system.

Let's deduce formulas transformations coordinates graphics, certain in the user's measurement system, in pixels. Let:

- Xmin, Xmax, Ymin, Ymax – minimum and maximum Coordinates displayed on the canvas for the X and Y axes, respectively. Data is presented in conventional units of length: centimeters, meters;

- MaxX, MaxY – canvas sizes along the X and Y axes, respectively, in pixels.

Then the scaling factors showing how many pixels there are per unit length of the user's measurement system are calculated along the abscissa and ordinate axes as follows:

$$Kx = \frac{MaxX}{Xmax - Xmin} \quad Ky = \frac{MaxY}{Ymax - Ymin}$$

Thus, for some point with coordinates Xi, Yi, expressed in the coordinate system of the graph (centimeters, meters, ...), we can obtain coordinates Xp, Yp, which display the position of the point in canvas pixels:

$$Xp = Kx \cdot Xi; \quad Yp = MaxY - Ky \cdot Yi.$$

In second expression taken into account, That fact, What axis ordinate V The coordinate system of the web is directed downwards.

Let's write script, which will show, How is used Canvas widget .

```
from tkinter import * # Import module
root = Tk() # Create the main window
root.title("Graphics") # Caption in the window title
root.resizable(False, False) # Not change size windows
                                # By width And
height Kp = 0.7 # use 70% of the screen
MaxX = root.winfo_screenwidth()* Kp # Current MaxY
dimensions = root.winfo_screenheight()* Kp # display
monitor
                                # V pixels
cv = Canvas(root, width = MaxX, height = MaxY,
            bg = "white")
cv.grid(row = 0, columnspan = 11)
```

IN first line script is being carried out import Tkinter module .

Please note that the module name is written with a lowercase letter letters . Then the script's main window is created, and its name is displayed in the form's header.

Resizing a window will require additional attention and processing, For example, redrawings graphs. Let's take it off this one problem, disabling window resizing. This is done using the following method:

```
resizable(bW, bH) ,
```

Where parameters bW - change size By width, And bH - resizing by height, take the logical value False or True .

The main window methods can be used to obtain the monitor display dimensions.

```
wininfo_screenwidth()    # Width V pixels  
wininfo_screenheight()   # Height in pixels
```

These dimensions vary greatly between monitors, so let's set the canvas dimensions to be 70% smaller than the display dimensions.

A widget is bound to a form when it's defined. This creates a widget ID, the value of which can be assigned to a variable. Please note the line, describing Canvas widget :

```
cv = Canvas(root,...)
```

root - main window name, cv – the name of the canvas we'll use when drawing primitives. Note that multiple canvases can be created on a form.

In our example, widget packing will be performed using the `grid()` method . As arguments, we pass the row number on which The canvas will be placed and how many columns it will occupy. We specified row zero because the widget will be placed at the top of the form. The number of columns occupied is 11. Refer to Fig. 8.2 and calculate how many widgets will be placed horizontally in the first row if each widget occupies one cell of the imaginary table. Note that the row and column numbers starts from zero and the first line is occupied by the canvas.

Create script With specified text And take a look on result.

Let's add to our form the labels (`Label`) that we drew on our layout, Fig. 8.3.

To describe labels, use the `Label()` method , whose arguments require the name of the window to which the label belongs. Additionally, you can specify the text and the width of the window in which you want to Place text, text parameters (color, font type and size), and text alignment parameters (left, center, right). The alignment value is specified either by the word `CENTER` (center) or by the symbol `or` combination symbols. Symbols - This first letters

English words, describing sides Sveta: **N** orth - North, **E** ast - East, **S** outh - South And **W** est - West, Fig. 8.4.

NW	N	NE
W	CENTER	E
SW	S	SE

Fig.8.4. - Directions text alignment

Object identification code, which is returned by the method

Label() , we will assign variable. Let's describe All eight tags ours layout.

```
lba0 = Label(root, text = "X:", width = 10,
              fg = "blue", font = "Ubutu, 12")
lba0.grid(row = 1, column = 0, sticky='e')
lba1 = Label(root, text = "Y:", width = 10,
              fg = "blue", font = "Ubutu, 12")
lba1.grid(row = 2, column = 0, sticky='e')
lba2 = Label(root, text = "Xmin:", width = 10,
              fg = "blue", font = "Ubutu, 12")
lba2.grid(row = 1, column = 2, sticky='e')
lba3 = Label(root, text = "Xmax:", width = 10,
              fg = "blue", font = "Ubutu, 12")
lba3.grid(row = 1, column = 4, sticky='e')
lba4 = Label(root, text = "Ymin:", width = 10,
              fg = "blue", font = "Ubutu, 12")
lba4.grid(row = 2, column = 2, sticky='e')
lba5 = Label(root, text = "Ymax:", width = 10,
              fg = "blue", font = "Ubutu, 12")
lba5.grid(row = 2, column = 4, sticky='e')
lba6 = Label(root, text = "Шар меток:", width = 10,
              fg = "blue", font = "Ubutu, 12")
lba6.grid(row = 1, column = 6, sticky='e')
lba7 = Label(root, text = "Смещение:", width = 10,
              fg = "blue", font = "Ubutu, 12")
lba7.grid(row = 2, column = 6, sticky='e')
```

The label placements specified in the grid() wrapper were calculated according to the previously designed layout, see Figure 8.2. Add this code to the previous one and see the result.

To complete the form, you need to add fields for input and output of data—Entry widgets . These fields are described in a completely analogous way to labels, for example:

```
ent0 = Entry(root, width = 5, font = "Ubuntu, 12")
```

```
ent0.grid(row = 1, column = 1)
```

Insert meaning string type V field Entry Can method

`insert()` :

```
ent0.insert(index, s) ,
```

where the string `s` is inserted before the character whose position is specified by the `index` parameter.

Delete symbols from widget Can method `delete()` :

```
ent0.delete(first, last=None) ,
```

where the parameter `first` specifies the starting position of the characters to be deleted, and the `last` parameter specifies the last position. In this case, the position specified in `last` is not turns on into the range to be deleted. If `last` is not specified, only one character is removed. If `END` is specified as the second parameter, all characters are removed.

Read meaning, introduced user V field, Can method

`get()` :

```
var = float(ent0.get())
```

The value read is of type string. Therefore, when inserting a numeric value, it must be converted to a string. type `str()`. When reading from the Entry field A numeric value must be converted back to the expected type. In our script, we will use real values: `float()`.

Exercise:

Add Entry widgets independently. There should be eight of them. The names of the variables with which you can access these widgets, compose according to that the same principle as the names for labels. That is, `entN`, where `N` – widget number.

Two first fields (`ent0` and `ent1`) are used to display mouse coordinates. Initialize them to zero, and initialize the remaining input fields to their initial values:

```
Xmin = -10.0; Xmax = 10.0 # Borders By abscissa
Ymin = -5.0; Ymax = 5.0   # Borders By ordinate
dX = 50   # Step of labels along axes
dY = 1.0  # Bias graphics
```

Note : Some variables used in our script will be global, What will allow use them in various functions without procedure transmissions. We received So, What would reduce number parameters passed to functions.

The designation of variables as global should be done more rationally.

Check it out work script.

Now you can add buttons (`Button`). When describing these widgets, you can specify the caption that will be placed on the button, parameters `text` (font, color) And What Very important, tie pressing

buttons to call a function that will receive the event from the button click and process it.

```
btn1 = Button(root, width = 20, bg = "#ccc",
               text = "Рисовать")
btn1.grid(row = 1, column = 10)
btn1.bind("<Button-1>", Draw)
```

The first line specifies the button ID (btn1), the root parameter indicates that the button is on the form, the remaining parameters specify the size, color background And text. The grid() method will place button V first line in the tenth column.

To handle the event associated with the button, the Draw() function will be called . Note that the Draw variable recorded without round brackets .

Code For second buttons will similar to:

```
btn2 = Button(root, width = 20, bg = "#ccc",
               text = "Выход")
btn2.grid(row = 2, column = 10)
btn2.bind("<Button-1>", Final)
```

Before those, How go to To description functions (head parts), Let's add a link between the "mouse click on canvas" event and the corresponding handler. Since our canvas identifier is named cv , the link string will look like this:

```
cv.bind('<Button-1>', showXY) ,
```

where is showXY A function that will be called by the '<Button-1>' event . Optionally, you can select a double-click as the event. In this case, we would specify the code <Double-Button-1> .

Read more O events And binding events V Internet.

The final step in creating a script window is to set up a continuous loop in which events will be processed, see Fig. 8.1.

To organize the cycle in which the script will be located When waiting for events, the mainloop() method is used :

```
root.mainloop()
```

The basic steps for preparing the window are complete. During the scripting process, three conditions for triggering events were created: a button "Draw," the "Exit" button, and a left-click event. We'll implement handling of these events as functions.

```
def Drav(event):
    '''Call functions For drawing graphs'''
    pass
#
```

```

def Final(event): '''
    Completion '''
    window_deleted() works
#
def window_deleted():
    ''' We're finishing up work By [X]'''
    if askyesno("Exit", "Finish work?"): root.destroy()
#
def showXY(event):
    ''' Conclusion coordinates mice
    ''' global ID1, ID2
    x = event.x; y = event.y
    ent0.delete(0,END) # Let's delete it
    old ent1.delete(0,END) # values
# Let's calculate And let's introduce new
    ent0.insert(0, str(round(Xmin + x/Kx, 2)))
    ent1.insert(0, str(round(Ymin + (MaxY - y)/Ky, 2)))
    cv.delete(ID1) # Let's delete old
    cv.delete(ID2) # lines
# Let's draw new lines

```

ID1 = cv.create_line(0,y,MaxX,y, dash = (3.5))
 ID2 = cv.create_line(x,0,x,MaxY, dash = (3.5))
 Mark, What V quality parameter functions indicated variable
 event . Through this one variable V function transmitted description events.

Draw() function will call other functions. We haven't developed them yet,
 and we're using the pass statement. This will allow us to test the script earlier.

Final() function The script should be triggered by clicking the "Exit"
 button and terminate the script. Since the script could terminate accidentally, we'll
 create a window asking for confirmation to close and two buttons, "Yes" and "No."

Tkinter has a module called `tkinter.messagebox` , which contains a
 set of windows with different functionalities. In our case, these are:

```
askyesno("Exit", "Finish work?")
```

Here `askyesno()` method, formative A query window with two buttons. Strings
 are specified as parameters. The first string is the window title, and the second is
 the message that will appear in the window. Clicking a button returns a Boolean
 value: True means the "Yes" button was clicked, and False means the "No" button
 was clicked.

There's another way to close a window—the [X] button in the upper-right
 corner of the form. To handle this event, use the `protocol()` method ,
 which intercepts the event associated with closing the window using the [X]
 button:

```
root.protocol('WM_DELETE_WINDOW', window_deleted)
```

Here, 'WM_DELETE_WINDOW' is the name of the protocol associated with the close window button ([X]), and `window_deleted` is the function that handles the event. Note that when handling this event, its parameters are not passed to the function.

Let's formulate this as two functions. One is called when the window is closed using the [X] button and displays a confirmation prompt, while the second calls the first. The window itself is closed by calling the `destroy()` method :

```
root.destroy()
```

The `showXY()` function Called by a "single left-click" event on the canvas, this function outputs the coordinates of the point on the canvas where the click occurred to the `Entry` fields . The implementation of this function is complicated for the following reasons:

- Point coordinates are two numbers representing the point's position in pixels in the canvas coordinate system. They need to be converted to the user's coordinate system. To do this, we use knowledge of the scale. along each axis. The resulting numbers will be real numbers, so we'll round them to two decimal places. Before outputting them to the `Entry` field numbers need to be converted to string type;

- point, V which produced cry, necessary visualize. For this Let's draw two perpendicular dashed lines that pass through a given point and are parallel to the coordinate axes. The method used to draw the lines returns an identifier that we use to delete the lines before re-rendering the graph.

Please refer to the function code above for a more complete understanding of this material. Note that the `cv.delete(ID1)` method does not raise an error if the `ID` is undefined: before the first draw, we delete the non-existent object.

Coordinates axes

To evaluate the values a function takes, coordinate rulers with markings are drawn on the canvas. The `plotXY()` function draws rulers along the edges of the canvas.

To draw a coordinate ruler, a rectangle is first drawn, then short line markers and text values corresponding to these markers are added to its sides. When a value is added, it is calculated, rounded to two decimal places, and then converted to a string.

In addition to this, we have implemented a method for reading the canvas coordinates using the left mouse click event.

Drawing functions

To draw functions, function values are calculated in a range From X_{min} to X_{max} . This uses a user-defined coordinate system. The abscissa and ordinate values are combined to form a list that is returned when the function is called. Since the task requires smooth transitions between the calculated function points, the calculations are performed in increments of one pixel along the X-axis. In a user-defined coordinate system, the increment is equal to $1/Kx$, where Kx is the scale factor along the X-axis.

Drawing functions begins with the occurrence of an event associated with a click on the "Draw" button. This triggers the Drav() function, which sequentially executes the following operations:

Cleaning canvases from previously drawn objects;

Receipt data from fields Entry. Data, at reading, are checked for an acceptable range of values. If an error occurs at this step, a warning message is displayed and The user can enter new data. The script continues to run;

It is drawn coordinate ruler By edges canvases;

The function that draws the graph is called. When called, the name of the function to be drawn and the color to be used for the graph are specified as arguments.

Listing script

Here is a listing of the script that solves the problem. It is fully functional, but could be significantly improved, for example, by The main part of the script can be implemented as a separate module. The user can use this module to display other functions graphically. They simply need to create their own script, import our module, and describe the functions for which the graph should be plotted.

But this topic is not developed here. Students are encouraged to independently consider possible improvements to the script.

```
from tkinter import *
from tkinter.messagebox import *
from math import sin, cos, pi, exp
#
def Sin2xDevx(): # Функция из задания
    eps = 0.0001
    Lfun = []
    x = Xmin
    while x <=Xmax:
        an = 1
        Sum = an
        n = 1
```

```

        while (abs(an) > eps):
            an *= -(2**2)*(x**2)/((2*n)*(2*n+1))
            Loss +=
            and n +=
            1
        Fun.append((x, 2*Sum))
        x += 1 / Kx
    return Fun
do Sin2xOnX(): # Function from tasks
    Lfun = []
    x = Xmin
    while x <= Xmax:
        if x != 0:
            for = sin(2*x)/x + dY
        else:
            fun = 2.0
        Fun.append((x, for))
        x += 1 / Kx
    will Fun
def GetData(): ''' Get
data '''
global Xmax, Xmin, Ymax,
Global Ymin dX, dY, Kx, Ky
tmpXmax = float(ent3.get())
tmpXmin = float(ent2.get())
tmpYmax = float(ent5.get())
tmpYmin = float(ent4.get())
tmpdY = float(ent7.get())
tmpdX = float(ent6.get())
if ((tmpXmin >= tmpXmax) or
    (tmpYmin >= tmpYmax) or
    (tmpdX <= 0)):
    showwarning(title = " Error tasks boundaries ",
                message = " Must to be carried out "
                " inequalities :\n"
                "Xmax > Xmin;\n"
                "Ymax > Ymin;\n" "
                Step tags > 2")
else:
    Xmax = tmpXmax
    Xmin = tmpXmin
    Ymax = tmpYmax
    Ymin = tmpYmin
    dX = tmpdX

```

$$dY = tmpdY$$

```

        Kx = MaxX / abs((Xmax - Xmin))
        Ky = MaxY / abs((Ymax - Ymin))
def SetMark(a, b, LrBt = 1):
    ''' Application labels
    ''' ax_XY = []
    if LrBt: # слева и справа
        ax_XY.append((a, b))
        ax_XY.append((a + 10, b))
    else:    # внизу и вверху
        ax_XY.append((a,b))
        ax_XY.append((a, b - 10))
    cv.create_line(ax_XY, fill='black', width = 2)
def plotXY():
    ''' Let's draw coordinate
rulers''' # Rectangle
    ax_XY = []
    ax_XY.append((5, 5))
    ax_XY.append((MaxX - 5, MaxY - 5))
    cv.create_rectangle(ax_XY, fill="white",
                        outline = "green", width = 2)
#
# Markup left And right sides y
    = Ymin
    y_pix = MaxY
    flg = False
    while y < Ymax:
        textY = str(round(y, 2)) # Text marks
        SetMark(0, y_pix, 1)    # Left
        if flg: # Let's inscribe every the second
            label cv.create_text(15, y_pix, text =
                textY,
                                anchor = W)
        SetMark(MaxX-10, y_pix, 1) # On the
        right if flg:
            cv.create_text(MaxX - 15, y_pix,
                            text = textY, anchor = E)
            y += dX # Units user
        y_pix -= dX * Ky # Units V pixels
        flg = not flg   # Markup through once
#
# Markup above And from
    below x = Xmin
    x_pix = 0
    flg = False

```

```
while x < Xmax:
```

```

    textX = str(round(x, 2)) # Text marks
    SetMark(x_pix, 0, 0)    # Top
    if flg:
        cv.create_text(x_pix, 15, text = textX,
                        anchor = N)
    SetMark(x_pix, MaxY, 0) # Below
    if flg:
        cv.create_text(x_pix, MaxY - 15,
                        text = textX, anchor = S)

    x += dX
    x_pix += dX*Kx
    flg = not flg

def Draw(event):
    '''Preparation canvases And
    Calling drawing
    functions'''
    cv.delete("all") # Cleaning canvas
    GetData()        # Get data
    plotXY()         # Draw coordinate rulers
    Fdraw(Sin2xDevx, 'blue') # Draw function 1
    Fdraw(Sin2xOnX, 'red')  # Draw function 2
    print('Drawing')      # Test message

def Fdraw(func, color):
    '''Getting values functions
    Transformation V Pixel
    Drawing on Canvas
    Lxy = func() # Values functions V units user Lpix
    = []        # Function values in pixels
    for xy in Lxy:
        x = Kx * (xy[0] - Xmin)
        if xy[1] != None:
            y = MaxY - Ky * (xy[1] - Ymin)
        else:
            y = 0
        Lpix.append((x,y))
    cv.create_line(Lpix, fill = color)

def Final(event):
    ''' We're finishing up work
    ''' window_deleted()

def window_deleted():
    ''' We're finishing up work By [X]'''
    if askyesno("Exit", "Finish work?"): root.destroy()

def showXY(event):

```

```

global ID1, ID2
x = event.x; y = event.y
ent0.delete(0,END)
ent1.delete(0,END)
ent0.insert(0, str(round(Xmin + x/Kx, 2)))
ent1.insert(0, str(round(Ymin + (MaxY - y)/Ky, 2)))
cv.delete(ID1)
cv.delete(ID2)
ID1 = cv.create_line(0,y,MaxX,y, dash = (3,5))
ID2 = cv.create_line(x,0,x,MaxY, dash = (3,5))

#
root = Tk()
root.title("Графика")
# Handler closings windows . Pressed button [X]
root.protocol('WM_DELETE_WINDOW', window_deleted)
root.resizable(False, False) # Not we change size
windows #
Kp = 0.7 # 70% from the display
# Initial parameters canvases. Unit amend. - pixels
MaxX = root.winfo_screenwidth() * Kp
MaxY = root.winfo_screenheight() * Kp
#
cv = Canvas(root, width = MaxX,
             height = MaxY, bg = "white")
cv.grid(row = 0, columnspan = 9)
cv.bind('<Button-1>', showXY) # Cry on canvas #
# Window graphics. Unit amend. - user Xmin = -
10.0; Xmax = 10.0
Ymin = -5.0; Ymax = 5.0
Xmid = 0; Ymid = 0 # Position beginning
coordinates #
# Bias And step argument. Unit amend. - custom dY = 1.0
# offset of the second graph
dX = 1.0 # step tags on coordinate rulers
# Identifiers cursors, drawn By click on canvas ID1 = 0;
ID2 = 0
# Large-scale coefficients: Pixel / user unit
Kx = MaxX / abs((Xmax - Xmin))
Ky = MaxY / abs((Ymax - Ymin))
#
lba0 = Label(root, text = "X:", width = 10,
             fg = "blue", font = "Ubutu, 12")
lba0.grid(row = 1, column = 0, sticky='e')
ent0 = Entry(root, width = 5, font = "Ubuntu, 12")

```

```

ent0.grid(row = 1, column = 1, sticky='w')
ent0.insert(0, 0)
lba1 = Label(root, text = "Y:", width = 10,
              fg = "blue", font = "Ubutu, 12")
lba1.grid(row = 2, column = 0, sticky='e')
ent1 = Entry(root, width = 5, font = "Ubuntu, 12")
ent1.grid(row = 2, column = 1, sticky='w')
ent1.insert(0, 0)

lba2 = Label(root, text = "Xmin:", width = 10,
              fg = "blue", font = "Ubutu, 12")
lba2.grid(row = 1, column = 2, sticky='e')
ent2 = Entry(root, width = 5, font = "Ubuntu, 12")
ent2.grid(row = 1, column = 3)
ent2.insert(0, Xmin)

lba3 = Label(root, text = "Xmax:", width = 10,
              fg = "blue", font = "Ubutu, 12")
lba3.grid(row = 1, column = 4, sticky='e')
ent3 = Entry(root, width = 5, font = "Ubuntu, 12")
ent3.grid(row = 1, column = 5)
ent3.insert(0, Xmax)

lba4 = Label(root, text = "Ymin:", width = 10,
              fg = "blue", font = "Ubutu, 12")
lba4.grid(row = 2, column = 2, sticky='e')
ent4 = Entry(root, width = 5, font = "Ubuntu, 12")
ent4.grid(row = 2, column = 3)
ent4.insert(0, Ymin)

lba5 = Label(root, text = "Ymax:", width = 10,
              fg = "blue", font = "Ubutu, 12")
lba5.grid(row = 2, column = 4, sticky='e')
ent5 = Entry(root, width = 5, font = "Ubuntu, 12")
ent5.grid(row = 2, column = 5)
ent5.insert(0, Ymax)

lba6 = Label(root, text = "Мар меток:", width = 10,
              fg = "blue", font = "Ubutu, 12")
lba6.grid(row = 1, column = 6, sticky='e')
ent6 = Entry(root, width = 5, font = "Ubuntu, 12")
ent6.grid(row = 1, column = 7)
ent6.insert(0, dX)

lba7 = Label(root, text = " Offset :", width = 10,

```

```

fg = "blue", font = "Ibutu, 12")
lba7.grid(row = 2, column = 6, sticky='e')
ent7 = Entry(root, width = 5, font = "Ubuntu, 12")
ent7.grid(row = 2, column = 7)
ent7.insert(0, dY)

btn1 = Button(root, width = 20, bg = "#ccc",
               text = "Рисовать")
btn1.grid(row = 1, column = 8)
btn1.bind("<Button-1>", Draw)

btn2 = Button(root, width = 20, bg = "#ccc",
               text = "Выход")
btn2.grid(row = 2, column = 8)
btn2.bind("<Button-1>", Final)

root.mainloop() # Cycle expectations events

```

Result work script

On Fig. 8.5 presented realized form With graphs.

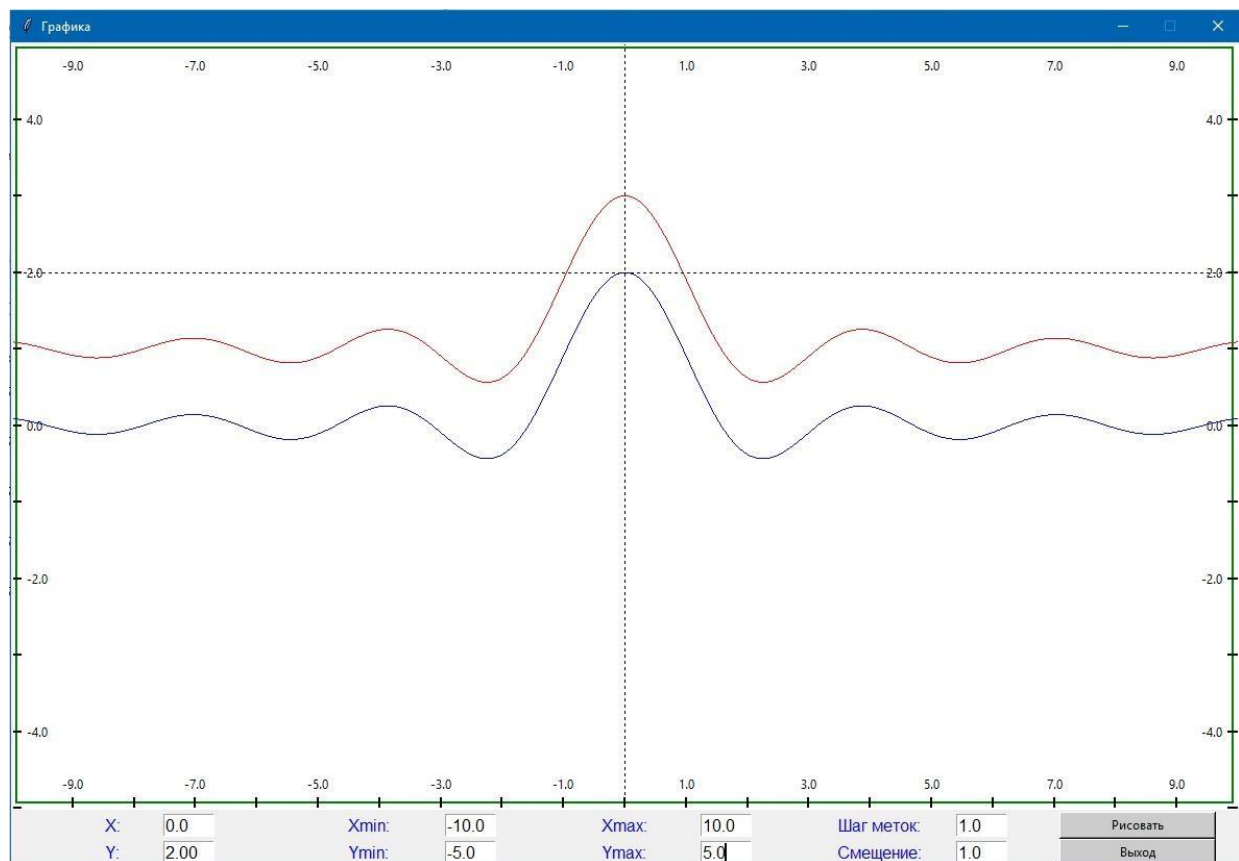


Fig.8.5. - Result script operation

Exercise To practical work No. 8

GUI, module Tkinter»

The image should occupy most of the screen, be accompanied by a title, contain axis names and gradations, and be scaled depending on the source data. For any acceptable values of the source data data image should fully place on screen. The program should not rely on specific values screen resolution. Display on screen V graphic mode graphs two functions on the interval from Xstart to Xcon with a step dx. The first function is defined using a Taylor series, its calculation must be performed with an accuracy of ε . The value parameter b For second functions is introduced With keyboards. Charts should

be smooth And differ flowers.

IN options with 12 By 20 required build graphic image in the form of pie or bar charts. To solve these problems, you need to become more familiar with the Tkinter module widgets than described above.

Option 1

$$y(x) = 2 \cdot \sum_{n=0}^{\infty} \frac{1}{(2 \cdot n + 1) \cdot x^{2 \cdot n + 1}} = 2 \cdot \left(\frac{1}{x} + \frac{1}{3 \cdot x^3} + \frac{1}{5 \cdot x^5} + \dots \right), \quad |x| > 1;$$

from $\left(x \cdot \frac{x+1}{x-1} + b \right)$
 $= \ln$

Option 2

$$y(x) = \sum_{n=0}^{\infty} \frac{(-1)^n \cdot x^n}{n!} = \left(1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \frac{x^4}{4!} - \dots \right), \quad |x| < \infty;$$

$z(x) = \exp^{-x} + b;$

Option 3

$$y(x) = \sum_{n=0}^{\infty} \frac{x^n}{n!} = \left(1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \dots \right), \quad |x| < \infty$$

$z(x) = x + b.$

Option 4

$$y(x) = \frac{\pi}{2} + \sum_{n=0}^{\infty} \frac{(-1)^{n+1}}{(2 \cdot n + 1) \cdot x^{2n+1}} = \frac{\pi}{2} - \frac{1}{x} + \frac{1}{3 \cdot x^3} - \frac{1}{5 \cdot x^5} + \dots, \quad x > 1;$$

$\square_{n+1}$

$z(x) = \arctg x + b.$

Option 5

$$y(x) = \sum_{n=0}^{\infty} \frac{(-1)^n \cdot x^{2n+1}}{(2n+1)!} = \frac{x^3}{3!} - \frac{x^5}{5!} + \dots$$

$$-x^7+\dots,\quad x\leq 1$$

$$z\left(x\right)=arctgx+b.$$

Option 6

$$y(x) = 2 \cdot \sum_{n=0}^{\infty} \frac{1}{(2 \cdot n + 1) \cdot x^{2 \cdot n + 1}} = 2 \cdot \left(\frac{1}{x} + \frac{1}{3 \cdot x^3} + \frac{1}{5 \cdot x^5} + \dots \right), \quad |x| > 1;$$

$$z(x) = \operatorname{Arth} x + b.$$

Option 7

$$\text{and } (x) = - \sum_{n=0}^{\infty} \frac{(-1)^n}{(2 \cdot n + 1) \cdot x^{2 \cdot n + 1}} = - \frac{1}{2} - \frac{1}{x} + \frac{1}{3 \cdot x^3} - \frac{1}{5 \cdot x^5} + \dots, \quad x < -1$$

$$z(x) = \operatorname{arctg} x + b.$$

Option 8

$$y(x) = \sum_{n=0}^{\infty} \frac{(-1)^n \cdot x^{2 \cdot n}}{n!} = 1 - x^2 + \frac{x^4}{2!} - \frac{x^6}{3!} + \frac{x^8}{4!} - \dots, \quad |x| < \infty$$

$$z(x) = e^{x^2} + b.$$

Variant 9

$$y(x) = \sum_{n=0}^{\infty} \frac{(-1)^n \cdot x^{2 \cdot n}}{(2 \cdot n)!} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \dots, \quad |x| < \infty$$

$$z(x) = \cos x + b.$$

Option 10

$$y(x) = \sum_{n=0}^{\infty} \frac{(-1)^n \cdot x^{2 \cdot n}}{(2 \cdot n + 1)!} = 1 - \frac{x^2}{3!} + \frac{x^4}{5!} - \frac{x^6}{7!} + \dots, \quad |x| < \infty$$

$$z(x) = \frac{\sin x}{x} + b.$$

Option 11

$$y(x) = 2 \cdot \sum_{n=0}^{\infty} \frac{(x-1)^{2 \cdot n}}{(2 \cdot n + 1) \cdot (x+1)^{2 \cdot n + 1}} = 2 \cdot \left(\frac{1}{x+1} + \frac{(x-1)^3}{3 \cdot (x+1)^3} + \frac{(x-1)^5}{5 \cdot (x+1)^5} + \dots \right), \quad x > 0;$$

$$z(x) = \ln x + b.$$

Option 12

Write a program that displays a pie chart. Provide the chart with a title and a name for each sector. Create the source data in a text file. Specify the number of sectors in the program as a named constant.

The construction of a pie chart should be implemented as a procedure. The procedure parameters are: coordinates of the chart center; radius; number of sectors; percentage array; array of names. For an example of the initial data, see

Table 1.

Option 13

Write a program that displays two pie charts side by side. Provide the chart with a title and a name for each sector. Create the source data in a text file. Specify the number of sectors in the program as a named constant.

The construction of a pie chart should be implemented as a procedure. The procedure parameters are: coordinates of the chart center; radius; number of sectors; percentage array; array of names. For an example of the initial data, see Table 1.

Option 14

Write a program that displays two bar charts. Place the charts side by side on the screen, each on its own coordinate axes. Provide each chart with a title and the units of measurement for the X and Y axes. Generate the initial data in a text file. Quantity columns to specify in the program as a named constant. The diagram construction should be formatted as a procedure. See Table 1 for an example of the source data.

Option 15

Write a program that displays two bar charts in the same coordinate plane. The chart Provide axis gradations and a title. Generate the source data in a text file. Specify the number of columns in the program as a named constant. Create a procedure for plotting the chart. See Table 1 for an example of source data.

Option 16

Write a program that displays a three-dimensional bar chart. Provide the chart with axis gradations and a title. Create the source data in a text file. Specify the number of columns. V program V view named constants. Construction The diagrams should be formatted as a procedure. For an example of the initial data, see Table 1.

Option 17

Write a program that displays a bar chart representing wholesale and retail prices for various coffee brands. The input data should be stored in a text file.

Design the diagram construction as a procedure. Procedure parameters: number of names; array of values wholesale prices; array of retail price values; array of product names. Place product names vertically below the x-axis.

Option 18

Write a program that displays a bar chart representing the maximum and average profit margins for different types of chocolate. Create the source data yourself in a text file.

Design the diagram construction as a procedure. Procedure parameters: number of names; array of values wholesale prices; array of retail price values; array of product names. Place product names vertically below the x-axis.

Option 19

Write a program that displays graphs of the maximum, minimum, and average dollar exchange rates over a specified number of days. Create the source data yourself in a text file.

Design the graph plot as a procedure. Procedure parameters: date array; number of days; arrays of maximum, minimum, and average values.

Option 20

Write a program that displays a three-dimensional bar chart of the German mark's exchange rate against the ruble over a specified number of days. Create the input data yourself in a text file.

Design the chart as a procedure. Procedure parameters: array of dates; number of days; array of Y-axis values; placeholder code.

Example original data For options 12 - 16 Table 1. –

Leaders of the global PC market

Rating 1996 G.	Supplier	Volume Sales 1996 d. thousand. pcs.	Share Market 1996 g., %	Volume Sales 1995 d. thousand. pcs.	Share Market 1995 g., %	Height 95/96, %
1	Compaq	7 036	10.3	5 757	9.8	22
2	IBM	6 081	8.9	4 785	8,1	27
3	Packard	4 247	6,2	4 392	7,5	-3
	Bell, NEC	3 587	5,2	4 627	7,9	22
4	Apple	2 995	4,4	2 023	3,4	48
5	HP	44 459	65.0	37 221	63.3	19
6	Other					

Note: Try it update information, having received necessary data from Internet networks .

Conclusion

The material presented in this manual contains a large amount of program code. During preparation for publication, all programs in the manual were tested and debugged using the WinPython interpreter.

Versions component : Python ver: 3.5.4, Tkinter ver: 8.6.4, IDLE ver: 3.5.4.

It is expected that the reader will carefully review the information presented and be able to independently identify potential errors that often arise during the publication's preparation. I would like to point out that correcting errors allows for a more thorough understanding of the material presented, and therefore it is strongly recommended that the provided programs be tested for functionality.

The solution algorithms used in this manual are not perfect. Moreover, they are written in such a way that an untrained reader can understand the main idea of the proposed solution. All programs can be further refined or modified to achieve a more acceptable solution, according to the reader's wishes.

PS: This teaching aid, along with supplementary material, is intended to be distributed electronically. Please contact the authors for information on where to place it and how to obtain the electronic version.

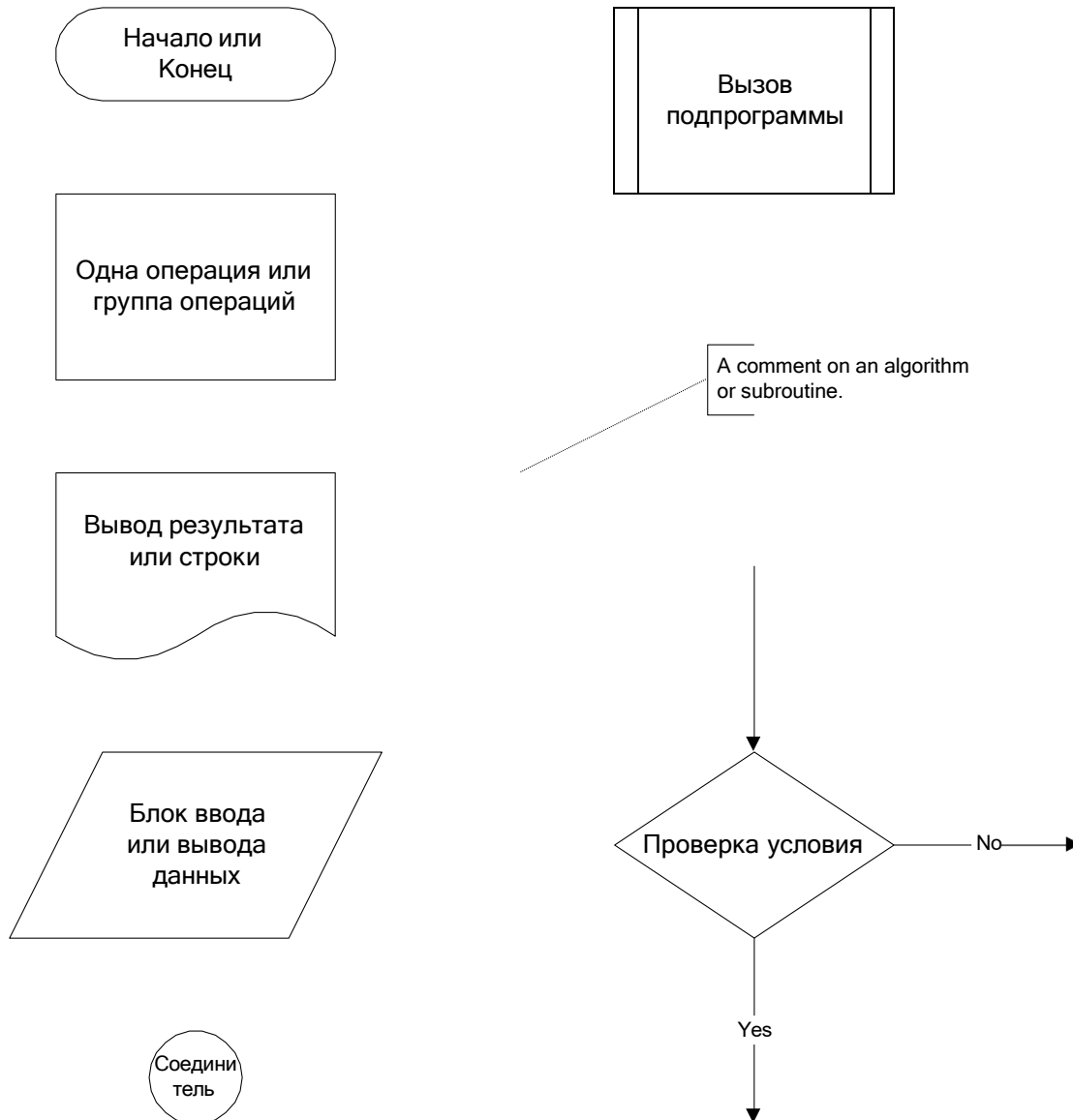
Table 1. - Functions And methods V Python 3

Constants (Math)	Description And example use
pi	Returns number π . <code>math.pi</code> => 3.141592653589793
e	Returns number e. <code>math.e</code> => 2.718281828459045
Functions (Sys)	
<code>int([<Object>[,<Number system>]])</code>	Transforms an object V whole number. Second The parameter specifies the number system of <u>the number being converted</u> . By default, the decimal number system is used. <pre>>>>int(7.5), int('71',10), int('0o71',8) *) (7, 71, 57) >>>int("0xA",16) 10 >>>int(), int("0b11111111",2) (0, 255)</pre>
<code>float([<Number string>])</code> or	Transforms whole number or line V material number. <pre>>>>float(7), float("7.1"), float("12.") (7.0, 7.1, 12.0) >>>float("inf"), float("-inf"), float("nan") (inf, inf, nan)</pre>
<code>bin(<Number>)</code>	Converts a decimal number to binary. Returns a string representation of the number. <pre>>>>bin(255), bin(1), bin(- 45) ('0b11111111', '0b1', '- 0b101101')</pre>
<code>oct(<Number>)</code>	Transforms decimal number V octal. Returns a string representation of a number. <pre>>>>oct(7), oct(8), ovt(64) ('0o7', '0o10', '0o100')</pre>
<code>hex(<Number>)</code>	Converts a decimal number to hexadecimal. Returns a string representation of the number. <pre>>>>hex(10), hex(16), hex(255) ('0xa', '0x10', '0xff')</pre>
<code>round(<Number>[, Numb er of characters after the period])</code>	Material number is rounded off to whole By the following rule: fractional part less 0.5 - rounding down; equals 0.5 – rounding to even numbers; greater than 0.5 – round up. <pre>>>>round(1.49), round(1.50), round(1.51) (1, 2, 2) >>>round(2.49), round(2.50), round(2.51) (2, 2, 3)</pre> Second parameter defines number numbers after comma. IN In this case, rounding is performed according to the rule: the discarded part

*) In Python, you can use single and double straight quotes . It is important that the initial and The ending quotation marks were the same. When typing a program in Word, you need to set the appropriate points autocorrects: File -> Parameters -> Spelling -> Parameters autocorrect (for Office 2010)

	less 0.5 - rounding down; more or equals 0.5 - rounding up. <pre>>>> round(2.449, 1), round(2.450, 1), round(2.451, 1) (2.49, 2.5, 2.5)</pre>
abs(<Number>)	Returns absolute meaning.
pow(<Number>, <Power>, [, <Divisor>])	Returns number, erected V degree. Degree Maybe be of real type. If a third parameter is specified, the remainder of the result divided by the value of this parameter is returned. <pre>>>>pow(3, 3), pow(3, 3, 2), pow(3, 3.25) (27, 1, 35.533998349717294)</pre>
max(<List numbers through comma>)	Returns maximum number from the list.
min(<List numbers through comma>)	Returns minimum number from the list.
sum(<Sequence> [, <Initial value>])	Returns amount values elements sequence (list, tuple) plus initial value. <pre>>>>sum([1, 2, 3], 7), sum((1, 2, 3)) (13, 6)</pre>
divmod(x, y)	Returns motorcade from two values: whole Part And remainder from divisions: $x // y$ And $x \% y$.
Functions (Math)	
sin(x), cos(x), tan(x)	Standard trigonometric functions. Corner is being asked V <u>radians</u> .
asin(x), acos(x), atan(x)	Reverse trigonometric functions
degrees(x)	Transformation radian V degrees
radians(x)	Transformation degrees V radians
exp(x)	Exponent
log(<Number>[, <Base>])	Logarithm numbers By basis <Base>. If base Not indicated, That is calculated natural logarithm.
log10(x), log2(x)	Decimal and binary logarithms (base 10 and base 2) respectively)
sqrt(x)	Square root
ceil(x)	Rounding to nearest greater whole
floor(x)	Rounding to nearest smaller whole
fabs(x)	Returns absolute meaning. Same, What And abs (x)
fmod(x, y)	Remainder from divisions. Same, What And $x \% y$.
factorial(n)	Factorial numbers.
fsum(<list numbers>)	Returns accurate amount numbers from given list. <pre>>>>sum([.1, .1, .1, .1, .1, .1, .1, .1, .1, .1]) 0.9999999999999999 >>>fsum([.1, .1, .1, .1, .1, .1, .1, .1, .1, .1]) 1.0</pre>

Symbols, used at compilation block diagrams algorithms (GOST 19.003-80)



At compilation programs V Word these symbols Can get via the menu:
"Insert" - Figures - "Block diagram".

To add text, right-click on the selected object and select "Add Text" from the context menu. You can use the "Shape Options..." context menu item to adjust the color, fill, and line thickness.

List literature

1. Many interesting And useful news from peace Python: <https://pythondigest.ru/>
2. Programming learning game: <https://checkio.org>
3. Online IDE: <https://repl.it>
4. Visualizer online on Python: <http://pythontutor.com/visualize.html#mode=edit>
5. Ranked Python quizzes: <https://www.hackerrank.com>
6. Video lectures "Programming on language Python For collection And data analysis "
7. Python Programming Course (Bioinformatics Institute)
8. Python: Fundamentals and Applications (Bioinformatics Institute)
9. Video lectures Python 3 Basics Tutorial Series»
10. Python Programming Course from Mail.Ru Group

MINISTRY OF SCIENCE AND HIGHER EDUCATION
RUSSIAN FEDERATION

FEDERAL STATE AUTONOMOUS EDUCATIONAL INSTITUTION OF
HIGHER EDUCATION
NORTH CAUCASIAN FEDERAL UNIVERSITY

GUIDELINES

to organize and conduct independent work
students in the discipline "Python Program Designing"
training areas 11.03.02 Infocommunication technologies and communication
systems, Profile " Intelligent data processing in infocommunication systems and
networks "

Stavropol, 2026

The guidelines for organizing independent student work are compiled in accordance with the requirements of the "Python Program Development" program for undergraduate students in the 11.03.02 Infocommunication Technologies and Communication Systems program. This teaching aid is dedicated to students' planned academic work, completed outside of class (or in class) time, based on assignments and with the instructor's guidance, but without their direct participation, or with their partial or direct involvement.

CONTENT

1. Independent work as the most important form educational process
2. Objectives and main tasks of independent work
3. Types independent work
4. Organization SRS
5. General recommendations for organization independent work
6. Independent work of a student is a necessary part of becoming a researcher
7. Methodological recommendations for students on individual forms of independent work
8. Educational, methodological and informational security disciplines

1. Independent work as the most important form of education process

Independent work is the planned educational, educational-research, scientific-research work of students, carried out outside of class (classroom) time on assignment and with the methodological guidance of the teacher, but without his direct participation (with partial direct participation of the teacher, leaving the leading role to the work of the students).

Initially, it is necessary to familiarize yourself with the technological map of independent work of the student.

No.	Section (topic) of the discipline and brief content	Developed competencies, indicators	full-time education				Independent work, hours	Forms of the current control academic performance
			Contact work of students with a teacher/of which in the form of practical training, hours					
			Lectures	Laboratory work	Practical classes			
4th semester								
1	Introduction to Python programming. Linear programs. Branching computational processes. Looping. Linear programs. Branching computing processes. Organization of cycles. Practical Work #1: "Linear Programs" Practical work #2: "Branching computing processes" Practical work No. 3: "Organization of cycles".	OPK-4 ID-1 OPK-4 ID-2 OPK-4 ID-3 OPK-4 ID-4 OPK-4			6	10	interview	
2	Functional programming in Python. Recursion. Functional programming in Python. Recursion.	OPK-4 ID-1 OPK-4 ID-2 OPK-4 ID-3 OPK-4 ID-4 OPK-4				10	interview	
3	Arrays. Sorting. Working with strings. Matrices. Arrays. Sorting. Working with strings. Matrices. Practical Work #4: "One-Dimensional Arrays" Practical Work #5: "Two-	OPK-4 ID-1 OPK-4 ID-2 OPK-4 ID-3 OPK-4 ID-4 OPK-4			6	10	interview	

	Dimensional Arrays and Functions"						
4	Working with files. Organizing the computing process. Working with structures and dictionaries. Working with files. Organization of the computing process. Working with structures and dictionaries. Practical Work #6: "Files"	OPK-4 ID-1 OPK-4 ID-2 OPK-4 ID-3 OPK-4 ID-4 OPK-4			6	10	interview
5	Working with stacks. Trees. Graphs. Working with the stack. Trees. Counts.	OPK-4 ID-1 OPK-4 ID-2 OPK-4 ID-3 OPK-4 ID-4 OPK-4				10	interview
6	Dynamic programming. Object-oriented programming in Python. Encapsulation, polymorphism, inheritance. Dynamic programming. Object-oriented programming in Python. Encapsulation, polymorphism, inheritance.	OPK-4 ID-1 OPK-4 ID-2 OPK-4 ID-3 OPK-4 ID-4 OPK-4				10	interview
7	Introduction to graphics in Python. Graphics programming. Introduction to Graphics in Python. Graphics programming. Practical Work #7: "Graphics Programming." Turtle Module Practical Work #8: "GUI, Tkinter Module"	OPK-4 ID-1 OPK-4 ID-2 OPK-4 ID-3 OPK-4 ID-4 OPK-4			6	10	interview
8	Working with Widgets. Model and View in Python. Working with widgets. Model and view in Python.	OPK-4 ID-1 OPK-4 ID-2 OPK-4 ID-3 OPK-4 ID-4 OPK-4				10	interview
9	Selected Algorithms in Python. Dijkstra's algorithm. Queen placement problem. Fibonacci numbers. The knapsack problem.	OPK-4 ID-1 OPK-4 ID-2 OPK-4 ID-3 OPK-4 ID-4 OPK-4				4	interview
	TOTAL for 3rd semester				24	84	
	TOTAL				24	84	

Independent student work at a university is an important form of academic and research activity. Independent student work plays a significant role in the rating system of learning. Studying at a university involves two parts, almost equal in scope and mutual influence: the learning process and self-study. Therefore, independent student work should be an effective and purposeful part of student work.

The concept for modernizing Russian education defines the main objectives of vocational education as "the preparation of a qualified worker of the appropriate level and profile, competitive in the labor market, competent, responsible, fluent in their profession and oriented in related fields of activity, capable of effective work in their specialty at the level of world standards, ready for continuous professional growth, social and professional mobility."

The solution to these problems is impossible without increasing the role of students' independent work on educational material, strengthening the responsibility of teachers for developing independent work skills, stimulating students' professional growth, and fostering creativity and initiative.

Society places a wide range of demands on modern professionals, among which a significant factor is the ability of graduates to independently acquire knowledge from various sources, systematize the information obtained, and evaluate specific financial situations. This ability is developed throughout the entire educational period through students' participation in laboratory and practical classes, completing assignments and tests, and writing coursework and final theses. Students' independent work plays a crucial role throughout the educational process.

Students' independent work options vary. The following are included in this course, "Python Programming":

Use of the material of the educational and methodological complex of the discipline

The first step is to familiarize yourself with the required minimum content of the core curriculum for the discipline being studied. Next, you need to review the course syllabus, which covers the content of lecture course topics, the relationship between lecture topics and laboratory/practical classes, and the topics and types of independent work. Specific reporting forms are provided for each type of independent work, as presented in the curriculum for the "Python Programming" course.

Working with literature

To successfully master the course, you must independently and thoroughly study the presented topics using the recommended sources of information provided in the curriculum.

Independent work introduces students to scientific creativity, searching for and solving current modern problems.

2. Objectives and main tasks SRS

The primary goal of organizing and implementing independent work should align with the goal of student education—the preparation of a bachelor's degree holder with a higher education degree. When organizing independent work, developing the ability to work independently to acquire knowledge, skills, and the ability to organize educational and research activities is an important and necessary condition.

The goal of students' independent work is to acquire fundamental knowledge, professional skills, and relevant professional abilities, as well as experience in creative and research activities. Independent work promotes the development of students' independence, responsibility, self-discipline, and a creative approach to solving academic and professional problems.

The objectives of the SRS are:

- systematization and consolidation of the acquired theoretical knowledge and practical skills of students;
- deepening and expansion of theoretical knowledge;
- development of skills in using regulatory, legal, reference documentation and special literature;
- development of students' cognitive abilities and activity: creative initiative, independence, responsibility and organization;
- the development of independent thinking, abilities for self-development, self-improvement and self-realization;
- development of research skills;
- use of material collected and obtained during independent study at seminars, in laboratory classes, when writing coursework and final qualifying papers, for effective preparation for final exams and exams.

3. Types of independent work

In the educational process of higher professional educational institutions, two types of independent work are distinguished: in-class, under the guidance of a teacher, and out-of-class. The close interrelationship between these types of work provides for the differentiation and effectiveness of its results and depends on the organization, content, and logic of the educational process (interdisciplinary connections, advanced knowledge, and etc.):

Independent classroom work on the subject is carried out during classes under the direct supervision of the teacher and according to his/her instructions.

Extracurricular independent work is performed by a student on the teacher's instructions, but without his or her direct participation.

The main types of independent work of students without the participation of a teacher in this discipline are:

- formation and assimilation of the content of lecture notes based on the educational literature recommended by the lecturer, including information educational resources (electronic textbooks, electronic libraries, etc.) etc.);
- preparation for laboratory or practical work, their design.

The main types of independent work of students with the participation of teachers are:

- current consultations;
- acceptance and defense of laboratory work (during the course) l/r, if they are provided for by the curriculum);
- completing assignments during practical classes (if such are provided for in the curriculum of the discipline);
- completion of a coursework or project, if provided for in the curriculum.

4. Organization SRS

The methodology for organizing students' independent work depends on the structure, nature, and characteristics of the "Python Program Development" course, the number of hours for its study, the type of assignments for students' independent work, the individual qualities of students, and the conditions of educational activities.

The process of organizing students' independent work includes the following stages:

- preparatory (defining goals, drawing up a program, preparing methodological support, preparation equipment);
- main (implementation of the program, use of techniques for searching for information, assimilation, processing, application, transfer of knowledge, recording of results, self-organization of the process work);
- final (assessment of significance and analysis of results, their systematization, assessment of the effectiveness of the program and work methods, conclusions on areas of optimization labor).

Students' activities to form and develop skills for independent study

In the process of independent work, the student acquires the skills of self-organization, self-control, self-management, self-reflection and becomes an active,

independent subject of educational activity.

When completing independent work under the supervision of a teacher, a student must:

- master the minimum content assigned to students for independent work and proposed by the teacher in accordance with the Federal State Educational Standards of Higher Education;
- plan independent work in accordance with the independent work schedule proposed by the teacher;
- The student must carry out independent work in the organizational forms provided for by the curriculum and the teacher's work program;
- perform independent work and report on its results in accordance with the schedule for presenting results, types and deadlines for reporting on independent work students.

a student can:

in addition to the minimum required content proposed by the teacher (subject to justification and agreement with him), determined by the program for this discipline:

- independently determine the level (depth) of elaboration of the content of the material;
- suggest topics and questions for independent study;
- within the general schedule for completing independent work, propose a reasonable individual schedule for completing and reporting on the results of independent work;
- propose your own options for organizational forms of independent work;
- use for independent work methodological aids, teaching aids, and developments in addition to the list proposed by the teacher;
- use not only control, but also self-control of the results of independent work in accordance with the methods of self-control proposed by the teacher or chosen independently.

Students' independent work should have a significant impact on the development of their future professional identity; it is planned by the student themselves. Each student determines their own work schedule and the amount of effort expended on mastering the academic content in each subject. They complete their extracurricular work according to their own individual plan, depending on their preparation, time, and other circumstances.

5. General recommendations for organizing independent work

The primary form of student independent work involves studying lecture notes,

supplementing them, referring to recommended literature, and actively participating in labs or practical classes (depending on the curriculum). However, to ensure successful learning and its intensification, the following subjective factors must be considered:

1. Knowledge of the school curriculum and a solid system of gaps necessary for mastering core university courses are essential. This is especially important for mathematics and engineering disciplines. It is essential to distinguish between gaps in knowledge that hinder the assimilation of new material and limited abilities. By investing effort in overcoming these gaps, a student will ensure adequate academic performance and gain confidence in their abilities. capabilities.

2. The presence of mental skills and abilities labor:

a) the ability to take notes during lectures and when working with a book;

b) mastery of logical operations: comparison, analysis, synthesis, generalization, definition of concepts, rules of systematization and classification.

3. The specifics of cognitive mental processes: attention, memory, speech, observation, intellect, and thinking. Poor development of each of these becomes a serious obstacle in studies.

4. Good performance, which is ensured by normal physical condition. After all, serious learning is a great, multifaceted and varied effort. The result of learning is assessed not by the quantity of information imparted, but by the quality of its assimilation, the ability to use it, and the development of one's ability for further independent education.

5. Matching your chosen activity and profession to your individual abilities. It is necessary to develop the ability to regulate your emotional state and eliminate circumstances that disrupt your business spirit and interfere with your intended work.

6. Mastering an optimal work style that ensures success in your work. Alternating work and breaks, rest periods, individually tailored sleep patterns, preference for evening or morning classes, stress tolerance during exams, and specific preparation techniques.

7. The level of self-esteem is determined by the established self-esteem. An adequate assessment of knowledge, strengths, and weaknesses is important. a component of human self-organization; without it, successful work on managing one's behavior and activities is impossible.

One of the main features of higher education is that constant external control is replaced by self-control; the active role in learning belongs not so much to the teacher as to the student.

Knowing the basic methods of scientific organization of mental work, one can achieve the best results with the least expenditure of time, resources and labor effort.

The effectiveness of assimilation of incoming information depends on a person's performance at any given moment of his activity.

Work capacity is a person's ability to work at a high level of intensity for a certain period of time. A distinction is made between internal and external factors of work capacity.

Internal factors of performance include intellectual abilities, willpower, and health status.

To external:

- organization of the workplace, work schedule and rest;
- level of work organization - the ability to obtain information and use it;
- the magnitude of mental loads.

The outstanding Russian physiologist N. E. Vvedensky identified the following conditions for the productivity of mental activity:

- you have to enter into every work gradually;
- the regularity and rhythm of work. Different people have more or less different tempos work;
- habitual consistency and systematicity activities;
- correct alternation of labor and rest.

Rest doesn't necessarily require complete inactivity; it can be achieved by simply changing tasks. Work capacity fluctuates throughout the day. The most productive *time is morning (8:00 a.m. to 2:00 p.m.)*, with peak productivity occurring between 10:00 a.m. and 1:00 p.m., followed by *afternoon (4:00 p.m. to 7:00 p.m.)* and *evening (8:00 p.m. to midnight)*. It's best to study very difficult material at the beginning of each work period (preferably morning) after a good rest. Breaks of 10-15 minutes are needed after 1-1.5 hours of work, and after 3-4 hours of work, rest should be longer – about an hour.

An integral part of the scientific organization of mental work is the mastery of the technique of mental work.

The time available for students to complete their curriculum consists of two components: one is scheduled classroom work at the university, and the other is independent work outside of class. Assignments and materials for independent work are provided during scheduled classes. During these classes, the instructor monitors independent work and assists students in organizing their work properly.

Independent study requires intense mental effort, which must not only be properly organized but also stimulated. It's crucial to be able to maintain sustained attention to the material being studied. Developing attention requires considerable willpower. Therefore, if a student notices they are frequently distracted during independent study, they should force themselves to focus. This process should be repeated regularly, as it trains their attention. Sustained attention develops when a person approaches the subject with interest.

You should organize your activities properly by time: 50 minutes of work, 5-10

minutes of break; after 3 hours of work, a 20-25 minute break. Otherwise, increasing fatigue will lead to instability of attention. A significant factor influencing the improvement of mental performance is regular physical exercise. Active recreation involves alternating mental and physical activity, which fully restores a person's performance.

6. Independent work of a student is a necessary link formation researcher

The progress of science and technology, information technology leads to a significant increase in scientific information, which places higher demands not only on the moral and ethical qualities of a person, but also, in particular, constantly increasing demands in the field of education - updating, modernization of general and professional knowledge, skills of a specialist.

All education should be a dynamic process, inherent to the individual and continuing throughout their life. Mastering scientific thought and the language of science is a necessary component in the self-organization of a future research specialist. This implies not so much the accumulation of knowledge as the mastery of scientifically sound methods for acquiring it. This, generally speaking, is the primary task of a university.

The specific nature of the university educational process, in which independent student work is increasingly emphasized, is that it seems to be the final and most appropriate link in achieving this goal. During university studies, a style and skills of learning (cognitive) activity are developed, the rational nature of which will facilitate the continuous renewal of knowledge in highly qualified university graduates.

However, this path faces certain challenges, particularly the student's transition from the synthetic learning process in secondary school to the analytical one in higher education. This is due both to the new content of instruction (the expansion of general education and the deepening of professional training) and to new, previously unknown forms of learning (lectures, seminars, laboratory classes, etc.). Students acquire not only the knowledge provided by the curriculum and textbooks but also must become familiar with the methods of acquiring knowledge so as to be able to evaluate what we know, where we know it from, and how we acquired it. All of this is achieved through independent work.

This is because independently acquired knowledge is more immediate, becoming personal property and a motivation for behavior, developing intellectual traits, attention, observation, critical thinking, and judgment. The instructor's role primarily consists of guiding the accumulation of knowledge (for first-year students), and in subsequent years, during senior years, collaboratively identifying problems and supporting students' independent explorations, as well as monitoring their activities. It should be noted that

one should not limit oneself to acquiring the knowledge outlined in the curriculum; one must continually deepen one's acquired knowledge, focusing it on a narrow, specific area that corresponds to the student's interests. In-depth study of all subjects covered by the curriculum is possible in practice, and good organization of work saves time, creating the conditions for a deep, systematic, and engaging study of the student's chosen topic.

Of course, all the advice, examples, and recommendations in this area, whether given by a teacher, certain publications, or other sources, do not guarantee any success without personal initiative. They do not provide ready-made recipes, but rather should encourage an analysis of one's own work, its goals, and its organization in accordance with individual characteristics. Taking into account personal capabilities, existing living and working conditions, and skills, based on these recommendations, it is possible to develop an individually valid set of methods and techniques, find one's own style, or refine it. After studying specific material, one has the time to evaluate its significance, suitability, and potential for its application, ultimately ensuring the success of one's studies and future professional activities.

7. Methodological recommendations for students on individual forms of independent work

The university education system implies significantly greater independence for students in planning and organizing their activities.

Working with literature

When working with literature, it's important to select the literature, learn to read it correctly, and keep records. To select literature, the library uses alphabetical and systematic catalogs.

It is important to remember that rational skills in working with a book always mean great savings of time and effort.

The instructor teaching the course recommends the appropriate selection of textbooks. Required literature may also be listed in the course's teaching materials.

When studying the material in a textbook, you should move on to the next question only after correctly understanding the previous one, describing on paper all the calculations and computations (including those that are omitted from the textbook or given in the lecture for independent derivation).

When studying any discipline, independent individual work plays a large and important role.

Particular attention should be paid to defining the course's key concepts. Students

should thoroughly analyze examples that illustrate these definitions and be able to construct similar examples independently. It's important to develop a clear understanding of what you're learning. It's helpful to take supporting notes. When studying the material in the textbook, it's helpful to supplement your lecture notes in the designated margins in your notebook. You should also note any questions you've identified for consultation with your instructor. teacher.

It is recommended to highlight the conclusions obtained as a result of the study in the notes so that they are better remembered when rereading the notes.

Experience shows that many students are helped by Compiling a keynote list containing the most important and frequently used formulas and concepts. This list helps students remember formulas and key points of the lecture, and can also serve as a constant reference. There are two types of reading: primary and secondary. *Primary* reading is attentive, unhurried reading, allowing you to pause at difficult passages. Afterward, there shouldn't be a single unclear point. The content may not always be clear after the primary reading.

The goal of *secondary* reading is to fully grasp the meaning of the whole (this reading may not be the second, but the third or fourth).

Reading a scientific text is part of cognitive activity. Its purpose is to extract the necessary information from the text. The effectiveness of the activity largely depends on how consciously the reader is guided by his or her own internal attitude when accessing the printed word (to find the necessary information, to assimilate the information in whole or in part, to critically analyze the material, etc.). actions.

There are four main attitudes towards reading scientific text:

1. information retrieval (the task is to find and highlight the required information);
2. assimilating (the reader's efforts are aimed at understanding and remembering as fully as possible both the information presented by the author and the entire logic of his reasoning);
3. analytical-critical (the reader strives to critically comprehend the material, having analyzed it, determining his attitude towards it) him);
4. creative (creates in the reader a readiness in one form or another - as a starting point for his reasoning, as an image for action by analogy, etc. - to use the author's judgments, his train of thought, the result of observation, the developed methodology, to supplement them, to subject them to a new verification).

types of reading is associated with the presence of various settings for accessing scientific text :

1. bibliographic – viewing catalog cards, recommended lists, summary lists of journals and articles for the year and etc.;

2. browsing - used to search for materials containing the required information, usually resorted to immediately after working with bibliography and catalogs, as a result of such browsing the reader determines which sources will be used in the future work;

3. introductory - implies a continuous, fairly detailed reading of selected articles, chapters, individual pages, the goal is to become familiar with the nature of the information, find out what issues the author has raised for consideration, and sort it out material;

4. Studying – involves a thorough mastery of the material; during such reading, the reader demonstrates trust in the author, a willingness to accept the information presented, and a focus on the most complete understanding is realized. material;

5. Analytical-critical and creative reading are two types of reading that are related to each other in that they both contribute to solving research problems. The first involves a focused critical analysis of both the information itself and the author's methods of obtaining and presenting it; the second involves searching for those judgments and facts about which, or in connection with which, the reader considers it necessary to express their own thoughts.

Of all the types of reading discussed, exploratory reading is the most important for students—it allows them to accumulate knowledge in various fields while working with academic literature. Therefore, this type of reading should be mastered first in the learning process. Furthermore, mastering this type of reading helps develop key techniques that improve the effectiveness of working with scientific texts.

The main types of systematic recording of what has been read:

1. Annotation is an extremely brief, coherent description of a book (article) viewed or read, its content, sources, nature and purpose;

2. Planning is a brief logical organization of the text, revealing the content and structure of the subject being studied. material;

3. Thesis is a concise reproduction of the author's main assertions without resorting to factual material;

4. Quoting is the verbatim copying of excerpts and extracts from the text that most significantly reflect a particular idea. author;

5. Note-taking is a brief and consistent presentation of the content read.

A summary is a complex way of presenting the contents of a book or article in a logical sequence. A summary integrates previous notes and allows for a comprehensive understanding of the book or article's content. Therefore, the ability to create an outline, make notes, take excerpts, and other notes determines the technique of summary writing.

Guidelines for writing notes:

1. Read the text carefully. Look up any unfamiliar words in reference books.

When taking notes, be sure to include reference information in the margins.

2. Highlight the main points, make a summary plan.

3. Briefly formulate the main points of the text, highlight the argumentation author.

4. Take notes on the material, strictly following the points of the plan. When taking notes, try to express your thoughts in your own words. Notes should be kept clearly, It's clear.

5. Write quotes correctly. When quoting, consider brevity and significance. thoughts.

In the summary, it is advisable to present not only thesis statements but also their evidence. When formatting the summary, strive for the conciseness of each sentence. The author's thoughts should be presented concisely, with care taken to maintain style and expressiveness. The number of additional elements in the summary should be logically justified, and entries should be arranged in a specific sequence consistent with the logical structure of the work. Leave margins for clarification and addition.

Mastering note-taking skills requires dedication and independent work on a daily basis from the student.

Laboratory/Practical classes (depending on the curriculum)

It should be emphasized that only after the lecture material has been mastered from a specific perspective (namely, the one from which it is presented in lectures) will it be reinforced in laboratory or practical classes, both through discussion and analysis of the lecture material and through completing laboratory or practical assignments and assigned problems. Under these conditions, the student will not only master the material well but also learn to apply it in practice, and will also receive additional incentive (and this is very important) to actively engage with the lecture.

When completing assignments independently, each step of the solution should be justified based on the theoretical principles of the course. If the student sees several possible solutions to a problem (task), they should be compared and the most efficient one selected. It is helpful to create a brief plan for solving the problem (task) before beginning calculations. Solutions to problems or examples should be presented in detail, with calculations arranged in strict order, separating auxiliary calculations from the main ones. Solutions should be accompanied by comments, diagrams, drawings, and illustrations, if necessary.

It's important to remember that each learning task should be completed until the final logical answer required by the conditions is reached, and, if possible, with a conclusion. This type of task should be continued until the student develops solid skills in solving them.

Self-check

After studying a specific topic from notes in the notes and textbook, as well as solving a sufficient number of relevant tasks in laboratory classes and independently, the student is recommended, using a sheet of supporting materials, to reproduce from memory the definitions, derivations of formulas, formulations of the main provisions and proofs.

If necessary, you need to carefully review the material again.

Sometimes, a student's lack of understanding of a particular topic only becomes apparent upon further study. In this case, it's necessary to go back and review the poorly understood material. An important criterion for mastering theoretical material is the ability to solve problems or pass tests on the material covered. However, it's important to remember that a correct solution to a problem can result from applying rote memorized formulas without understanding the essence of the theoretical principles.

Consultations

If, while independently studying theoretical material or completing assignments, a student encounters questions that they cannot resolve independently, they should contact the instructor for clarification or guidance. In their questions, students should clearly explain the difficulty they are experiencing and the nature of the difficulty. Consultation should also be sought if they have any doubts about the correctness of their answers to self-assessment questions.

Preparing for the test in the course "Developing Python Programs"

The study of many general and specialized disciplines culminates in a test. Preparing for a graded test helps consolidate, deepen, and generalize the knowledge acquired during the course of study, as well as apply it to solving practical problems. By preparing for this test, students address existing knowledge gaps, deepen, systematize, and organize their knowledge.

To successfully pass the test, it is important for the student to fully complete the assignments of laboratory or practical work during the semester, prepare a report on each work and a report on the results of independent study of the topics, according to the curriculum.

Systematic preparation for classes throughout the semester will allow you to use the time during the exam session to systematize your knowledge.

Current assessment of students is carried out by teachers conducting practical classes in the discipline in the following forms: an interview during the defense of practical work assignments with the provision of a written (or electronic) report, depending on the requirements.

8. Educational, methodological and informational support disciplines

Main literature	1. Baldzhi, A. S.; Mathematics in Python: a tutorial / A.S. Baldzhi, M.B. Khripunova, I.A. Aleksandrova; Financial University under the Government of the Russian Federation, 1, Elements of linear algebra and analytic geometry. - Moscow: Prometheus, 2018. - 76 p.: table. - http://biblioclub.ru/. - Bibliography in the book. - ISBN 978-5-907003-86-6 2. Borzunov, S. V.; Algebra and Geometry with Examples in Python Electronic resource / Borzunov S. V., Kurgalin S. D.: a textbook for universities. - St. Petersburg: Lan, 2020. - 444 p. - ISBN 978-5-8114-5489-1 3. Shkaberina, G. Sh.; Programming. Basics of the Python Language Electronic resource / Shkaberina G. Sh., Rezova N. L.: study guide. - Krasnoyarsk: Siberian State University named after Academician M. F. Reshetnev, 2018. - 92 p. - APPROVED BY the Editorial and Publishing Council of the University as a study guide for undergraduate students in the areas of training 09.03.01 "Computer Science and Computer Engineering", 09.03.02 "Information Systems and Technology", 09.03.04 "Software Engineering" of all forms of education
Further reading	1. Suzi, R.A; The Python Programming Language. Electronic resource: a tutorial / R.A. Suzi. - The Python Programming Language, 2020-07-28. - Moscow: Internet University of Information Technologies (INTUIT), 2016. - 350 p. - The book is in the basic version of the IPRbooks electronic library system. - ISBN 5-9556-0058-2 2. Sheludko, V. M; Fundamentals of Programming in the High-Level Language Python Electronic resource: Textbook / V. M. Sheludko. - Rostov-on-Don, Taganrog: Publishing House of the Southern Federal University, 2017. - 146 p. - The book is in the premium version of the IPR BOOKS electronic library system. - ISBN 978-5-9275-2649-9 3. Sheludko, V. M; The High-Level Programming Language Python. Functions, Data Structures, Additional Modules Electronic resource: Textbook / V. M. Sheludko. - Rostov-on-Don, Taganrog: Publishing House of the Southern Federal University, 2017. - 107 p. - The book is in the premium version of the IPR BOOKS electronic library system. - ISBN 978-5-9275-2648-2

Software	Alt Workstation 10 Alt Workstation K Alt "Server" Office suite - P7-Office