МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Методические указания по выполнению практических работ по дисциплине Основы анализа данных в Python

Направление подготовки	42.03.02 – «Журналистика»
Направленность (профиль)	Мультимедийная журналистика и цифровые технологии

Ставрополь, 2026

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель освоения дисциплины «Основы анализа данных в Python» - формирование у студентов, обучающихся по направлению подготовки 42.03.02 Журналистика, базовых теоретических знаний и практических профессиональных компетенций в области программирования на языке Python, достаточных для понимания принципов работы с данными, выполнения основных операций по сбору, обработке и анализу простых наборов данных, а также применения полученных компетенций для решения задач, связанных с анализом информации в профессиональной журналистской деятельности.

Задачи освоения дисциплины « Основы анализа данных в Python»:

1. Изучить базовый синтаксис языка Python, основные типы данных, операторы и управляющие конструкции (условия, циклы), необходимые для написания простых программ и скриптов для обработки данных.
2. Освоить основные встроенные структуры данных Python (строки, списки, словари, кортежи, множества), их свойства и методы работы, ориентированные на хранение, организацию и первичную обработку различных типов данных.
3. Научиться работать с файлами в Python для чтения, записи и обработки данных из внешних источников (текстовые файлы, простые форматы данных), а также применять базовые алгоритмы для анализа текстовой информации и простых наборов данных (поиск, подсчет частоты, сортировка).
4. Развить навыки алгоритмического мышления и отладки программ, а также получить представление о выборе подходящих структур данных и методов для эффективного решения задач по анализу и обработке информации в контексте журналистских расследований и создания медиапродуктов.

В результате усвоения тем и разделов дисциплины «Основы анализа данных в Python», работы на практических занятиях у обучающихся должны сформироваться следующие компетенции:

Индекс	**Формулировка:**
ПК-3	Способен участвовать в производственном процессе выпуска журналистского текста и (или) продукта с применением современных цифровых технологий, специализированного программного обеспечения

ПЛАНЫ ПРАКТИЧЕСКИХ ЗАНЯТИЙ

Практическое занятие 1

Тема: Введение в Python для анализа данных

Цель занятия: ознакомиться с языком программирования Python, его основными возможностями для анализа данных и подготовить рабочее окружение.

Формируемые компетенции: ПК-3

Актуальность: Python является одним из самых востребованных инструментов для работы с данными в современном мире. Освоение его основ критически важно для журналистов, стремящихся проводить собственные расследования на базе данных и создавать аналитический контент.

Теоретическая часть.

Современная журналистика все больше опирается на данные. Умение собирать, обрабатывать, анализировать и визуализировать данные открывает новые возможности для поиска тем, проверки гипотез и представления информации аудитории. В этом контексте язык программирования Python становится мощным инструментом в арсенале журналиста. Python зарекомендовал себя как один из самых популярных языков благодаря своему простому синтаксису и широким возможностям.

Что такое Python и почему он важен для анализа данных? Python — это универсальный язык программирования высокого уровня, который отличает высокая читаемость кода и относительно низкий порог вхождения. Он широко используется в веб-разработке, автоматизации, научных исследованиях и, что особенно важно для нас, в анализе данных. По сравнению с другими языками, Python более "человекочитаем", в нем меньше двусмысленностей и исключений, что облегчает его изучение, особенно для тех, кто не имеет обширного опыта в программировании. Для журналистов, которые часто работают в условиях жестких дедлайнов, простота и скорость разработки на Python являются огромным преимуществом.

Преимущества Python для журналистов и аналитиков:

1. Простота и читаемость: Синтаксис Python интуитивно понятен, что позволяет быстро освоить основы и сосредоточиться на логике анализа данных, а не на сложностях языка.
2. Обширные библиотеки: Python располагает огромным количеством готовых библиотек и фреймворков, разработанных специально для анализа и обработки данных (например, Pandas, NumPy, SciPy), визуализации (Matplotlib, Seaborn, Plotly), веб-скрапинга (BeautifulSoup, Scrapy), работы с текстом (NLTK, SpaCy), геоданными (GeoPandas, Folium) и даже сбора данных из социальных сетей через API. Это означает, что большинство задач уже имеет готовые решения, которые нужно лишь научиться использовать.
3. Сообщество и ресурсы: Вокруг Python сформировалось огромное и дружелюбное сообщество разработчиков. Существует огромное количество учебных материалов, форумов, онлайн-курсов и готовых примеров кода, что значительно упрощает процесс обучения и решения возникающих проблем.
4. Универсальность: Python можно использовать для решения самых разных задач, возникающих в дата-журналистике: от автоматизированного сбора данных с сайтов и из социальных сетей до сложного статистического анализа и создания интерактивных визуализаций для публикации.

Установка и настройка Python для работы с данными:
Для начала работы с Python необходима его установка на компьютер. Рекомендуется использовать дистрибутив Anaconda, который включает сам Python и множество предустановленных библиотек для анализа данных, что избавляет от необходимости

устанавливать их по отдельности. Процесс установки достаточно стандартен для большинства операционных систем (Windows, macOS, Linux). После установки Python и необходимых библиотек важно выбрать удобную среду разработки (IDE - Integrated Development Environment) или интерактивную оболочку для написания и выполнения кода. Популярные варианты включают:

- Jupyter Notebook/JupyterLab: Идеально подходят для анализа данных. Они позволяют писать и выполнять код по блокам, комбинировать код с текстом, формулами и визуализациями, что удобно для проведения исследований и документирования процесса анализа.
- VS Code (Visual Studio Code): Популярный и многофункциональный редактор кода с широкими возможностями для разработки на Python, включая отладку, интеграцию с системами контроля версий и поддержку расширений для работы с данными.
- PyCharm: Полноценная IDE, предлагающая мощные инструменты для разработки, особенно для крупных проектов.

Настройка среды включает установку необходимых библиотек (Pandas, NumPy, Matplotlib и др.), если они не были включены в дистрибутив по умолчанию, и освоение базовых функций выбранного инструмента (создание файлов/ноутбуков, выполнение кода). Первый шаг к освоению Python и анализу данных – это запуск первой программы, даже самой простой, чтобы увидеть, как работает интерпретатор.

Вопросы для обсуждения:

1. Обзор языка Python и его применения в анализе данных.
2. Преимущества Python для журналистов и аналитиков.
3. Инструкции по установке Python и необходимых библиотек (Pandas, NumPy, Matplotlib).
4. Настройка среды разработки (Jupyter Notebook, VS Code).

Список литературы, рекомендуемый к использованию по данной теме

Основная литература:

1. Аллен Дауни Основы Python. Научитесь думать как программист / Аллен Б. Дауни ; пер. с англ. С. Черникова ; [науч. ред. А. Родионов]. — Москва : Манн, Иванов и Фербер, 2021. — 304 с.

Дополнительная литература:

1. Бэрри Пол Изучаем программирование на Python. -М., 2022.
2. Зингаро Даниэль Python без проблем. М., 2023.

Интернет-ресурсы:

1. http://www.biblioclub.ru – Университетская библиотека онлайн.
2. http://elibrary.ru - Научная электронная библиотека.
3. http://cyberleninka.ru/ - Научная электронная библиотека.
4. IPR SMART - Цифровой образовательный ресурс

Практическое занятие 2

Тема: Основы программирования на Python

Цель занятия: изучить базовые конструкции языка Python: переменные, типы данных, операторы, функции.

Формируемые компетенции: ПК-3

Актуальность: Владение базовым синтаксисом Python является фундаментом для написания любого кода, в том числе для задач анализа данных.

Теоретическая часть.

Прежде чем перейти к специализированным библиотекам для анализа данных, необходимо освоить базовые строительные блоки языка Python. Эти основы включают понимание переменных, типов данных, операторов и функций.

Первые шаги в Python: синтаксис и базовые конструкции. Основы синтаксиса Python отличаются лаконичностью и использованием отступов для определения структуры кода (блоков). В отличие от многих других языков, где используются фигурные скобки или ключевые слова like begin/end, Python полагается на отступы, что делает код визуально более чистым и читаемым.

- Переменные: Переменные используются для хранения значений в памяти компьютера. В Python переменной можно присвоить любое значение, используя оператор присваивания =. Имя переменной может состоять из букв, цифр и символа подчеркивания, но не может начинаться с цифры и является регистрозависимым. Например: count = 10, user_name = "Alice".
- Типы данных: Python имеет несколько базовых встроенных типов данных (см. Таблицу 2.1 в "Простом Python"). Наиболее часто используемые в анализе данных:
  - Числа: Целые (int) и числа с плавающей точкой (float). Например: age = 30, price = 199.99.
  - Строки (str): Последовательности символов, заключенные в кавычки (одинарные или двойные). Используются для работы с текстом. Например: headline = "Новость дня".
  - Булевы значения (bool): Могут принимать только два значения: True (истина) или False (ложь). Часто используются в условных операторах.
- Операторы: Используются для выполнения операций над данными. Арифметические операторы (+, -, *, /, ** возведение в степень), операторы сравнения (==, != , <, >, <=, >=), логические операторы (and, or, not). Примеры простых программ для выполнения арифметических операций могут выглядеть так:

```
a = 15
b = 7
sum_result = a + b
difference = a - b
product = a * b
division = a / b # Результат будет float
print(f"Сумма: {sum_result}")
print(f"Разность: {difference}")
print(f"Произведение: {product}")
print(f"Частное: {division}")
```

content_copydownload

Use code with caution.Python

Функции в Python: создание и использование. Функции позволяют группировать блоки кода, выполняющие определенную задачу, и вызывать этот код по имени. Это делает программу более организованной, избегает повторения кода и упрощает его отладку.

- Определение функций: Функции определяются с помощью ключевого слова def, за которым следует имя функции, круглые скобки для параметров и двоеточие. Тело функции отделяется отступом.

```
def greet(name):
    print(f"Привет, {name}!")
```

content_copydownload

Use code with caution.Python

- Параметры и аргументы: Параметры — это имена, указанные в определении функции (в скобках). Аргументы — это фактические значения, которые передаются функции при ее вызове.
- Использование функций: Функция вызывается по ее имени с указанием аргументов в скобках.

greet("Мир") # Выведет "Привет, Мир!"
content_copydownload
Use code with caution.Python

- Примеры функций для обработки данных: В анализе данных функции могут использоваться для выполнения типовых операций, например, вычисления статистик (среднее, медиана), форматирования данных, проверки условий. Например:
- def calculate_average(numbers):
- """Вычисляет среднее арифметическое списка чисел."""
- if not numbers: # Проверяем, не пустой ли список
- return 0
- total = sum(numbers) # sum() - встроенная функция
- return total / len(numbers) # len() - встроенная функция, возвращает длину списка
-
- data = [10, 15, 20, 25, 30]
- average = calculate_average(data)

print(f"Среднее значение: {average}") # Выведет "Среднее значение: 20.0"
content_copydownload
Use code with caution.Python

Понимание этих базовых элементов синтаксиса и структур является необходимым условием для дальнейшего изучения более сложных тем, связанных с анализом данных с использованием специализированных библиотек.

Вопросы для обсуждения:

1. Основы синтаксиса Python: переменные, типы данных, операторы.
2. Примеры простых программ для выполнения арифметических операций.
3. Определение функций, их параметры и аргументы.
4. Примеры функций для обработки данных (например, вычисление статистик).

Список литературы, рекомендуемый к использованию по данной теме

Основная литература:

1. Аллен Дауни Основы Python. Научитесь думать как программист / Аллен Б. Дауни ; пер. с англ. С. Черникова ; [науч. ред. А. Родионов]. — Москва : Манн, Иванов и Фербер, 2021. — 304 с.

Дополнительная литература:

1. Бэрри Пол Изучаем программирование на Python. -М., 2022.
2. Зингаро Даниэль Python без проблем. М., 2023.
3. Интернет-ресурсы:

1. http://www.biblioclub.ru – Университетская библиотека онлайн.
2. http://elibrary.ru - Научная электронная библиотека.
3. http://cyberleninka.ru/ - Научная электронная библиотека.
4. IPR SMART - Цифровой образовательный ресурс

Практическое занятие 3

Тема: Работа с данными в Python

Цель занятия: изучить основные типы и структуры данных Python, используемые для хранения и организации информации, а также методы работы с файлами.

Формируемые компетенции: ПК-3

Актуальность: Умение эффективно работать с различными типами данных и считывать/записывать информацию из файлов и в файлы является базовым навыком для любого аналитика данных.

Теоретическая часть.

Данные, с которыми приходится работать журналисту, могут иметь самую разную структуру: от простых списков имен до сложных таблиц со статистикой или текстовых документов. Python предлагает набор встроенных типов и структур данных, которые позволяют удобно хранить и манипулировать информацией. Понимание этих структур критически важно.

Типы данных и структуры данных. Помимо базовых "атомарных" типов вроде чисел и строк, Python предоставляет более сложные "коллекции", способные хранить наборы данных. Наиболее важные для анализа:

- Списки (list): Упорядоченные, изменяемые коллекции элементов. Элементы в списке могут иметь разные типы и доступны по индексу (начиная с 0). Списки создаются с помощью квадратных скобок [].
- news_categories = ["Политика", "Экономика", "Культура", "Спорт"]

article_info = ["Заголовок статьи", 2500, True] # Список может содержать разные типы

content_copydownload

Use code with caution.Python

Списки поддерживают множество операций: добавление/удаление элементов, изменение по индексу, срезы (получение подсписка), сортировка.

- Кортежи (tuple): Упорядоченные, *неизменяемые* коллекции элементов. Похожи на списки, но после создания их нельзя изменить. Создаются с помощью круглых скобок ().

geographic_coords = (55.7558, 37.6173) # Координаты неизменны

content_copydownload

Use code with caution.Python

Кортежи используются там, где требуется гарантировать неизменность данных.

- Словари (dict): Неупорядоченные (в старых версиях Python), изменяемые коллекции пар "ключ-значение". Каждому уникальному ключу соответствует одно значение. Ключи должны быть неизменяемыми объектами (строки, числа, кортежи), значения могут быть любого типа. Словари создаются с помощью фигурных скобок {}.
- article_metadata = {
- "title": "Анализ данных в журналистике",
- "author": "Иванов И.И.",
- "date": "2023-10-27",
- "views": 1500

}

content_copydownload

Use code with caution.Python

Словари идеально подходят для хранения структурированных данных, где доступ к элементам осуществляется по осмысленным ключам, а не по числовому индексу.

- Множества (set): Неупорядоченные коллекции *уникальных* элементов. Используются для хранения уникальных значений и выполнения операций над множествами (объединение, пересечение).

Примеры работы с текстовыми данными и числовыми данными: Базовые типы данных и структуры позволяют выполнять простые операции:

- Работа со строками: объединение (+), повторение (*), получение длины (len()), доступ по индексу, срезы, поиск подстроки, замена символов (replace()), разделение на список по разделителю (split()), удаление пробелов (strip()).
- Работа с числами: арифметические операции, сравнения, встроенные функции (sum(), min(), max()).

Чтение и запись данных из файлов. Большинство данных для анализа хранится в файлах. Умение работать с файлами — базовый навык. Python предоставляет встроенные функции для работы с файлами (open(), read(), write(), close()) и удобный способ управления файлами с помощью контекстного менеджера (with open(...)).

- Работа с текстовыми файлами (CSV, JSON). Наиболее распространенные форматы для обмена данными. CSV (Comma Separated Values) — простой формат для табличных данных, где значения разделены запятыми (или другими разделителями). JSON (JavaScript Object Notation) — формат для структурированных данных, основанный на парах "ключ-значение" и списках, очень похож на словари и списки Python. Для работы с этими форматами в Python есть встроенные модули csv и json, а также мощные внешние библиотеки вроде Pandas.

```
# Пример чтения из текстового файла
with open('data.txt', 'r', encoding='utf-8') as f:
    content = f.read()
    print(content)

# Пример записи в текстовый файл
data_to_write = "Новая строка данных\n"
with open('output.txt', 'w', encoding='utf-8') as f:
    f.write(data_to_write)

# Пример работы с JSON (требует импорта модуля json)
import json
data_dict = {"name": "Article 1", "views": 2000}
with open('data.json', 'w', encoding='utf-8') as f:
    json.dump(data_dict, f, ensure_ascii=False, indent=4) # Запись словаря в JSON файл

with open('data.json', 'r', encoding='utf-8') as f:
    loaded_data = json.load(f) # Чтение JSON файла в словарь Python
print(loaded_data["views"])
```

content_copydownload

Use code with caution.Python

Примеры загрузки данных из файлов для дальнейшего анализа: Для журналистских задач часто требуется загрузить данные из CSV-файла (например, статистика по преступлениям, демографические данные), JSON-файла (например, выгрузка из API) или простого текстового файла (например, сборщик новостей). Базовые методы работы с файлами в Python позволяют это сделать, хотя для более сложных случаев и больших объемов данных используются специализированные библиотеки, такие как Pandas (рассматривается в Теме 4), которая значительно упрощает процесс чтения табличных данных.

Вопросы для обсуждения:

1. Типы данных и структуры данных.
2. Строки, списки, словари, кортежи и их применение в анализе данных.
3. Примеры работы с текстовыми данными и числовыми данными.
4. Чтение и запись данных из файлов.
5. Работа с текстовыми файлами (CSV, JSON).
6. Примеры загрузки данных из файлов для дальнейшего анализа.

Список литературы, рекомендуемый к использованию по данной теме

Основная литература:

1. Аллен Дауни Основы Python. Научитесь думать как программист / Аллен Б. Дауни ; пер. с англ. С. Черникова ; [науч. ред. А. Родионов]. — Москва : Манн, Иванов и Фербер, 2021. — 304 с.

Дополнительная литература:

1. Бэрри Пол Изучаем программирование на Python. -М., 2022.
2. Зингаро Даниэль Python без проблем. М., 2023.
3. Интернет-ресурсы:

1. http://www.biblioclub.ru – Университетская библиотека онлайн.
2. http://elibrary.ru - Научная электронная библиотека.
3. http://cyberleninka.ru/ - Научная электронная библиотека.
4. IPR SMART - Цифровой образовательный ресурс

Практическое занятие 4

Тема: Анализ данных с помощью Pandas

Цель занятия: освоить основы работы с библиотекой Pandas для загрузки, очистки, преобразования и базового анализа структурированных данных.

Формируемые компетенции: ПК-3

Актуальность: Pandas – это де-факто стандарт в Python для работы с табличными данными. Умение использовать эту библиотеку критически важно для анализа любых структурированных наборов данных в журналистских расследованиях.

Теоретическая часть.

Большинство данных, с которыми сталкивается журналист – будь то статистика, результаты опросов, бюджетные данные или выгрузки из баз данных – имеют табличную структуру. Для эффективной работы с такими данными в Python существует мощная и гибкая библиотека Pandas. Она предоставляет высокоуровневые структуры данных и инструменты для быстрого и удобного анализа данных. Pandas является одним из ключевых инструментов в науке о данных.

Введение в библиотеку Pandas. Основными структурами данных в Pandas являются Series и DataFrame.

- Series (Серия): Это одномерный массив (похож на список Python) с ассоциированным индексом. Может хранить данные любого типа.
- import pandas as pd
- views = pd.Series([1500, 2000, 1800, 2200], index=['Article A', 'Article B', 'Article C', 'Article D'])

print(views)

content_copydownload

Use code with caution.Python

- DataFrame (Датафрейм): Это двумерная табличная структура данных с именованными столбцами (как в таблицах баз данных или электронных таблицах) и

ассоциированным индексом для строк. Каждый столбец DataFrame является объектом Series. Датафреймы идеально подходят для представления большинства реальных наборов данных.

```
data = {
   'city': ['Москва', 'Санкт-Петербург', 'Екатеринбург'],
   'population': [12630000, 5380000, 1490000],
   'region': ['Центральный', 'Северо-Западный', 'Уральский']
}
cities_df = pd.DataFrame(data)
print(cities_df)
```

content_copydownload

Use code with caution.Python

Чтение данных из файлов и их визуализация. Pandas значительно упрощает загрузку данных из различных форматов файлов. Наиболее часто используются функции:

- pd.read_csv(): для чтения данных из CSV-файлов. Поддерживает множество параметров для настройки разделителей, кодировки, обработки пропущенных значений и т.д.
- pd.read_excel(): для чтения данных из файлов Excel.
- pd.read_json(): для чтения данных из JSON-файлов.

```
# Пример чтения данных из CSV файла (предполагаем, что есть файл 'articles.csv')
# При необходимости указываем encoding='utf-8' или 'cp1251'
try:
   articles_df = pd.read_csv('articles.csv', encoding='utf-8')
   print("Данные загружены успешно:")
   print(articles_df.head()) # Вывести первые 5 строк
except FileNotFoundError:
   print("Файл articles.csv не найден.")
except Exception as e:
   print(f"Произошла ошибка при чтении файла: {e}")

# Простая визуализация (используя встроенные в Pandas возможности, основанные на Matplotlib)
# Если в данных есть числовой столбец, например 'views'
# if 'views' in articles_df.columns:
#    articles_df['views'].hist()
#    import matplotlib.pyplot as plt
#    plt.title("Распределение просмотров")
#    plt.xlabel("Просмотры")
#    plt.ylabel("Количество статей")
#    plt.show()
```

content_copydownload

Use code with caution.Python

Pandas также имеет встроенные методы для базовой визуализации, основанные на библиотеке Matplotlib, что позволяет быстро построить графики для первичного анализа данных.

Основные операции с данными в Pandas. Pandas предоставляет богатый набор инструментов для манипулирования и преобразования данных в DataFrame:

- Выбор данных: Доступ к столбцам (df['column_name']) или строкам по индексу (df.loc[], df.iloc[]).

- Фильтрация: Выбор строк по условию. Например, выбрать все статьи с более чем 1000 просмотров: articles_df[articles_df['views'] > 1000].
- Сортировка: Упорядочивание строк по значениям одного или нескольких столбцов. Например, отсортировать статьи по дате: articles_df.sort_values(by='date', ascending=False).
- Добавление/удаление столбцов: Создание новых столбцов на основе существующих или удаление ненужных.
- Обработка пропущенных значений: Выявление, удаление или заполнение пропущенных данных (NaN).
- Группировка и агрегация: Группировка данных по категориям (например, по автору, по теме) и применение агрегирующих функций (среднее, сумма, количество, медиана) к группам. Это один из самых мощных инструментов для подведения итогов по категориям. Например, посчитать среднее количество просмотров по каждой категории новостей: articles_df.groupby('category')['views'].mean().

Примеры анализа данных из журналистских исследований:
С помощью Pandas журналист может:

- Загрузить данные из открытых источников (сайты госзакупок, статистические порталы).
- Очистить данные от ошибок и пропусков.
- Отфильтровать данные по интересующим критериям (например, выбрать все контракты на сумму свыше N).
- Сгруппировать данные (например, по годам, по ведомствам) и рассчитать итоговые статистики (общая сумма расходов, количество контрактов).
- Найти аномалии или выбросы (например, необычно высокие расходы).
- Подготовить данные для визуализации.

Эти базовые операции в Pandas позволяют быстро получить представление о данных, найти интересные закономерности и подготовить почву для более глубокого анализа и создания дата-историй.

Вопросы для обсуждения:

1. Введение в библиотеку Pandas.
2. Основные объекты Pandas: DataFrame и Series.
3. Чтение данных из файлов и их визуализация.
4. Основные операции с данными в Pandas.
5. Фильтрация, сортировка, агрегация данных.
6. Примеры анализа данных из журналистских исследований.

Список литературы, рекомендуемый к использованию по данной теме

Основная литература:

1. Аллен Дауни Основы Python. Научитесь думать как программист / Аллен Б. Дауни ; пер. с англ. С. Черникова ; [науч. ред. А. Родионов]. — Москва : Манн, Иванов и Фербер, 2021. — 304 с.

Дополнительная литература:

1. Бэрри Пол Изучаем программирование на Python. -М., 2022.
2. Зингаро Даниэль Python без проблем. М., 2023.

Интернет-ресурсы:

1. http://www.biblioclub.ru – Университетская библиотека онлайн.
2. http://elibrary.ru - Научная электронная библиотека.
3. http://cyberleninka.ru/ - Научная электронная библиотека.
4. IPR SMART - Цифровой образовательный ресурс

Практическое занятие 5

Тема: Визуализация данных с Matplotlib и Seaborn

Цель занятия: освоить базовые принципы и методы создания различных типов графиков для представления данных с использованием библиотек Matplotlib и Seaborn.

Формируемые компетенции: ПК-3

Актуальность: Визуализация данных – ключевой этап дата-журналистики, позволяющий сделать сложные данные понятными и наглядными для широкой аудитории, а также обнаружить скрытые закономерности в процессе анализа.

Теоретическая часть.

После того как данные собраны и проанализированы, возникает задача представить их в понятной и убедительной форме. Визуализация данных решает эту задачу, превращая ряды чисел в графики, диаграммы и карты. Хорошая визуализация не только иллюстрирует выводы анализа, но и может сама по себе стать основой для истории или помочь обнаружить неочевидные тренды. В Python для этих целей существует несколько мощных библиотек. Наиболее популярными являются Matplotlib и Seaborn. Python предоставляет возможности для создания графики и визуализации данных.

Основы визуализации данных.
Matplotlib – это фундаментальная библиотека для создания статичных, анимированных и интерактивных визуализаций в Python. Seaborn – это библиотека, построенная поверх Matplotlib, которая предоставляет более высокоуровневый интерфейс для создания красивых и информативных статистических графиков.
Процесс создания графика в Matplotlib обычно включает:

1. Импорт библиотеки: import matplotlib.pyplot as plt.
2. Подготовку данных: данных, которые будут отображены на осях X и Y.
3. Выбор типа графика: в зависимости от типа данных и задачи (например, линейный график для трендов, столбчатая диаграмма для сравнения категорий).
4. Построение графика с использованием функций Matplotlib.
5. Настройку элементов графика: добавление заголовка, подписей осей, легенды, изменение цветов и стилей.
6. Отображение графика: plt.show().

Построение графиков: линейные, столбчатые, круговые диаграммы.

- Линейные графики: Используются для отображения изменений данных во времени или по другой упорядоченной категории. plt.plot(x_data, y_data).
- # Пример линейного графика: динамика просмотров статьи по дням
- days = [1, 2, 3, 4, 5]
- views_per_day = [100, 150, 120, 200, 180]
- plt.plot(days, views_per_day)
- plt.title("Динамика просмотров статьи")
- plt.xlabel("День")
- plt.ylabel("Количество просмотров")

plt.show()
content_copydownload
Use code with caution.Python

- Столбчатые диаграммы: Подходят для сравнения значений между различными категориями. plt.bar(categories, values).
- # Пример столбчатой диаграммы: количество статей по категориям
- categories = ['Политика', 'Экономика', 'Культура']
- counts = [50, 35, 45]
- plt.bar(categories, counts)
- plt.title("Количество статей по категориям")

- plt.xlabel("Категория")
- plt.ylabel("Количество")

plt.show()
content_copydownload
Use code with caution.Python

- Круговые диаграммы (Pie charts): Используются для показа долей целого. plt.pie(sizes, labels=labels). *Важно: часто критикуются за сложность сравнения долей, рекомендуется использовать с осторожностью.*
- # Пример круговой диаграммы: доли трафика из разных источников
- sources = ['Прямой', 'Соцсети', 'Поиск']
- shares = [30, 45, 25]
- plt.pie(shares, labels=sources, autopct='%1.1f%%')
- plt.title("Источники трафика")

plt.show()
content_copydownload
Use code with caution.Python

Seaborn облегчает создание более сложных статистических графиков (например, ящичковые диаграммы, тепловые карты, распределения) и имеет привлекательные цветовые палитры по умолчанию.

Примеры визуализации данных из журналистских расследований: Визуализация данных позволяет журналисту:

- Проиллюстрировать тренд (например, рост или падение показателей с течением времени).
- Сравнить данные по регионам, категориям или группам.
- Показать распределение данных (например, как доходы распределяются по населению).
- Выявить взаимосвязи между переменными.
- Сделать выводы анализа более понятными и убедительными для читателя.

Создание интерактивных графиков. Для веб-публикаций или презентаций часто требуются интерактивные графики, позволяющие пользователю взаимодействовать с данными (наводить курсор для получения информации, масштабировать, фильтровать). Библиотека Plotly – один из лучших инструментов для создания таких графиков в Python. Plotly позволяет создавать веб-интерактивные графики, которые можно легко встраивать в онлайн-материалы. Примеры интерактивных графиков могут включать карты с всплывающими подсказками, динамические графики, меняющиеся при выборе параметров, или графики с возможностью детализации данных.

Вопросы для обсуждения:

1. Основы визуализации данных.
2. Построение графиков: линейные, столбчатые, круговые диаграммы.
3. Примеры визуализации данных из журналистских расследований.
4. Создание интерактивных графиков.
5. Использование библиотеки Plotly для создания интерактивных визуализаций.
6. Примеры интерактивных графиков для презентаций.

Список литературы, рекомендуемый к использованию по данной теме

Основная литература:

1. Аллен Дауни Основы Python. Научитесь думать как программист / Аллен Б. Дауни ; пер. с англ. С. Черникова ; [науч. ред. А. Родионов]. — Москва : Манн, Иванов и Фербер, 2021. — 304 с.

Дополнительная литература:

1. Бэрри Пол Изучаем программирование на Python. -М., 2022.
2. Зингаро Даниэль Python без проблем. М., 2023.

Интернет-ресурсы:

1. http://www.biblioclub.ru – Университетская библиотека онлайн.
2. http://elibrary.ru - Научная электронная библиотека.
3. http://cyberleninka.ru/ - Научная электронная библиотека.
4. IPR SMART - Цифровой образовательный ресурс

Практическое занятие 6

Тема: Обработка текстовых данных

Цель занятия: освоить базовые методы работы с текстом в Python и познакомиться с инструментами для лингвистического анализа.

Формируемые компетенции: ПК-3

Актуальность: большой объем информации в журналистике представлен в текстовом формате (статьи, комментарии, стенограммы, посты в соцсетях). Умение автоматизированно обрабатывать и анализировать текст позволяет выявлять тренды, мнения и ключевые темы.

Теоретическая часть.

Текстовые данные являются одним из основных источников информации для журналиста. Ручной анализ больших объемов текста крайне трудоемок и неэффективен. Python с его мощными инструментами для работы со строками и специализированными библиотеками для обработки естественного языка (NLP - Natural Language Processing) предоставляет возможность автоматизировать многие задачи, связанные с текстовым анализом.

Основы обработки текста в Python.

В Python строки являются неизменяемыми последовательностями символов. Базовые операции со строками включают:

- Создание строк: С помощью одинарных ('...'), двойных ("...") или тройных кавычек ('''...''' или """..."""). Тройные кавычки удобны для многострочных текстов.
- Работа со строками:
  - Поиск: Определение наличия подстроки (in) или ее позиции (find(), index()).
  - Замена: Замена одних подстрок на другие (replace()).
  - Разделение текста: Разбиение строки на список подстрок по заданному разделителю (split()).
  - Объединение: Сборка списка строк в одну строку с заданным разделителем (join()).
  - Изменение регистра: Преобразование к верхнему (upper()) или нижнему (lower()) регистру.
  - Удаление пробелов: Удаление пробельных символов с начала и/или конца строки (strip()).
  - Форматирование: Вставка значений переменных в строку (например, f-строки в Python 3.6+).

```
text = "Это пример текста для анализа."
words = text.split(" ") # Разделение по пробелу
print(words) # Выведет ['Это', 'пример', 'текста', 'для', 'анализа.']

new_text = " ".join(words) # Объединение обратно с пробелом
print(new_text) # Выведет "Это пример текста для анализа."

replaced_text = text.replace("пример", "образец")
```

print(replaced_text) # Выведет "Это образец текста для анализа."

content_copydownload

Use code with caution.Python

Эти базовые операции позволяют выполнять первичную очистку и подготовку текстовых данных.

Анализ текста с использованием NLTK и SpaCy.

Для более глубокого лингвистического анализа используются специализированные библиотеки, такие как NLTK (Natural Language Toolkit) и SpaCy. NLTK – более старая и обширная библиотека, часто используемая для исследований и обучения. SpaCy – более новая, быстрая и ориентированная на "продакшн" библиотека с хорошей поддержкой русского языка.

Основные задачи, решаемые с их помощью:

- Токенизация: Разбиение текста на базовые единицы – токены (слова, знаки препинания).
- Лемматизация/Стемминг: Приведение слов к их нормальной форме (лемме) или основе (стему). Например, "бегу", "бежал", "бежать" -> "бежать". Это нужно для корректного подсчета частотности слов.
- Частотный анализ слов: Подсчет количества вхождений каждого слова в тексте. Позволяет выявить наиболее часто употребляемые термины.
- Удаление "стоп-слов": Удаление часто встречающихся, но не несущих смысловой нагрузки слов (предлоги, союзы, частицы).
- Анализ тональности (сентимента): Определение эмоциональной окраски текста (позитивная, негативная, нейтральная).
- Извлечение именованных сущностей (NER - Named Entity Recognition): Выделение из текста имен людей, организаций, географических названий.

Пример использования NLTK для токенизации и частотного анализа (требует установки nltk и скачивания соответствующих корпусов)

import nltk

from nltk.corpus import stopwords

from nltk.tokenize import word_tokenize

text = "Анализ текстовых данных в журналистике становится все более актуальным инструментом."

tokens = word_tokenize(text.lower(), language='russian') # Токенизация и приведение к нижнему регистру

stop_words = set(stopwords.words('russian'))

filtered_tokens = [word for word in tokens if word.isalnum() and word not in stop_words] # Удаление стоп-слов и знаков препинания

freq_dist = nltk.FreqDist(filtered_tokens) # Частотный анализ

print(freq_dist.most_common(5)) # Вывести 5 самых частых слов

Пример использования SpaCy (требует установки spacy и модели для русского языка ru_core_news_sm)

import spacy

nlp = spacy.load("ru_core_news_sm")

doc = nlp("Владимир Путин посетил Москву 27 октября.")

for entity in doc.ents: # Извлечение именованных сущностей

print(f"{entity.text} - {entity.label_}") # Выведет "Владимир Путин - PER", "Москву - LOC", "27 октября - DATE"

content_copydownload

Use code with caution.Python

Примеры применения в журналистике:

- Анализ комментариев пользователей под статьями или в социальных сетях для выявления преобладающего мнения (анализ тональности).
- Автоматическое выделение ключевых тем в большом архиве новостей.
- Извлечение имен компаний или людей из текстовых документов для создания базы данных.
- Анализ речей политиков или официальных документов для выявления частотности употребления определенных терминов или фраз.

Эти инструменты открывают широкие возможности для получения ценной информации из неструктурированных текстовых данных.

Вопросы для обсуждения:

1. Основы обработки текста в Python.
2. Работа со строками: поиск, замена, разделение текста.
3. Примеры анализа текстовых данных (например, анализ новостных статей).
4. Анализ текста с использованием NLTK и SpaCy.
5. Токенизация, лемматизация, частотный анализ слов.
6. Примеры применения в журналистике (анализ тональности, ключевые темы).

Список литературы, рекомендуемый к использованию по данной теме

Основная литература:

1. Аллен Дауни Основы Python. Научитесь думать как программист / Аллен Б. Дауни ; пер. с англ. С. Черникова ; [науч. ред. А. Родионов]. — Москва : Манн, Иванов и Фербер, 2021. — 304 с.

Дополнительная литература:

1. Бэрри Пол Изучаем программирование на Python. -М., 2022.
2. Зингаро Даниэль Python без проблем. М., 2023.

Интернет-ресурсы:

1. http://www.biblioclub.ru – Университетская библиотека онлайн.
2. http://elibrary.ru - Научная электронная библиотека.
3. http://cyberleninka.ru/ - Научная электронная библиотека.
4. IPR SMART - Цифровой образовательный ресурс

Практическое занятие 7

Тема: Веб-скрапинг и сбор данных

Цель занятия: изучить методы автоматизированного сбора данных с веб-сайтов (веб-скрапинг) и рассмотреть этические и правовые аспекты этого процесса.

Формируемые компетенции: ПК-3

Актуальность: множество потенциально ценных данных для журналистских расследований доступно только на веб-страницах. Веб-скрапинг позволяет автоматизировать сбор этой информации, что значительно расширяет возможности дата-журналистики.

Теоретическая часть.

Значительный объем открытых данных, представляющих интерес для журналистов (например, информация с сайтов государственных органов, судебных решений, открытых реестров, специализированных порталов), представлен в виде веб-страниц, не предоставляющих удобных инструментов для массовой выгрузки. Веб-скрапинг (или парсинг веб-страниц) – это процесс автоматизированного извлечения структурированных данных из неструктурированного или полуструктурированного контента веб-страниц.

Python является отличным инструментом для веб-скрапинга благодаря наличию специализированных библиотек. Python может использоваться для извлечения данных из интернета.

Основы веб-скрапинга. Базовый процесс веб-скрапинга включает:

1. Отправку запроса к веб-странице для получения ее HTML-кода. Библиотека requests в Python идеально подходит для этой задачи.
2. Парсинг (анализ) полученного HTML-кода для извлечения нужных данных. HTML имеет древовидную структуру, и парсеры помогают навигировать по ней.
3. Сохранение извлеченных данных в структурированном формате (например, CSV, JSON, база данных).

Использование библиотек BeautifulSoup и Scrapy для сбора данных с сайтов.

- BeautifulSoup: Это библиотека Python для парсинга HTML и XML документов. Она создает дерево парсинга, которое позволяет легко извлекать данные, следуя структуре HTML-тегов. BeautifulSoup часто используется в связке с библиотекой requests для получения контента.

```
# Пример использования requests и BeautifulSoup
import requests
from bs4 import BeautifulSoup

url = 'http://example.com' # Замените на реальный URL
try:
    response = requests.get(url)
    response.raise_for_status() # Вызовет исключение для плохих ответов (4xx или 5xx)
    soup = BeautifulSoup(response.text, 'html.parser')

    # Пример извлечения заголовка страницы
    title = soup.title.string
    print(f"Заголовок страницы: {title}")

    # Пример поиска всех ссылок на странице
    # for link in soup.find_all('a'):
    #     print(link.get('href'))

except requests.exceptions.RequestException as e:
    print(f"Ошибка при получении страницы: {e}")
```

content_copydownload

Use code with caution.Python

BeautifulSoup отлично подходит для скрапинга простых или средних по сложности сайтов.

- Scrapy: Это более мощный и комплексный фреймворк для веб-скрапинга. Он предназначен для создания крупномасштабных скрапинг-проектов, включает в себя механизмы для обработки запросов, обхода ссылок, обработки данных и хранения результатов. Scrapy более сложен в освоении, но предоставляет больше возможностей для скрапинга сложных сайтов, требующих обработки форм, аутентификации, соблюдения очередности запросов и т.д.

Примеры сбора данных для журналистских расследований:

- Сбор контактной информации или данных о компаниях с открытых реестров.
- Извлечение текстов судебных решений или законодательных актов для последующего анализа.

- Сбор данных о недвижимости, земельных участках, декларациях чиновников с открытых порталов.
- Извлечение комментариев пользователей с форумов или блогов для анализа общественного мнения (с учетом этики).

Этика и правовые аспекты веб-скрапинга. Веб-скрапинг может быть связан с этическими и правовыми вопросами. Неконтролируемый скрапинг может создать излишнюю нагрузку на сервер веб-сайта, что равносильно DDoS-атаке. Некоторые данные могут быть защищены авторским правом или содержать персональные данные, сбор которых регулируется законодательством. Основные правила и ограничения при сборе данных:

1. Проверить файл robots.txt: Большинство сайтов имеют файл /robots.txt, который содержит инструкции для поисковых роботов и других автоматизированных сборщиков данных о том, какие разделы сайта можно сканировать, а какие нет. Добросовестные скраперы должны следовать этим инструкциям.
2. Соблюдать условия пользовательского соглашения: Некоторые сайты в своих условиях использования прямо запрещают автоматизированный сбор данных. Игнорирование этих условий может иметь правовые последствия.
3. Не создавать чрезмерную нагрузку: Делать запросы к сайту с разумной задержкой, чтобы не перегружать сервер.
4. Уважать частную жизнь: Избегать сбора персональных данных, если это не является общедоступной информацией или не требуется для общественно значимого расследования, и при этом соблюдать законодательство о персональных данных.
5. Цитировать источник: При использовании собранных данных в публикации указывать источник информации.

Примеры этичного сбора данных:

- Сбор данных с официальных государственных порталов, специально предназначенных для публикации открытых данных.
- Скрапинг сайтов с явным разрешением на использование данных для некоммерческих или исследовательских целей.
- Сбор общедоступной информации, такой как заголовки новостей или списки организаций, с соблюдением правил сайта.

Понимание и соблюдение этих принципов является неотъемлемой частью профессиональной дата-журналистики.

Вопросы для обсуждения:

1. Основы веб-скрапинга.
2. Использование библиотек BeautifulSoup и Scrapy для сбора данных с сайтов.
3. Примеры сбора данных для журналистских расследований.
4. Этика и правовые аспекты веб-скрапинга.
5. Основные правила и ограничения при сборе данных.
6. Примеры этичного сбора данных.

Список литературы, рекомендуемый к использованию по данной теме

Основная литература:

1. Аллен Дауни Основы Python. Научитесь думать как программист / Аллен Б. Дауни ; пер. с англ. С. Черникова ; [науч. ред. А. Родионов]. — Москва : Манн, Иванов и Фербер, 2021. — 304 с.

Дополнительная литература:

1. Бэрри Пол Изучаем программирование на Python. -М., 2022.
2. Зингаро Даниэль Python без проблем. М., 2023.

Интернет-ресурсы:

1. http://www.biblioclub.ru – Университетская библиотека онлайн.
2. http://elibrary.ru - Научная электронная библиотека.

3. http://cyberleninka.ru/ - Научная электронная библиотека.
4. IPR SMART - Цифровой образовательный ресурс

Практическое занятие 8

Тема: Анализ данных из социальных сетей

Цель занятия: ознакомиться с возможностями сбора и анализа данных из социальных сетей с использованием API и рассмотреть примеры их применения в журналистике.

Формируемые компетенции: ПК-3

Актуальность: социальные сети являются огромным источником информации об общественных настроениях, трендах, событиях и мнениях. Умение собирать и анализировать эти данные с помощью Python позволяет журналистам получать уникальные инсайты для своих материалов.

Теоретическая часть.

Социальные сети (ВКонтакте, Telegram и др.) генерируют колоссальные объемы данных, которые отражают самые актуальные события и мнения пользователей. Для журналиста эти данные могут стать ценным ресурсом для мониторинга информационной повестки, анализа общественных реакций на события, выявления трендов и даже поиска героев или свидетелей событий. Автоматизированный сбор данных из социальных сетей, в отличие от веб-скрапинга, часто осуществляется через специальные интерфейсы – API (Application Programming Interface), предоставляемые самими платформами.

Сбор данных из социальных сетей. Большинство крупных социальных сетей предлагают API, который позволяет разработчикам (и журналистам) получать доступ к определенным типам данных (например, публичные посты, комментарии, информация о пользователях, тренды) в структурированном формате, минуя необходимость парсить HTML. Работа через API является предпочтительным методом, так как он более стабилен, надежен и соответствует правилам использования платформы. Каждая социальная сеть имеет свой собственный API с уникальными правилами доступа и ограничениями. Для работы с API в Python существуют специализированные библиотеки или можно использовать стандартную библиотеку requests для отправки HTTP-запросов к API.

1. Регистрацию приложения разработчика на платформе социальной сети для получения ключей доступа (API Key, Secret Key) или токенов.
2. Изучение документации API конкретной платформы, чтобы понять, какие данные доступны и как отправлять запросы.
3. Написание кода на Python для отправки запросов к API с использованием полученных ключей/токенов.
4. Обработку ответов API (обычно в формате JSON) и извлечение нужных данных.
5. Сохранение данных для дальнейшего анализа.

Анализ взаимодействий и трендов.

Собранные данные из социальных сетей могут быть проанализированы для выявления различных закономерностей:

Анализ хештегов: Определение наиболее популярных или быстрорастущих хештегов по определенной теме или в определенный период.

Анализ упоминаний: Выявление, кто или что чаще всего упоминается в контексте определенной темы.

Анализ лайков и репостов: Оценка виральности и популярности контента.

Анализ связей: Построение графов взаимодействия между пользователями или сообществами.

Анализ тональности

Примеры применения в журналистике (анализ общественного мнения):

Мониторинг реакций в социальных сетях на важные новости или события для понимания общественного мнения и поиска новых углов освещения.

Выявление зарождающихся социальных трендов или проблем, которые активно обсуждаются пользователями.

Анализ публичных высказываний политиков или компаний в их официальных аккаунтах.

Исследование активности и контента групп, связанных с определенными интересами или движениями.

Поиск экспертов или свидетелей событий через анализ их публичных профилей и публикаций (с соблюдением этики и приватности).

Важно помнить, что данные из социальных сетей могут не представлять мнение всего общества, а лишь активной части пользователей определенной платформы. Кроме того, сбор и использование данных из социальных сетей требуют особого внимания к этическим нормам и правилам конфиденциальности данных пользователей.

Вопросы для обсуждения:

1. Сбор данных из социальных сетей.
3. Примеры анализа данных из социальных сетей.
4. Анализ взаимодействий и трендов.
5. Анализ хештегов, упоминаний, лайков и репостов.
6. Примеры применения в журналистике (анализ общественного мнения).

Список литературы, рекомендуемый к использованию по данной теме

Основная литература:

1. Аллен Дауни Основы Python. Научитесь думать как программист / Аллен Б. Дауни ; пер. с англ. С. Черникова ; [науч. ред. А. Родионов]. — Москва : Манн, Иванов и Фербер, 2021. — 304 с.

Дополнительная литература:

1. Бэрри Пол Изучаем программирование на Python. -М., 2022.
2. Зингаро Даниэль Python без проблем. М., 2023.

Интернет-ресурсы:

1. http://www.biblioclub.ru – Университетская библиотека онлайн.
2. http://elibrary.ru - Научная электронная библиотека.
3. http://cyberleninka.ru/ - Научная электронная библиотека.
4. IPR SMART - Цифровой образовательный ресурс

Практическое занятие 9

Тема: Геопространственный анализ данных

Цель занятия: ознакомиться с основами работы с геопространственными данными в Python и рассмотреть примеры их визуализации и анализа для журналистских расследований.

Формируемые компетенции: ПК-3

Актуальность: многие важные данные имеют географическую привязку (адреса, координаты объектов, границы регионов). Анализ и визуализация геоданных позволяют выявлять пространственные закономерности и представлять информацию на картах, что делает истории более наглядными и понятными.

Теоретическая часть.

Данные о географическом расположении объектов или событий могут быть чрезвычайно информативными. Где расположены промышленные предприятия, загрязняющие воздух? В каких районах города самый высокий уровень преступности? Как распределяются доходы населения по регионам? Ответы на эти вопросы часто лежат в плоскости геопространственного анализа данных. Python, хотя и не является изначально специализированным ГИС-инструментом, обладает мощными библиотеками, которые позволяют эффективно работать с географическими данными и создавать картографические визуализации.

Основы работы с геопространственными данными.

Геопространственные данные (или геоданные) – это данные, которые имеют прямое или косвенное отношение к местоположению на поверхности Земли. Они могут быть представлены в различных форматах, включая:

Векторные данные: Представляют объекты в виде точек, линий и полигонов (замкнутых областей). Примеры: координаты городов (точки), реки или дороги (линии), границы стран или районов (полигоны). Часто хранятся в таких форматах, как Shapefile (.shp), GeoJSON, KML.

Растровые данные: Представляют поверхность в виде сетки пикселей, где каждый пиксель содержит значение (например, высота, температура, плотность населения). Примеры: спутниковые снимки, цифровые модели рельефа.

Использование библиотек GeoPandas и Folium для анализа геоданных.

Для работы с векторными геоданными в Python широко используется библиотека GeoPandas. Она расширяет функциональность Pandas, добавляя поддержку геопространственных типов данных (геометрии) и операций. GeoPandas позволяет читать геоданные из различных форматов, объединять их с табличными данными (DataFrame), выполнять геопространственные операции (например, определить, находится ли точка внутри полигона, рассчитать расстояния, пересечения) и проводить базовый анализ.

Примеры визуализации геоданных (карты, тепловые карты):

Карты с маркерами: Отображение конкретных точек (например, адреса происшествий, расположения объектов).

Карты с полигонами: Отображение границ (районов, избирательных округов) с возможностью раскраски полигонов в зависимости от показателей (например, уровень доходов, результаты выборов).

Тепловые карты (Heatmaps): Отображение плотности событий или объектов в определенной области, позволяющее увидеть "горячие" и "холодные" точки.

Карты с маршрутами или линиями: Отображение путей перемещения, транспортных связей и т.д.

Применение геопространственного анализа в журналистике:

Анализ распределения преступности: Выявление районов с наибольшим количеством правонарушений.

Исследование экологических проблем: Отображение расположения источников загрязнения и их влияния на прилегающие территории.

Анализ доступности: Изучение доступности социальных объектов (больниц, школ) для жителей разных районов.

Визуализация результатов выборов: Отображение результатов голосования по избирательным участкам или регионам.

Создание карт для интерактивных дата-историй: Встраивание динамических карт в веб-публикации, позволяющих читателю исследовать данные самостоятельно.

Геопространственный анализ и визуализация делают данные более "приземленными" и позволяют связать абстрактные цифры с конкретными местами, что очень важно для создания релевантных и понятных журналистских материалов.

Вопросы для обсуждения:

1. Основы работы с геопространственными данными.
2. Использование библиотек GeoPandas и Folium для анализа геоданных.
3. Примеры визуализации геоданных (карты, тепловые карты).
4. Применение геопространственного анализа в журналистике.
5. Примеры журналистских расследований с использованием геоданных (например, анализ экологических проблем).

Список литературы, рекомендуемый к использованию по данной теме

Основная литература:

1. Аллен Дауни Основы Python. Научитесь думать как программист / Аллен Б. Дауни ; пер. с англ. С. Черникова ; [науч. ред. А. Родионов). — Москва : Манн, Иванов и Фербер, 2021. — 304 с.

Дополнительная литература:

1. Бэрри Пол Изучаем программирование на Python. -М., 2022.
2. Зингаро Даниэль Python без проблем. М., 2023.

Интернет-ресурсы:

1. http://www.biblioclub.ru – Университетская библиотека онлайн.
2. http://elibrary.ru - Научная электронная библиотека.
3. http://cyberleninka.ru/ - Научная электронная библиотека.
4. IPR SMART - Цифровой образовательный ресурс

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ СТУДЕНТОВ ПО ОРГАНИЗАЦИИ САМОСТОЯТЕЛЬНОЙ РАБОТЫ ПО ДИСЦИПЛИНЕ «Основы анализа данных в Python»

Направление подготовки	42.03.02 – Журналистика
Направленность (профиль)	Мультимедийная журналистика и цифровые технологии

Ставрополь, 2026

СОДЕРЖАНИЕ

1. Введение

Методические рекомендации к самостоятельной работе студентов по дисциплине «Основы анализа данных в Python» разработаны в соответствии с рабочей программой дисциплины по направлению подготовки 42.03.02 – Журналистика, Направленность (профиль) – Мультимедийная журналистика и цифровые технологии .

Цель методических рекомендаций к СРС «Основы анализа данных в Python» - формирование у студентов, обучающихся по направлению подготовки 42.03.02 Журналистика, базовых теоретических знаний и практических профессиональных компетенций в области программирования на языке Python, достаточных для понимания принципов работы с данными, выполнения основных операций по сбору, обработке и анализу простых наборов данных, а также применения полученных компетенций для решения задач, связанных с анализом информации в профессиональной журналистской деятельности.

Задачи освоения дисциплины «Основы анализа данных в Python»:

- Изучить базовый синтаксис языка Python, основные типы данных, операторы и управляющие конструкции (условия, циклы), необходимые для написания простых программ и скриптов для обработки данных.
- Освоить основные встроенные структуры данных Python (строки, списки, словари, кортежи, множества), их свойства и методы работы, ориентированные на хранение, организацию и первичную обработку различных типов данных.
- Научиться работать с файлами в Python для чтения, записи и обработки данных из внешних источников (текстовые файлы, простые форматы данных), а также применять базовые алгоритмы для анализа текстовой информации и простых наборов данных (поиск, подсчет частоты, сортировка).
- Развить навыки алгоритмического мышления и отладки программ, а также получить представление о выборе подходящих структур данных и методов для эффективного решения задач по анализу и обработке информации в контексте журналистских расследований и создания медиапродуктов.

2. Общая характеристика самостоятельной работы студента при изучении дисциплины «Основы анализа данных в Python»

Самостоятельная работа студента в рамках дисциплины «Основы анализа данных в Python» понимается как планируемая учебная работа, выполняемая во внеаудиторное время по заданию и при методическом руководстве преподавателя, но без его непосредственного участия.

Самостоятельная работа направлена на формирование следующей компетенции:

Индекс	**Формулировка:**
ПК-3	Способен участвовать в производственном процессе выпуска журналистского текста и (или) продукта с применением современных цифровых технологий, специализированного программного обеспечения

Цель самостоятельной работы студентов в процессе изучения дисциплины «Основы анализа данных в Python» – научить студента осмысленно и самостоятельно работать с учебным материалом, научной информацией по дисциплине, актуальными исследованиями, а также привить навыки самостоятельного реализации журналистских проектов, основанных на данных.

Задачи самостоятельной работы:

-систематизировать и закрепить полученные теоретические знания и практические умения студентов;

- развить познавательные способности и активность студентов: творческую инициативу, самостоятельность, ответственность и организованность;

- сформировать и развить навыки ведения самостоятельной работы и овладения методикой исследования при решении разрабатываемых в учебной деятельности проблем и вопросов;

-повысить уровень подготовленности к самостоятельной работе в соответствии с содержанием дисциплины.

При изучении дисциплины предусматриваются следующие формы самостоятельной работы студента:

- самостоятельное изучение основной и дополнительной литературы по дисциплине, изучение научных публикаций;
- работа с электронными ресурсами в сети Интернет;
- подготовка к практическим занятиям;
- презентация проекта.

3. Технологическая карта самостоятельной работы

Коды реализуемых компетенций, индикатора(ов)	Вид деятельности студентов	Средства и технологии оценки	Объем часов, в том числе		
			СРС	Контактная работа с преподавателем	Всего
_7_ семестр					
ПК-3 ИД-1$_{ПК-3}$ ИД-2$_{ПК-3}$	Подготовка к практическим занятиям	собеседование	64	8	72
Итого за 7 семестр			64	8	72
Итого			64	8	72

Для выполнения самостоятельной работы необходимо пользоваться литературой, которая предложена в списке рекомендуемой литературы, Интернет-ресурсами или другими источниками по усмотрению студента.

Самостоятельная работа рассчитана на разные уровни мыслительной деятельности. Выполненная работа позволит приобрести не только знания, но и умения, навыки, а также выработать свою методику подготовки, что очень важно в дальнейшем процессе научной деятельности.

Предусмотрены следующие виды контроля: оценка участия в подготовке и проведении круглых столов.

Подробные критерии оценивания компетенций приведены в Фонде оценочных средств для проведения текущей и промежуточной аттестации.

4. Методические рекомендации по изучению теоретического материала

Чтение основной и дополнительной литературы по дисциплине.

Самостоятельная работа при чтении учебной литературы начинается с изучения конспекта материала, сделанного при слушании лекций преподавателя. Полученную информацию необходимо осмыслить. При необходимости, в конспект лекций могут быть внесены схемы, другая дополнительная информация.

Список литературы, необходимой для подготовки к практическим занятиям и лабораторным работам, круглым столам указан в методических рекомендациях к практическим занятиям. При изучении материала рекомендуется конспектировать основные идеи, составлять схемы, таблицы.

Конспектирование научно-исследовательской литературы

При изучении рекомендованных научных публикаций рекомендуется составление краткого конспекта. Конспект представляет собой дословные выписки из текста источника. При этом необходимо понимать, что конспект – это не полное переписывание чужого текста. Необходимо знать, что при написании конспекта сначала прочитывается текст – источник, в нём выделяются основные положения, подбираются примеры, идёт перекомпоновка материала, а уже затем оформляется текст конспекта. Конспект может быть полным, когда работа идёт со всем текстом источника или неполным, когда интерес представляет какой-либо один или несколько вопросов, затронутых в источнике.

Работа с электронными ресурсами в сети Интернет.

Для повышения эффективности самостоятельной работы студент должен уметь работать в поисковой системе сети Интернет и использовать найденную информацию при подготовке к занятиям. Часть ресурсов указывается в методических указаниях к практическим и лабораторным работам. Особое внимание стоит уделить поиску в электронных библиотечных системах. Поиск информации можно вести по автору, заглавию, виду издания, году издания или издательству. Также в сети Интернет доступна услуга по скачиванию учебных материалов.

5. Методические указания (по видам работ, предусмотренных рабочей программой дисциплины)

Процедура проведения презентации индвидуальных творческих проектов состоит из следующих этапов:

1. Получение темы проекта от преподавателя;
2. Подготовка содержания
3. Подготовка презентации
4. Защита проекта.

При формулировки темы творческих проектов предполагается, что каждая тема может быть раскрыта как в рамках базового, так и в рамках продвинутого уровней. Главное отличие повышенного уровня состоит **в высокой самостоятельности студента, который дает не шаблонное, а творческое решение предложенной задачи.**

Критерии оценивания проектов:

1. Последовательное выполнение предложенных алгоритмов.
2. Выбор и использование методов и приемов.
3. Творческое решение поставленной задачи
4. Анализ процесса и результата.
5. Личное участие

Подготовка докладов

Доклад - сообщение по заданной теме, с целью внести знания из дополнительной литературы, систематизировать материл, проиллюстрировать примерами, развивать навыки самостоятельной работы с научной литературой, познавательный интерес к научному познанию.

Тема доклада должна быть согласованна с преподавателем и соответствовать теме учебного занятия. Материалы при его подготовке, должны соответствовать научно-методическим требованиям вуза и быть указаны в докладе. Необходимо соблюдать регламент, оговоренный при получении задания. Иллюстрации должны быть достаточными, но не чрезмерными.

Работа студента над докладом включает отработку умения самостоятельно обобщать материал и делать выводы в заключении, умения ориентироваться в материале и отвечать на дополнительные вопросы слушателей, отработку навыков ораторства, умения проводить диспут.

Для подготовки к доклада необходимо самостоятельно выявить и изучить научные публикации и материалы профессиональных изданий на выбранную тему, подобрать аргументы, привлекая примеры из журналистской практики и высказывания отечественных или зарубежных журналистов.

Объем доклада - 5-10 страниц печатного текста. Шрифт - Times New Roman, кегль - 14, интервал - 1,5.

Доклад должен быть подготовлено самостоятельно и сдан преподавателю на практическом занятии.

При проверке задания оцениваются полнота раскрытия выбранной темы, ясность изложения собственной позиции и уровень ее аргументации, уровень привлечения фактов, мнений специалистов, ученых

Тема докладов

1. Роль Python в современном рабочем процессе дата-журналиста: Инструменты и библиотеки для каждого этапа анализа.
2. Выбор и эффективное использование структур данных Python (списки, словари и др.) для различных типов журналистских данных.
3. Работа с "грязными" данными в журналистике: Методы очистки и стандартизации неструктурированных и полуструктурированных файлов с помощью Python.
4. Продвинутые техники преобразования и агрегации данных в Pandas для получения глубоких аналитических выводов.
5. Кейс-стади: Применение Pandas для анализа открытых государственных данных (например, госзакупки, бюджетные расходы) в расследовательской журналистике.
6. Принципы и лучшие практики создания убедительных визуализаций для журналистских материалов с использованием Python-библиотек.
7. Использование интерактивных визуализаций (на основе Plotly) для вовлечения аудитории и улучшения восприятия сложных данных в онлайн-СМИ.
8. Автоматизированный контент-анализ большого объема текстовых данных (новостей, комментариев) с использованием Python и библиотек NLP.
9. Выявление скрытых закономерностей и аномалий в текстовых данных с применением базовых методов обработки текста в Python.

10. Преодоление технических вызовов при веб-скрапинге для журналистских расследований (обход блокировок, работа с динамическим контентом).
11. Этические и правовые аспекты сбора и использования данных из открытых источников в журналистике: Обзор кейсов и рекомендаций.
12. Анализ трендов и динамики обсуждений в социальных сетях с использованием данных, полученных через API.
13. Визуализация сетевых связей и анализ пользовательской активности в социальных сетях для изучения информационных потоков.
14. Картографическое представление данных: Создание тематических карт и геовизуализаций для дата-историй с использованием GeoPandas и Folium.
15. Интеграция различных типов данных (табличные, текстовые, геоданные) в Python для комплексного анализа в журналистских проектах.

Базовый уровень

Вопросы для проверки уровня обученности

Знать
1. Преимущества Python для журналистов, занимающихся анализом данных.
2. Процесс установки Python и основных библиотек для анализа данных
3. Две популярные среды разработки (IDE) или интерактивные оболочки для работы с данными в Python.
4. Понятие переменной в Python и ее роль при работе с данными (с примером).
5. Разница между изменяемыми (mutable) и неизменяемыми (immutable) типами данных в Python (на примерах).
6. Использование функций в программировании на Python для анализа данных.
7. Структура данных "список" (list) в Python и примеры операций с ним.
8. Структура данных "словарь" (dict) в Python и случаи его применения в журналистике.
9. Основные режимы работы с файлами в Python (чтение, запись) и использование with open(...).
10. Структура формата данных CSV и его чтение в Python (стандартные средства и Pandas).
11. Структура формата данных JSON и его преобразование в структуры Python.
12. Основные объекты библиотеки Pandas: Series и DataFrame (ключевые отличия).
13. Упрощение загрузки табличных данных из файлов с помощью Pandas.
14. Процесс фильтрации строк в DataFrame Pandas (с примером).
15. Сортировка данных в DataFrame Pandas.

Уметь, Владеть
1. Операция "группировка" (groupby) в Pandas и примеры ее применения.
2. Типы графиков для визуализации временных рядов и категориальных данных.
3. Назначение библиотек Matplotlib и Seaborn для визуализации данных в Python.
4. Интерактивные графики (Plotly) и их эффективность для онлайн-СМИ.
5. Базовые операции для обработки строк в Python при работе с текстовыми данными.
6. Токенизация и лемматизация/стемминг в анализе текстовых данных.
7. Применение NLTK или SpaCy для анализа тональности комментариев из социальных сетей.
8. Базовый принцип работы веб-скрапинга (парсинга веб-страниц).
9. Ключевое отличие между библиотеками BeautifulSoup и Scrapy для веб-скрапинга.
10. Основные этические и правовые нормы при сборе данных с веб-сайтов (веб-скрапинге).
11. API социальной сети для сбора данных из соцсетей.

12. Примеры использования данных из социальных сетей (например, анализ хештегов) в журналистском материале.
13. Понятие геопространственных данных и их актуальность для журналистских расследований.
14. Использование библиотек GeoPandas и Folium для визуализации географических данных.
15. Пример журналистского расследования с комплексным использованием инструментов Python.

Повышенный уровень
Повышенный уровень предполагает выполнение **практического задания** в и его объяснения на основе теоретического знания данного курса.

8.1. Перечень основной и дополнительной литературы, необходимой для освоения дисциплины (модуля)
8.1.1. Перечень основной литературы:
1. Аллен Дауни Основы Python. Научитесь думать как программист / Аллен Б. Дауни ; пер. с англ. С. Черникова ; [науч. ред. А. Родионов]. — Москва : Манн, Иванов и Фербер, 2021. — 304 с.
8.1.2. Перечень дополнительной литературы:
1. Бэрри Пол Изучаем программирование на Python. -М., 2022.
2. Зингаро Даниэль Python без проблем. М., 2023.

8.2. Перечень учебно-методического обеспечения самостоятельной работы обучающихся по дисциплине (модулю)
- Методические указания по организации самостоятельной работы студентов по дисциплине «Основы анализа данных в Python» для студентов направления подготовки 42.03.02 – Журналистика / Горбачев А.М. – Ставрополь: СКФУ, 2025.
- Методические указания к практическим работам по дисциплине «Основы анализа данных в Python» для студентов направления подготовки 42.03.02 – Журналистика / Горбачев А.М. - Ставрополь: СКФУ, 2025.
-

8.3. Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины (модуля)
1. http://www.biblioclub.ru – Университетская библиотека онлайн.
2. http://elibrary.ru - Научная электронная библиотека.
3. http://cyberleninka.ru/ - Научная электронная библиотека.
4. IPR SMART - Цифровой образовательный ресурс

8.2. Перечень учебно-методического обеспечения самостоятельной работы обучающихся по дисциплине (модулю)
- Методические указания по организации самостоятельной работы студентов по дисциплине «Основы анализа данных в Python» для студентов направления подготовки 42.03.02 – Журналистика / Горбачев А.М. – Ставрополь: СКФУ, 2025.
- Методические указания к практическим работам по дисциплине «Основы анализа данных в Python» для студентов направления подготовки 42.03.02 – Журналистика / Горбачев А.М. - Ставрополь: СКФУ, 2025.