**Методическое указание: Анализ данных в Excel с использованием Python для студентов факультета журналистики**

**Введение**

Это методическое указание обучит вас анализу данных в Excel с использованием языка программирования Python. Мы пошагово разберем, как считывать данные из файла, анализировать их, строить визуализации и генерировать данные самостоятельно. Это не только облегчит вашу будущую работу с реальными данными, но и научит основным навыкам аналитики и программирования.

**Основы программирования для анализа данных**

Python — мощный и понятный язык, широко используемый в аналитике данных. Его простой синтаксис позволяет освоить основы даже новичкам. Чтобы приступить к работе, важно понимать несколько ключевых понятий:

1. **Переменные** — это контейнеры для хранения данных. Например, чтобы сохранить число 20, пишем возраст = 20.
2. **Функции** — специальные блоки кода, выполняющие определенные задачи. Например, print() выводит текст на экран.
3. **Модули и библиотеки** — заранее написанные блоки кода, которые выполняют часто используемые задачи, такие как работа с файлами и создание графиков. Мы будем использовать несколько библиотек, например:
   * **Pandas** для работы с таблицами Excel.
   * **Matplotlib** и **Seaborn** для построения графиков.
   * **Faker** для генерации случайных данных.

Чтобы облегчить вам начало, в этом указании мы будем использовать библиотеки, выполняющие за нас часть работы. Начнем с подготовки нужной среды для работы.

**Часть 1: Установка и настройка среды**

Прежде чем перейти к анализу данных, необходимо подготовить рабочее место. Открываем консоль: WIN+R и вписываем cmd (terminal на macOS). Если Python уже установлен на вашем компьютере, откройте командную строку или терминал и установите нужные библиотеки, набрав команду:

pip install pandas matplotlib seaborn faker openpyxl

Каждая из этих библиотек нужна для выполнения отдельных задач в нашем проекте:

* **Pandas** — библиотека для работы с таблицами, как с файлами Excel.
* **Matplotlib** и **Seaborn** — инструменты для построения графиков и диаграмм.
* **Faker** — библиотека для генерации случайных данных (имена, даты, города и т.д.), что позволит создать пример таблицы с данными.
* **Openpyxl** — поддерживает чтение и запись данных в файлы Excel.

Эти библиотеки можно использовать в любом текстовом редакторе для Python, таком как **VSCode** или **IDLE**.

**Примечание:** в командной строке **pip** — это менеджер пакетов Python, который позволяет устанавливать и обновлять дополнительные библиотеки.

**Часть 2: Чтение и анализ данных из файла Excel**

Предположим, что у вас уже есть файл Excel с информацией о студентах. Наше задание — прочитать эти данные, проанализировать их и создать несколько графиков.

**Шаг 1: Подготовка к работе с Excel-файлом**

Чтобы загрузить данные из файла Excel, нам нужно использовать библиотеку Pandas, которая помогает эффективно обрабатывать и анализировать таблицы. Для примера, у нас будет файл students\_data.xlsx со следующими столбцами:

* **ФИО** — Фамилия, Имя и Отчество студента
* **Возраст** — возраст студента
* **Факультет** — название факультета, на котором учится студент
* **Средний балл** — средний балл студента
* **Город проживания** — город, в котором живет студент

**Шаг 2: Импорт данных**

После подготовки файла Excel в Python можно считывать таблицу, обрабатывая её как базу данных. Это делается с помощью функции read\_excel() библиотеки Pandas. Начнем с импорта библиотеки и загрузки данных из файла:

1. Создайте новый файл Python с расширением .py, например, student\_analysis.py.
2. Вставьте следующий код для чтения данных:

# Импортируем библиотеку pandas для работы с таблицами

import pandas as pd

# Указываем путь к файлу Excel

file\_path = 'students\_data.xlsx'

# Читаем файл и загружаем данные в переменную 'df'

df = pd.read\_excel(file\_path)

# Выводим первые несколько строк данных, чтобы убедиться, что они загружены правильно

print(df.head())

Каждая строка здесь выполняет конкретную задачу:

* import pandas as pd импортирует библиотеку Pandas и позволяет использовать её функции.
* file\_path указывает путь к файлу Excel.
* pd.read\_excel(file\_path) загружает данные в Python и сохраняет их в переменную df (сокращение от “data frame”).
* print(df.head()) отображает первые пять строк таблицы, чтобы убедиться, что данные загружены правильно. Ваша таблица должна появиться в командной строке, и вы увидите столбцы с данными.

**Совет:** Использование функции head() удобно для проверки структуры данных, особенно когда таблица большая и содержит много строк.

**Шаг 3: Работа с данными**

Теперь у нас есть данные, и мы можем начать их анализировать. Вот несколько основных команд, которые помогут вам быстро изучить содержимое таблицы:

1. **Получение общей информации о таблице**:

# Выводит краткую информацию о таблице: количество строк, столбцов и типы данных

print(df.info())

Функция info() покажет, сколько строк и столбцов в таблице, а также тип данных в каждом столбце. Это особенно полезно, если вы хотите понять, с каким типом информации имеете дело.

1. **Просмотр основных статистических показателей**:

# Получает базовую статистику: среднее, максимальное и минимальное значения, стандартное отклонение и т.д.

print(df.describe())

Команда describe() выполняет базовый статистический анализ данных числовых столбцов, таких как возраст и средний балл. Результат покажет среднее значение, минимальное и максимальное значения, а также другие показатели, такие как стандартное отклонение.

**Совет:** Эти команды помогают сориентироваться в данных и понять их структуру перед созданием графиков.

**Пример анализа данных**

Для начала давайте выберем простой анализ: посмотрим, сколько студентов на каждом факультете и как распределяется их средний балл.

1. **Распределение студентов по факультетам**:

import matplotlib.pyplot as plt

import seaborn as sns

import pandas as pd

# Указываем путь к файлу Excel  
file\_path = 'students\_data.xlsx'  
  
# Читаем файл и загружаем данные в переменную 'df'  
df = pd.read\_excel(file\_path)

# Настраиваем стиль графика

sns.set(style="whitegrid")

# Построение графика с количеством студентов на каждом факультете

plt.figure(figsize=(10, 6))

sns.countplot(data=df, x='Факультет', palette='viridis')

plt.title("Распределение студентов по факультетам")

plt.xlabel("Факультет")

plt.ylabel("Количество студентов")

plt.xticks(rotation=45) # Поворачиваем метки по оси X для удобства

plt.show()

Здесь:

* Мы используем sns.countplot из библиотеки **Seaborn** для построения столбчатой диаграммы.
* Параметры x='Факультет' и data=df указывают, что мы анализируем данные столбца "Факультет".
* Настройки plt.title, plt.xlabel, и plt.ylabel добавляют заголовок и подписи к осям графика.

**Картинка фото**

Такой график помогает увидеть, сколько студентов учится на каждом факультете, что дает визуальное представление о количестве студентов.

**Примечание:** В Python команды выполняются сверху вниз, поэтому убедитесь, что все строки кода идут в правильной последовательности.

**Заключение первой главы**

Теперь мы научились:

* Задавать Python-среду для работы с таблицами.
* Импортировать данные из файла Excel.
* Выполнять базовый анализ структуры и статистики данных.
* Строить первый график для визуализации распределения студентов по факультетам.

Эти базовые шаги дадут вам основу для понимания анализа данных и работы с таблицами. Если вы уверенно выполняете все эти шаги, переходите к следующей главе!

**Часть 3: Глубокий анализ данных и построение графиков для визуализации**

В этой части мы углубимся в анализ и узнаем, как строить графики для представления различных данных, таких как возраст студентов, средний балл по факультетам, распределение студентов по городам, и оценим влияние некоторых показателей друг на друга. Мы также рассмотрим, как добавлять к графикам заголовки, легенды и прочие элементы для улучшения восприятия.

**Шаг 1: Распределение студентов по возрасту**

Для начала посмотрим, как распределен возраст студентов. Построим гистограмму, которая покажет количество студентов для каждого возраста. Такой график удобно использовать, чтобы увидеть, в каком возрастном диапазоне находится большинство студентов.

1. **Построение гистограммы для распределения возраста**:

# Импортируем необходимые библиотеки

import matplotlib.pyplot as plt

import seaborn as sns

import pandas as pd

# Указываем путь к файлу Excel  
file\_path = 'students\_data.xlsx'  
  
# Читаем файл и загружаем данные в переменную 'df'  
df = pd.read\_excel(file\_path)

# Настраиваем стиль и размер графика

sns.set(style="whitegrid")

plt.figure(figsize=(10, 6))

# Строим гистограмму для столбца "Возраст"

sns.histplot(df['Возраст'], bins=10, kde=True, color='skyblue')

plt.title("Распределение студентов по возрасту")

plt.xlabel("Возраст")

plt.ylabel("Количество студентов")

plt.show()

Здесь:

* + sns.histplot() строит гистограмму, принимая данные о возрасте из столбца "Возраст".
  + bins=10 задает количество интервалов для возраста, а kde=True добавляет линию плотности, показывающую общую тенденцию распределения.
  + plt.title, plt.xlabel, и plt.ylabel добавляют заголовок и подписи к осям графика.

1. **Картинка фото**

Гистограмма показывает, как распределены студенты по возрастным категориям, что полезно для общего понимания структуры студентов в вузе.

**Шаг 2: Анализ среднего балла по факультетам**

Следующий шаг — понять, как распределяются средние баллы по факультетам. Мы построим диаграмму "box plot", которая позволяет визуализировать разброс баллов, включая среднее значение, минимальные и максимальные баллы, а также выбросы, если они есть.

1. **Построение box plot для среднего балла по факультетам**:

# Импортируем необходимые библиотеки

import matplotlib.pyplot as plt

import seaborn as sns

import pandas as pd

# Указываем путь к файлу Excel  
file\_path = 'students\_data.xlsx'  
  
# Читаем файл и загружаем данные в переменную 'df'  
df = pd.read\_excel(file\_path)

# Строим "box plot" для среднего балла по факультетам

plt.figure(figsize=(12, 6))

sns.boxplot(data=df, x='Факультет', y='Средний балл', palette='coolwarm')

plt.title("Распределение среднего балла по факультетам")

plt.xlabel("Факультет")

plt.ylabel("Средний балл")

plt.xticks(rotation=45)

plt.show()

В этом коде:

* + sns.boxplot() строит box plot с данными о факультетах и среднем балле.
  + palette='coolwarm' добавляет цветовую палитру, чтобы график выглядел интереснее.
  + plt.xticks(rotation=45) поворачивает метки факультетов для удобства чтения.

1. **Картинка фото**

Box plot поможет нам увидеть, как средние баллы студентов распределены по факультетам, а также выявить факультеты с наибольшим разбросом в оценках.

**Шаг 3: Распределение студентов по городам**

Теперь проанализируем, из каких городов приехали студенты. Построим круговую диаграмму, которая покажет процентное распределение студентов по городам.

1. **Построение круговой диаграммы для распределения по городам**:

# Импортируем необходимые библиотеки

import matplotlib.pyplot as plt

import seaborn as sns

import pandas as pd

# Указываем путь к файлу Excel  
file\_path = 'student\_data.xlsx'  
  
# Читаем файл и загружаем данные в переменную 'df'  
df = pd.read\_excel(file\_path)

# Получаем данные для круговой диаграммы

city\_counts = df['Город проживания'].value\_counts()

# Строим круговую диаграмму

plt.figure(figsize=(8, 8))

plt.pie(city\_counts, labels=city\_counts.index, autopct='%1.1f%%', startangle=140)

plt.title("Распределение студентов по городам проживания")

plt.show()

В этом коде:

* + df['Город проживания'].value\_counts() подсчитывает количество студентов для каждого города.
  + plt.pie() строит круговую диаграмму, где autopct='%1.1f%%' выводит процентное значение для каждого города, а startangle=140 поворачивает диаграмму для лучшего расположения сегментов.

1. **Картинка фото**

Круговая диаграмма наглядно показывает, откуда приехали студенты и в каких городах проживает большинство из них.

**Шаг 4: Влияние среднего балла на распределение по форме обучения**

Наконец, давайте посмотрим, влияет ли средний балл на форму обучения (очная или заочная форма). Для этого мы построим "strip plot" — он покажет, как средние баллы распределяются по каждой форме обучения.

1. **Построение strip plot для среднего балла по форме обучения**:

# Импортируем необходимые библиотеки

import matplotlib.pyplot as plt

import seaborn as sns

import pandas as pd

# Указываем путь к файлу Excel  
file\_path = 'student\_data.xlsx'  
  
# Читаем файл и загружаем данные в переменную 'df'  
df = pd.read\_excel(file\_path)

# Строим strip plot для среднего балла по форме обучения

plt.figure(figsize=(10, 6))

sns.stripplot(data=df, x='Форма обучения', y='Средний балл', jitter=True, palette='viridis')

plt.title("Средний балл в зависимости от формы обучения")

plt.xlabel("Форма обучения")

plt.ylabel("Средний балл")

plt.show()

Здесь:

* + sns.stripplot() строит точки для среднего балла в зависимости от формы обучения, где каждая точка представляет одного студента.
  + jitter=True немного смещает точки, чтобы было удобнее рассмотреть каждое значение, если они совпадают.

1. **Картинка фото**

Такой график поможет увидеть, отличается ли средний балл у студентов, обучающихся на разных формах.

**Итоги главы**

Мы рассмотрели несколько примеров анализа данных, включая:

* Построение гистограммы для распределения возраста студентов.
* Box plot для визуализации разброса среднего балла по факультетам.
* Круговую диаграмму для анализа распределения студентов по городам.
* Strip plot для оценки взаимосвязи между средним баллом и формой обучения.

Эти графики помогут вам быстрее и нагляднее оценивать данные, выявлять закономерности и делать выводы. В следующей главе мы изучим, как генерировать данные с помощью библиотеки Faker.

**Часть 4: Создание и заполнение Excel-файла данными с помощью библиотеки Faker**

В этой главе мы научимся создавать Excel-файлы с данными для анализа, используя библиотеку **Faker**. Это поможет быстро сгенерировать тестовые данные для практики и моделирования реальных ситуаций, что особенно полезно для обучения и тестирования.

Мы будем создавать таблицу с основными сведениями о студентах: ФИО, возраст, факультет, дата зачисления, средний балл, город проживания, специальность, номер телефона, электронную почту, наличие общежития и форма обучения. Такие данные позволят впоследствии провести полноценный анализ, как мы делали в предыдущих главах.

**Шаг 1: Установка библиотеки Faker**

Для начала установим библиотеку **Faker**. Она позволяет автоматически создавать случайные, но реалистичные данные для самых разных нужд, включая имена, города, адреса, номера телефонов и многое другое.

1. Откройте ваш командный терминал (или терминал в VS Code) и установите Faker, выполнив следующую команду:

pip install faker openpyxl

* + **openpyxl** потребуется для работы с Excel-файлами, так как она позволяет записывать данные непосредственно в формат .xlsx.

**Шаг 2: Настройка кода для генерации данных**

Теперь напишем Python-код, который создаст таблицу с 200 строками информации о студентах. Код будет генерировать случайные данные для каждого студента.

Создайте новый файл generate\_student\_data.py и добавьте в него следующий код:

from faker import Faker

import pandas as pd

import random

# Создаем объект Faker для генерации данных

fake = Faker('ru\_RU') # Устанавливаем язык на русский

# Задаем данные для факультетов и форм обучения

faculties = ["Журналистика", "Экономика", "Информатика", "Математика", "Филология"]

specialities = {

"Журналистика": ["Медиа", "Печатные СМИ", "Радиожурналистика"],

"Экономика": ["Бухгалтерия", "Финансовый анализ", "Экономическая теория"],

"Информатика": ["Программирование", "Сети", "Базы данных"],

"Математика": ["Прикладная математика", "Теоретическая математика", "Статистика"],

"Филология": ["Русский язык", "Иностранные языки", "Литература"]

}

forms\_of\_study = ["Очная", "Заочная"]

# Генерируем данные для 200 студентов

students\_data = []

for \_ in range(200):

faculty = random.choice(faculties)

speciality = random.choice(specialities[faculty])

student = {

"ФИО": fake.name(),

"Возраст": random.randint(17, 30),

"Факультет": faculty,

"Дата зачисления": fake.date\_this\_decade(),

"Средний балл": round(random.uniform(3.0, 5.0), 2),

"Город проживания": fake.city(),

"Специальность": speciality,

"Телефон": fake.phone\_number(),

"Электронная почта": fake.email(),

"Общежитие": random.choice(["Есть", "Нет"]),

"Форма обучения": random.choice(forms\_of\_study)

}

students\_data.append(student)

# Конвертируем данные в DataFrame для сохранения в Excel

df = pd.DataFrame(students\_data)

# Сохраняем DataFrame в Excel-файл

df.to\_excel("students\_data.xlsx", index=False)

print("Данные успешно сгенерированы и сохранены в файл student\_data.xlsx")

Здесь:

* + fake.name(), fake.city(), fake.phone\_number(), и fake.email() создают случайные имена, города, номера телефонов и email-адреса.
  + random.choice(faculties) случайным образом выбирает факультет для студента.
  + random.uniform(3.0, 5.0) генерирует средний балл от 3.0 до 5.0 с округлением до двух знаков.
  + fake.date\_this\_decade() создает случайные даты зачисления в текущем десятилетии.

**Шаг 3: Запуск кода и проверка данных**

Чтобы сгенерировать файл с данными, выполните файл generate\_student\_data.py, либо запустите код удобным способом:

python generate\_student\_data.py

После выполнения программы в той же директории появится файл student\_data.xlsx со сгенерированными данными. Откройте его в Excel или другом редакторе электронных таблиц, чтобы убедиться, что данные были созданы и выглядят корректно.

**Шаг 4: Основные правила для написания кода**

Перед тем как завершить, обратим внимание на несколько важных правил программирования, которые помогут вам лучше понять код:

1. **Понятные названия переменных**: Всегда используйте названия, которые легко читаются и описывают содержимое переменной, например, students\_data, faculty, specialities.
2. **Комментарии**: Не забывайте добавлять комментарии, поясняющие важные шаги программы. Это полезно как для других людей, которые читают ваш код, так и для вас, когда вы будете возвращаться к нему позже.
3. **Проверка кода на ошибки**: Перед запуском программы убедитесь, что код написан правильно. Python поможет, если в коде обнаружится синтаксическая ошибка (например, пропущена запятая), но если ошибка в логике — например, неверные названия переменных или библиотек, — программа может завершиться с ошибкой.
4. **Реалистичные данные**: При генерации тестовых данных старайтесь создавать реалистичные диапазоны значений. Например, возраст студентов мы ограничили значениями от 17 до 30 лет, а средний балл — от 3.0 до 5.0.

**Заключение**

Поздравляем! Вы успешно освоили базовые этапы работы с данными: от генерации тестовых наборов до проведения анализа и построения визуализаций. Выполнив каждый шаг этого методического указания, вы научились генерировать структурированные данные для анализа, загружать их в Python, а также создавать графики для наглядного представления информации. Эти навыки — основа, которая будет полезна в дальнейшем при работе с данными, независимо от выбранной вами специализации.

В процессе вы приобрели опыт:

* **Создания данных с помощью библиотеки Faker** — полезный инструмент для генерации обучающих и тестовых данных.
* **Использования библиотеки Pandas** для работы с таблицами: загрузка, сохранение, фильтрация и обработка данных.
* **Построения различных графиков** с библиотекой Matplotlib и Seaborn, которые помогут вам быстро и наглядно интерпретировать большие объемы информации.

Этот комплексный подход к работе с данными — умение генерировать, структурировать, анализировать и визуализировать — востребован в самых разных профессиях, связанных с анализом данных. Мы рассмотрели несколько основных правил программирования, включая понятные названия переменных, добавление комментариев и тщательную проверку кода, которые делают ваш код читаемым и надежным.

Дальше можно углубиться в анализ и работать с реальными данными, постепенно изучая более сложные аналитические и статистические методы. Надеемся, что это методическое указание поможет вам уверенно двигаться в этом направлении, и ваши будущие проекты будут полезны и интересны. Удачи в работе с данными!