

Практическая работа 8

Работа с файлами

Язык Python содержит большой набор быстрых и удобных функций по работе с файлами. В ходе задания вы научитесь создавать, редактировать и читать информацию из файлов. Все манипуляции с файлами мы будем делать за счет Питон встроенных функций.

Многие языки программирования предоставляют классы для работы с файлами и директориями проекта. Язык Python обладает множеством классов для записи и чтения данных из файлов.

Работа с файлами

При работе с файлами всегда необходимо помнить две вещи:

- Перед началом работы с файлом его необходимо открыть;
- После завершения работы с файлом его необходимо закрыть.

Если файл не открыт или же неверно открыт, то вы не можете полноценно работать с его содержимым.

С закрытием все проще, но и коварнее. Если вы не закроете файл, то программа будет работать верно, тем не менее, чем больше будет открытых файлов, тем больше программа будет перегружена и в какой-то момент она просто зависнет или выключиться.

Исключения и файлы

Поскольку не всегда известно будет ли файл в проекте или на компьютере пользователя, то всегда лучше открывать файлы за счёт использования исключений. Выполняйте открытие файлов в блоке try except и тем самым вы обезопасите себя от любых непредвиденных обстоятельств.

Работа с файлами

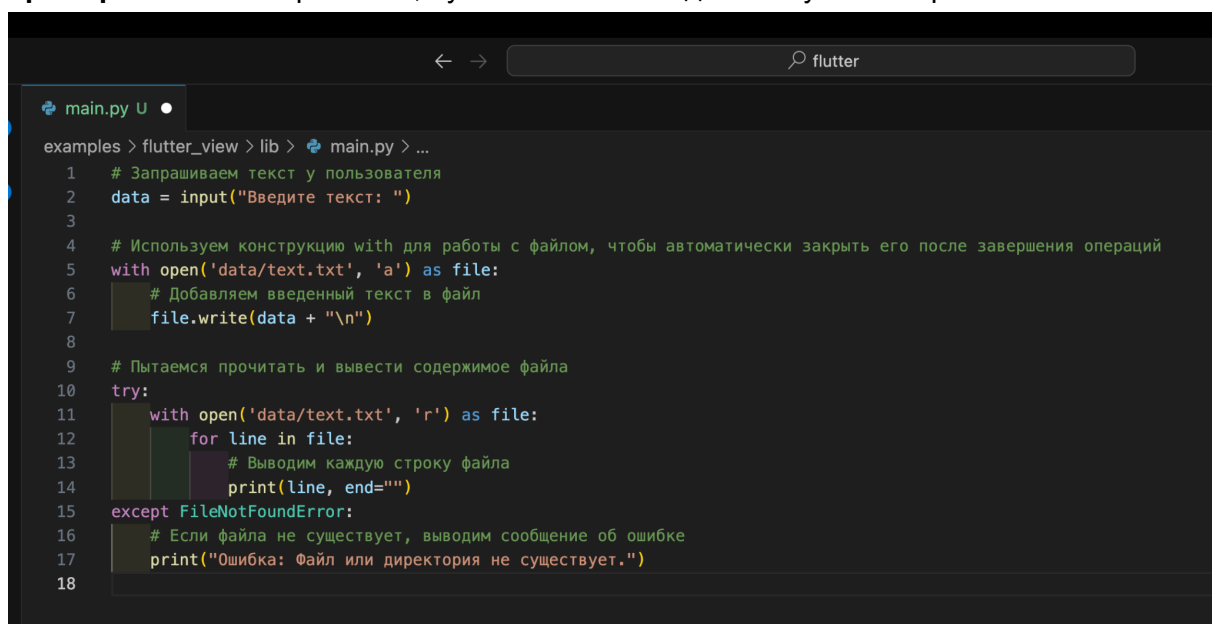
Для открытия файла существует функция open, которая открывает файл разными способами.

Вот все возможные типы открытия файла:

Режим	Обозначение
'r'	открытие на чтение (является значением по умолчанию).
'w'	открытие на запись, содержимое файла удаляется, если файла не существует, создается новый.
'x'	открытие на запись, если файла не существует, иначе исключение.
'a'	открытие на дозапись, информация добавляется в конец файла.
'b'	открытие в двоичном режиме.
't'	открытие в текстовом режиме (является значением по умолчанию).
'+'	открытие на чтение и запись

Для записи текста в файл существует метод write, а для чтения метод read. После того, как с файлом была закончена работа его обязательно необходимо закрыть. Это делается при помощи метода close.

Пример 1 :Работа с файлами, нужно сначала создать папку data с файлом text.txt



```

examples > flutter_view > lib > main.py > ...
1  # Запрашиваем текст у пользователя
2  data = input("Введите текст: ")
3
4  # Используем конструкцию with для работы с файлом, чтобы автоматически закрыть его после завершения операций
5  with open('data/text.txt', 'a') as file:
6      # Добавляем введенный текст в файл
7      file.write(data + "\n")
8
9  # Пытаемся прочитать и вывести содержимое файла
10 try:
11     with open('data/text.txt', 'r') as file:
12         for line in file:
13             # Выводим каждую строку файла
14             print(line, end="")
15 except FileNotFoundError:
16     # Если файла не существует, выводим сообщение об ошибке
17     print("Ошибка: Файл или директория не существует.")
18

```

Пример 2 Работа с файлом

- Создайте файл hi.txt и поместите в него строку: «Какая-угодно информация».
- Откройте файл для чтения и выведите информацию на экран.

```
main.py U ●
examples > flutter_view > lib > main.py > ...
1  # 1. Создание файла и запись в него строки
2
3  with open('hi.txt', 'w') as file:
4      file.write("Какая-угодно информация")
5
6  # 2. Чтение файла и вывод информации на экран
7
8  with open('hi.txt', 'r') as file:
9      data = file.read()
10     print(data)
11
```

Самостоятельная работа

1. Чтение и подсчет строк

- Создайте текстовый файл с несколькими строками текста.
- Напишите программу, которая читает содержимое этого файла и выводит количество строк.

2. Слова в файле

- Создайте текстовый файл с несколькими строками текста.
- Напишите программу, которая читает файл и выводит список всех уникальных слов, а также количество их появлений в файле.

3. Копирование файла

- Напишите программу, которая принимает имена двух файлов в качестве входных данных: исходного файла и целевого файла. Программа должна скопировать содержимое исходного файла в целевой файл.

4. Объединение файлов

- Напишите программу, которая принимает список имен файлов и имя целевого файла. Программа должна объединить содержимое всех указанных файлов в один целевой файл.

5. Поиск и замена

- Напишите программу, которая принимает имя файла, слово для поиска и слово для замены. Программа должна заменить все вхождения искомого слова на заданное слово-замену.

6. Обработка CSV-файлов

- Создайте CSV-файл с данными (например, имя, возраст, город).
- Напишите программу, которая читает этот файл и выводит информацию в отформатированном виде.

7. Логирование

- Напишите простую программу, которая просит пользователя вводить данные и записывает эти данные в файл вместе с текущей датой и временем.

8. Обратный порядок строк

- Напишите программу, которая читает файл и создает новый файл, в котором строки записаны в обратном порядке.

9. Зашифрованный дневник

- Напишите программу "дневник", которая позволяет пользователю вводить записи и сохранять их в файле.
- При сохранении записи в файл, программа должна шифровать текст.
- Также программа должна уметь читать записи из файла и дешифровать их для отображения пользователю.

10. Статистика файла

- Напишите программу, которая анализирует текстовый файл и выводит статистику: количество символов, количество слов, количество предложений, среднюю длину слова и т. д.