# Файлы

# Работа с файлами

Работа с файлами – важная часть программирования, которая позволяет приложениям сохранять, читать и обрабатывать данные, а также передавать информацию между пользователями или системами.

Файлы – это контейнеры для хранения информации, которая может быть текстовой, бинарной или иной структурированной.

**Типы файлов:**

- Текстовые файлы:
  - Содержат текстовые данные (символы, строки).
  - Примеры: .txt, .csv, .json.
  - Структура понятна человеку.
- Бинарные файлы:
  - Содержат данные в бинарном формате.
  - Примеры: изображения (.jpg, .png), видео (.mp4), программы (.exe).

# Операции с файлами

**Основные действия:**

1. Открытие файла.
2. Чтение данных.
3. Запись данных.
4. Закрытие файла.

# Открытие файла

Файл открывается с помощью функции **open()**. Она возвращает объект файла, с которым можно работать.

```
file = open('имя_файла', 'режим')
```

```
file = open('data.txt', 'w+', encoding='utf-8')
```

```
with open('data.txt', 'w+', encoding='utf-8') as f:
# действия с файлом
```

**Режимы работы с файлами/режим отрытия(access_mode)**

| Режим | Описание |
|---|---|
| "r" | Чтение (по умолчанию). Ошибка, если файла нет. |
| "w" | Запись. Создаёт файл или очищает его, если существует. |
| "a" | Добавление в конец файла. |
| "b" | Бинарный режим (например, "rb", "wb"). |
| "x" | Создание нового файла. Ошибка, если файл уже существует. |

# Закрытие файла

После окончания работы файлы надо закрывать — так же, как мы это делаем на компьютере. Для этого используется функция **close()**.

```
f = open('data.txt', 'w+')
# действия с файлом
f.close()
```

При работе с текстовым файлом через инструкцию with, инструкция close не нужна, потому что with автоматически закроет файл.

# Чтение файла и запись в файл

- метод **read()** - для чтения содержимого документа
- метод **readlines()** - преобразует все строки файла в список
- метод **readline()** - построчно выводит данные файла (удобно при работе с массивными документами)
- метод **write()** - записывает новую информацию в файл

# Чтение файла

```
with open('file.txt', 'r', encoding='utf-8') as f:
    data = f.read(6) # в файле лежит строка «Привет, Python!»
print(data)
>>> Привет

with open('file.txt', 'r', encoding='utf-8') as f:
    data = f.read() # в файле лежит строка «Привет, Python!»
print(data)
>>> Привет, Python!
```

# Чтение файла

Функция **readline()** используется для построчного чтения содержимого файла. Она используется для крупных файлов. С ее помощью можно получать доступ к любой строке в любой момент.

```
with open('file.txt', 'r', encoding='utf-8') as f:
   print(f.readline()) # если в файле одна строка, если много, дублируем


with open('file.txt', 'r', encoding='utf-8') as f:
   for line in f: # или перебором строк
       print(line)
```

# Чтение файла

Все строки файла можно прочитать с помощью метода **readlines()**

```
with open('file.txt', 'r', encoding='utf-8') as f:
    data = f.readlines()
print(data)
>>> ['Морковь\n', 'Сметана\n', 'Мука\n', 'Яблоки']

with open('file.txt', 'r', encoding='utf-8') as f:
    data = list(f)
print(data)
>>> ['Морковь\n', 'Сметана\n', 'Мука\n', 'Яблоки']
```

# Запись в файл

```
with open('file.txt', 'a', encoding='utf-8') as f:
    data = 'Привет, Python!'
    f.write(data)
```

В файл можно записать сразу список строк. Для этого применяется метод writelines().

```
with open('file.txt', 'a', encoding='utf-8') as f:
    grocery = ['Морковь', 'Яблоки', 'Мука', 'Молоко']
    f.writelines(grocery)

with open('file.txt', 'a', encoding='utf-8') as f:
    grocery = ['Морковь\n', 'Яблоки\n', 'Мука\n', 'Молоко\n']
    f.writelines(grocery)
```

# Примеры

Чтение текстового файла и запись в новый файл:

```
with open("input.txt", "r") as input_file:
      content = input_file.readlines()
with open("output.txt", "w") as output_file:
      for line in content:
            output_file.write(f"Обработанная строка: {line}")
```

Создание списка чисел и запись в файл:

```
numbers = [1, 2, 3, 4, 5]
with open("numbers.txt", "w") as file:
      for number in numbers:
            file.write(f"{number}\n")
```

# Работа с файловой системой

В Python можно работать не только с конкретным файлом, но и со всей системой. Перемещаться между каталогами, создавать новые файлы и переименовывать существующие.

Для работы с файловой системой на Python используют встроенная библиотека OS.
Ее необходимо отдельно импортировать в код проекта, чтобы получить доступа к ее методам **import** os.

```
import os
print(os.getcwd())
>>> /Users/daniilshat/PycharmProjects/pythonProject
```

```
import os
# содержимое текущего каталога
print(os.listdir())
# содержимое директории PycharmProjects
print(os.listdir('/Users/daniilshat/PycharmProjects'))
>>> ['file.txt', 'main.py', '.idea']
>>> ['.DS_Store', 'pythonProject', 'bot', 'love-couples']
```