
Московский авиационный институт
(национальный исследовательский университет)

# Работа с файлами

# Последовательность действий при работе с файлами

1. Открытие файла.
2. Чтение / запись содержимого файла.
3. Закрытие файла.

# Открытие файла. Функция open()

```
open(file, mode='r', buffering=-1,
encoding=None, errors=None, newline=None,
closefd=True, opener=None)
```

# Режимы открытия файла с помощью функции open()

| Символ | Режим открытия |
|---|---|
| `'r'` | Для чтения (значение по умолчанию) |
| `'w'` | Для записи (существующий файл будет перезаписан) |
| `'x'` | Эксклюзивное создание и запись (если файл существует, будет брошено исключение) |
| `'a'` | Для добавления в конец файла |
| `'b'` | Двоичный режим |
| `'t'` | Текстовый режим (значение по умолчанию) |
| `'+'` | Для обновления (чтения и записи) |

```python
# src/13. Files/example_01/main.py

if __name__ == "__main__":
    file = open("example.txt", "wt")

    print(type(file))
    print(dir(file))

    file.close()
```

---

<class '_io.TextIOWrapper'>
['_CHUNK_SIZE', '__class__', '__del__', '__delattr__', '__dict__', '__dir__', '__doc__', '__enter__', '__eq__', '__exit__', '__format__', '__ge__', '__getattribute__', '__getstate__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__iter__', '__le__', '__lt__', '__ne__', '__new__', '__next__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', '_checkClosed', '_checkReadable', '_checkSeekable', '_checkWritable', '_finalizing', 'buffer', 'close', 'closed', 'detach', 'encoding', 'errors', 'fileno', 'flush', 'isatty', 'line_buffering', 'mode', 'name', 'newlines', 'read', 'readable', 'readline', 'readlines', 'reconfigure', 'seek', 'seekable', 'tell', 'truncate', 'writable', 'write', 'write_through', 'writelines']

```python
# src/13. Files/example_02/main.py
# Демонстрация записи текстовых данных в файл

if __name__ == "__main__":
    file = open("example.txt", "wt")

    file.write("Hello, ")
    file.write("world!\n")

    lines = ["Привет, мир!\n", "你好世界!\n"]
    file.writelines(lines)

    file.close()
```

Содержимое файла example.txt:

```
Hello, world!
Привет, мир!
你好世界!
```

```python
# src/13. Files/example_03/main.py
# Демонстрация чтения текстового файла в виде одной строки

if __name__ == "__main__":
    file = open("example.txt", "rt")

    text = file.read()
    print(text)

    file.close()
```

---

```
Hello, world!
Привет, мир!
你好世界!
```

```python
# src/13. Files/example_04/main.py
# Демонстрация чтения текстового файла в виде списка строк

if __name__ == "__main__":
    file = open("example.txt", "rt")

    data = file.readlines()
    print(f"{data=}")

    file.close()
```

---

```
data=['Hello, world!\n', 'Привет, мир!\n', '你好世界!\n']
```

```python
# src/13. Files/example_05/main.py
# Демонстрация построчного чтения текстового файла

if __name__ == "__main__":
    file = open("example.txt", "rt")

    for line in file:
        print(line)

    file.close()
```

---

```
Hello, world!

Привет, мир!

你好世界!
```

```python
# src/13. Files/example_06/main.py
# Демонстрация порционного чтения текстового файла

if __name__ == "__main__":
    file = open("example.txt", "rt")

    # Будем читать файл последовательно по 8 символов
    buffer_size = 8

    while (text := file.read(buffer_size)) != "":
        print(text, end="|")

    file.close()
```

---

```
Hello, w|orld!
Пр|ивет, ми|р!
你好世界!|
|⏎
```

```python
# src/13. Files/example_08/writer.py
# Демонстрация одновременного открытия файла разными скриптами

if __name__ == "__main__":
    file = open("example.txt", "wt")

    lines = ["Hello, world!\n" "Привет, мир!\n", "你好世界!\n"]
    file.writelines(lines)

    input("Нажмите Enter...")

    file.close()
```

---

```python
# src/13. Files/example_08/reader.py
# Демонстрация одновременного открытия файла разными скриптами

if __name__ == "__main__":
    file = open("example.txt", "rt")

    lines = file.readlines()
    print(f"{lines=}")

    input("Нажмите Enter...")

    file.close()
```

```python
# src/13. Files/example_09/writer.py
# Демонстрация автоматического закрытия файла с помощью оператора with

if __name__ == "__main__":
    lines = ["Hello, world!\n" "Привет, мир!\n", "你好世界!\n"]

    with open("example.txt", "wt") as file:
        file.writelines(lines)

    input("Нажмите Enter...")
```

---

```python
# src/13. Files/example_09/reader.py
# Демонстрация автоматического закрытия файла с помощью оператора with

if __name__ == "__main__":
    with open("example.txt", "rt") as file:
        lines = file.readlines()

    print(f"{lines=}")
    input("Нажмите Enter...")
```

```python
# src/13. Files/example_10/writer.py
# Указание кодировки текстового файла

if __name__ == "__main__":
    lines = ["Hello, world!\n" "Привет, мир!\n"]

    with open("example.txt", "wt", encoding="cp1251") as file:
        file.writelines(lines)
```

---

```python
# src/13. Files/example_10/reader.py
# Указание кодировки текстового файла

if __name__ == "__main__":
    with open("example.txt", "rt", encoding="cp1251") as file:
        lines = file.readlines()

    print(f"{lines=}")
```

```python
# src/13. Files/example_11/reader.py
# Ошибка указания неправильной кодировки файла при чтении

if __name__ == "__main__":
    with open("example.txt", "rt", encoding="cp1251") as file:
        lines = file.readlines()

    print(f"{lines=}")
```

---

```
lines=['Hello, world!\n', 'РµСЂРёРІРµС,, РјРёСЂ!\n', 'дS\xa0eЃSдё–з•Њ!\n']
```

```python
# src/13. Files/example_12/writer.py
# Ошибка указания неправильной кодировки файла при записи

if __name__ == "__main__":
    lines = ["Hello, world!\n", "Привет, мир!\n", "你好世界!\n"]

    with open("example.txt", "wt", encoding="cp1251") as file:
        file.writelines(lines)
```

---

```
Traceback (most recent call last):
  File ".../writer.py", line 8, in <module>
    file.writelines(lines)
  File "/usr/lib/python3.11/encodings/cp1251.py", line 19, in encode
    return codecs.charmap_encode(input,self.errors,encoding_table)[0]
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
UnicodeEncodeError: 'charmap' codec can't encode characters in position 0-3: character maps to <undefined>
```

```python
# src/13. Files/example_13/writer.py
# Запись данных в файл в двоичном (бинарном) формате

if __name__ == "__main__":
    data_str = "\n".join(["Hello, world!", "Привет, мир!", "你好世界!"])
    data_bytes = data_str.encode(encoding="utf8")

    print(f"{type(data_bytes)=}")
    print(f"{data_bytes=}")

    with open("example.txt", "wb") as file:
        file.write(data_bytes)
```

---

```
type(data_bytes)=<class 'bytes'>
data_bytes=b'Hello, world!\n\xd0\x9f\xd1\x80\xd0\xb8\xd0\xb2\xd0\xb5\xd1\x82,
\xd0\xbc\xd0\xb8\xd1\x80!\n\xe4\xbd\xa0\xe5\xa5\xbd\xe4\xb8\x96\xe7\x95\x8c!'
```

---

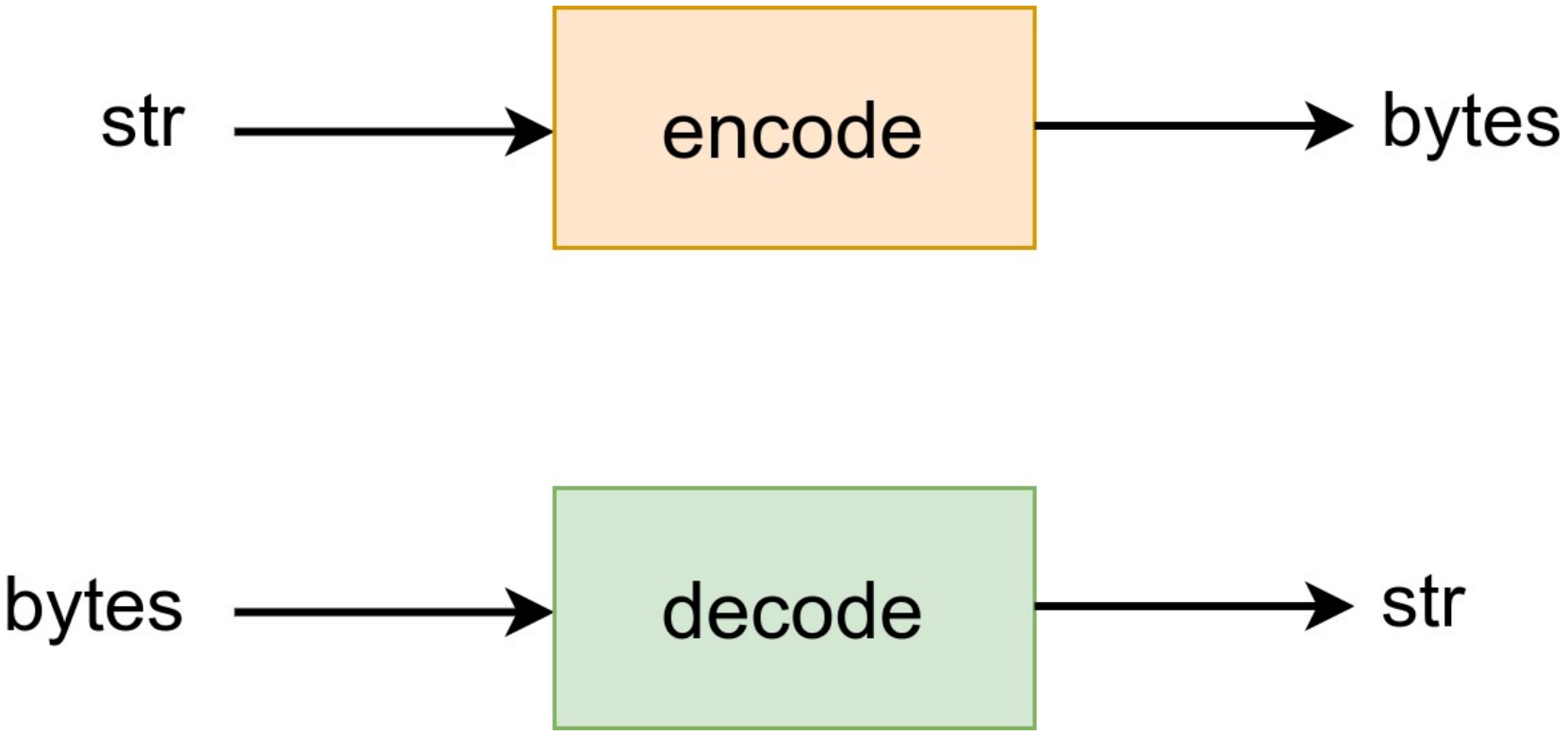
```
00000000: 48 65 6C 6C 6F 2C 20 77|6F 72 6C 64 21 0A D0 9F   Hello, world!...
00000010: D1 80 D0 B8 D0 B2 D0 B5|D1 82 2C 20 D0 BC D0 B8   ..........., ....
00000020: D1 80 21 0A E4 BD A0 E5|A5 BD E4 B8 96 E7 95 8C   ..!.............
00000030: 21                                                !
```

```python
# src/13. Files/example_13/reader.py
# Чтение данных из файла в двоичном (бинарном) формате

if __name__ == "__main__":
    with open("example.txt", "rb") as file:
        data_bytes = file.read()

    print(f"{type(data_bytes)=}")
    print(f"{data_bytes=}")

    data_str = data_bytes.decode(encoding="utf8")

    print(f"{data_str=}")
```

---

```
type(data_bytes)=<class 'bytes'>
data_bytes=b'Hello, world!\n\xd0\x9f\xd1\x80\xd0\xb8\xd0\xb2\xd0\xb5\xd1\x82,
\xd0\xbc\xd0\xb8\xd1\x80!\n\xe4\xbd\xa0\xe5\xa5\xbd\xe4\xb8\x96\xe7\x95\x8c!'
data_str='Hello, world!\nПривет, мир!\n你好世界!'
```

# encode() / decode()



```python
# src/13. Files/example_14/main.py
# Ошибка указания неправильной кодировки файла при кодировании

if __name__ == "__main__":
    text = "\n".join(["Hello, world!", "Привет, мир!", "你好世界!"])
    data_bytes = text.encode("cp1251")
```

---

```
Traceback (most recent call last):
  File ".../example_14/main.py", line 6,
in <module>
    data_bytes = text.encode("cp1251")
                 ^^^^^^^^^^^^^^^^^^^^
  File "/usr/lib/python3.11/encodings/cp1251.py", line 12, in encode
    return codecs.charmap_encode(input,errors,encoding_table)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
UnicodeEncodeError: 'charmap' codec can't encode characters in position 27-30: character maps to <undefined>
```

# Работа с файловой системой

# Стандартные модули для работы с файловой системой

- **`os`** — содержит функции для использования относительно низкоуровневых возможностей операционной системы.
- **`os.path`** — содержит функции для работы с путями файловой системы.
- **`shutil`** — содержит высокоуровневые функции для работы с файловой системой.
- **`pathlib`** — содержит классы для работы с файловой системой в объектно-ориентированном стиле.

# Некоторые функции из стандартного модуля `os.path` (1)

- **`os.path.exists()`** — предназначена для определения существования указанного пути в файловой системе.
- **`os.path.abspath()`** — возвращает абсолютный путь к указанному файлу или директории.
- **`os.path.basename()`** — возвращает имя файла или конечной директории без родительских директорий для указанного пути.
- **`os.path.dirname()`** — возвращает директорию, в которой содержится указанный файл или указанная директория.

# Некоторые функции из стандартного модуля `os.path` (2)

- **`os.path.getsize()`** — возвращает размер файла.
- **`os.path.getmtime()`** — возвращает время последней модификации файла.
- **`os.path.isfile()`** — позволяет определить, является ли указанный путь путем до файла.
- **`os.path.isdir()`** — позволяет определить, является ли указанный путь путем до директории.

# Некоторые функции из стандартного модуля `os.path` (3)

- **`os.path.join()`** — возвращает строку, содержащую путь в файловой системе, объединяющий переданные в эту функцию пути с учетом особенностей операционной системы.
- **`os.path.split()`** — разделяет переданный путь на отдельные составляющие.
- **`os.path.splitext()`** — разделяет имя файла и его расширение.

# Некоторые функции из стандартного модуля os (1)

- **`os.getcwd()`** — возвращает текущую (рабочую) директорию.
- **`os.chdir()`** — изменить текущую (рабочую) директорию.
- **`os.mkdir()`** — создать новую директорию.
- **`os.makedirs()`** - создать новую директорию и все родительские директории, указанные в параметре, если они не существуют.

# Некоторые функции из стандартного модуля os (2)

- **`os.rmdir()`** — удалить указанную директорию. Эта директория должна быть пустая.
- **`os.removedirs()`** — рекурсивно удалить указанные директорию и родительские директории. Все удаляемые директории должны быть пустыми.
- **`os.rename()`** — переименовать файл или директорию.

# Некоторые функции из стандартного модуля os (3)

- **`os.stat()`** — получить информацию о файле или директории.
- **`os.listdir()`** — возвращает список файлов и директорий в указанном родительском пути.
- **`os.scandir()`** — возвращает итератор по списку файлов и директорий в указанном родительском пути.
- **`os.walk()`** — возвращает итератор, позволяющий рекурсивно обойти дерево директорий, начиная с указанной родительской директории.

# Примеры работы с файловой системой. Использование модуля os.path

src/13. Files/example_15/

# Примеры работы с файловой системой. Использование модуля pathlib

src/13. Files/example_16/

# Некоторые функции из стандартного модуля `shutil`

- **`shutil.copy()`** — копировать файл.
- **`shutil.copytree()`** — рекурсивно копировать директорию.
- **`shutil.rmtree()`** — рекурсивно удалить директорию.
- **`shutil.move()`** — переместить файл или директорию.