IT CAREER HUB

Python

Основы работы с файлами

Преподаватель

Портрет

Имя Фамилия

Текущая должность

Количество лет опыта

Какой у Вас опыт - ключевые кейсы

Самые яркие проекты

Дополнительная информация по вашему усмотрению

Корпоративный e-mail

Социальные сети (по желанию)

Важно

- Камера должна быть включена на протяжении всего занятия
- В течение занятия вопросы задавать в чате или когда преподаватель спрашивает, есть ли у Вас вопросы
- Вести себя уважительно и этично по отношению к остальным участникам занятия
- Организационные вопросы по обучению решаются с кураторами, а не на тематических занятиях
- Во время занятия будут интерактивные задания, будьте готовы включить камеру или демонстрацию экрана по просьбе преподавателя

Повторение

- Введение в файловую систему
- Модуль os. Навигация по файловой системе
- Работа с файлами и директориями
- Работа с путями
- Модуль sys. Работа с командной строкой

План занятия

- Введение в работу с файлами. Типы файлов
- Функция open
- Режимы работы с файлами
- Параметр encoding
- Функции open и close
- Контекстный менеджер with
- Чтение из файла и запись в файл
- Другие популярные режимы
- Контекстный менеджер с несколькими файлами

ICH IT CAREER HUB

ОСНОВНОЙ БЛОК

ICH IT CAREER HUB

Введение в работу с файлами. Типы файлов

Работа с файлами

Работа с файлами — важная часть программирования, поскольку данные часто хранятся во внешних файлах. Python позволяет создавать, читать, изменять и удалять файлы.

Почему важно уметь работать с файлами?

- Хранение данных
- Обмен данными
- Обработка больших объёмов информации
- Автоматизация работы с документами

Типы файлов

Текстовые

Бинарные

Текстовые файлы

Описание

- Хранят информацию в виде обычного текста
- Используют кодировки
- Примеры расширений: .txt, .csv, .json, .html, .py

Пример

data.txt

Имя, Возраст
Анна, 25
Иван, 30

Бинарные файлы

Описание

- Хранят данные в двоичном формате (не читаемом человеком).
- Используются для хранения изображений, видео, исполняемых файлов, баз данных.
- Примеры: .jpg, .png, .mp4, .exe, .zip.

Пример

image.jpg

\xFF\xD8\xFF\xE0\x00\x10JFIF\x00\x01\x01\x00\
x00\x01\x00\x01\x00\x00

IT CAREER HUB

ВОПРОСЫ

ICH IT CAREER HUB

Функция open

Полезно знать

Функция open

Эта функция используется для работы с файлами в Python. Она позволяет открывать файлы для чтения и записи данных, включая текстовую и бинарную информацию. Функция возвращает объект файла.

Использование функции open

Синтаксис

```
open(file, mode="mode",
encoding="encoding")
```

Пояснения

- `file` – путь к файлу (относительный или абсолютный).
- `mode` *(необязательный)* – режим работы с файлом (`"r"`, `"w"`, `"a"`, `"b"` и др.).
- `encoding` *(необязательный)* – кодировка текста (`"utf-8"`, `"cp1251"` и др.). Используется только для текстовых файлов.

IT CAREER HUB

Режимы работы с файлами

Режимы работы с файлами

При открытии файла с помощью `open()` необходимо указать режим работы.

По умолчанию файл открывается в режиме `"rt"` (чтение текста)

ICH
IT CAREER HUB

Режимы работы с файлами

Режим	Описание
`"r"`	Чтение (по умолчанию). Ошибка, если файла нет.
`"w"`	Запись. Создаёт новый файл или перезаписывает существующий.
`"a"`	Добавление в конец файла. Создаёт новый, если его нет.
`"x"`	Создание нового файла. Ошибка, если файл уже существует.
`"b"`	Бинарный режим (например, `"rb"`, `"wb"` для изображений).
`"t"`	Текстовый режим (по умолчанию).
`"+"`	Открытие файла как для чтения, так и для записи (`"r+"`, `"w+"`, `"a+"`, `"x+"`).

Важно

Режимы можно комбинировать, например:

- `"rb"` – чтение в бинарном режиме.
- `"wt"` – запись в текстовом режиме.
- `"a+"` – добавление и чтение.

IT CAREER HUB

ВОПРОСЫ

ICH IT CAREER HUB

Параметр encoding

Полезно знать

ICH
IT CAREER HUB

Параметр encoding в функции open()

Этот параметр используется для указания кодировки текста при чтении или записи файлов. Он особенно важен при работе с файлами, содержащими символы, отличные от стандартных ASCII.

Почему `encoding` важен?

- Обеспечивает правильное чтение и запись файлов
- Универсальность кода
- Предотвращает ошибки `UnicodeDecodeError`

Популярные кодировки

Кодировка	Описание
utf-8 (по умолчанию)	Универсальная кодировка, поддерживает все языки.
cp1251	Кодировка Windows для русского языка.
ascii	Ограничена английскими символами (0-127), устаревшая.
utf-16	Двухбайтовая кодировка, встречается в Windows-файлах.
utf-32	Четырёхбайтовая кодировка, занимает больше места.
iso-8859-1 (Latin-1)	Кодировка для западноевропейских языков.

Особенности encoding

- Если `encoding` не указан, Python использует кодировку по умолчанию, которая зависит от операционной системы.
- Используется только для текстовых файлов (при `"rb"` и `"wb"` `encoding` не нужен).
- Неверная кодировка может вызвать `UnicodeDecodeError`

IT CAREER HUB

ВОПРОСЫ

ICH IT CAREER HUB

Функция close()

Полезно знать

Функция close()

Эта функция используется для закрытия файла после его открытия с помощью `open()`. Закрытие файла необходимо для освобождения ресурсов и предотвращения возможных ошибок при повторном доступе к нему

Пример использования close()

```
# Можно указать просто "r", а не mode="r"
file = open("example.txt", "r", encoding="utf-8")  # Открываем файл для чтения
content = file.read()  # Читаем содержимое файла
print(content)
file.close()  # Закрываем файл вручную
```

Зачем закрывать файл?

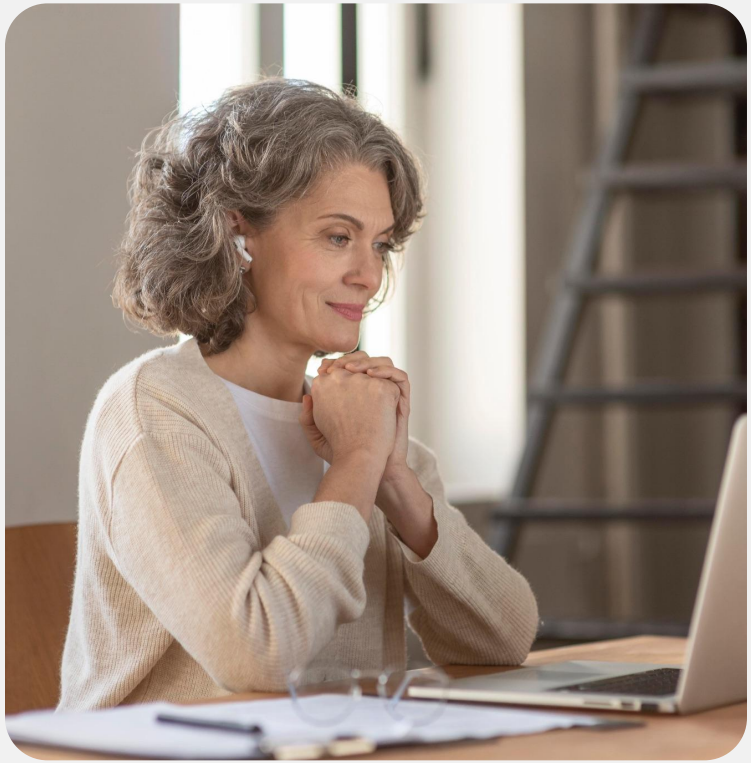
- Освобождение системных ресурсов
- Гарантированное сохранение данных
- Предотвращение ошибок доступа

IT CAREER HUB

ВОПРОСЫ

IT CAREER HUB

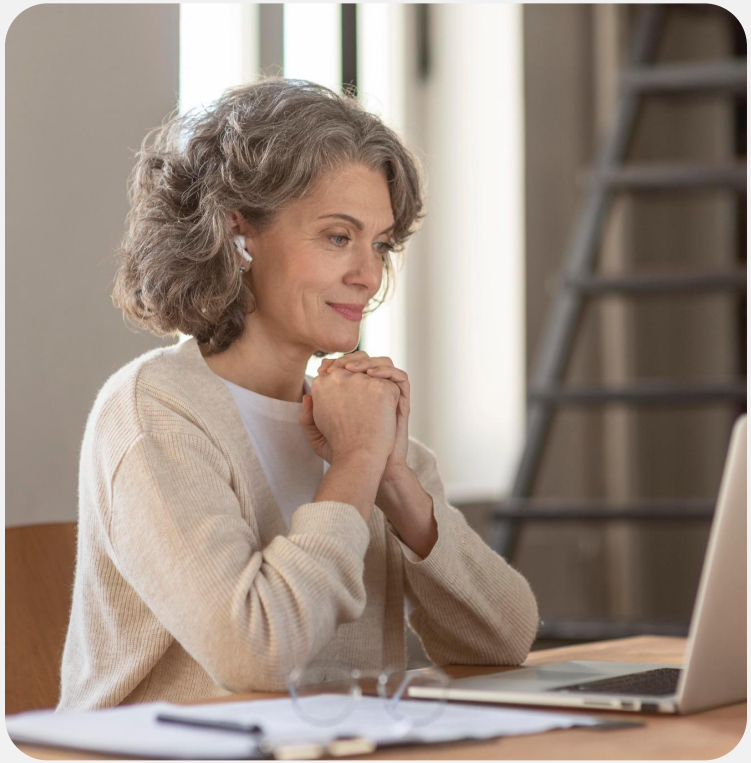
ЗАДАНИЯ



Выберите верный вариант ответа

Какой режим работы с файлом приведёт к ошибке, если файл уже существует?

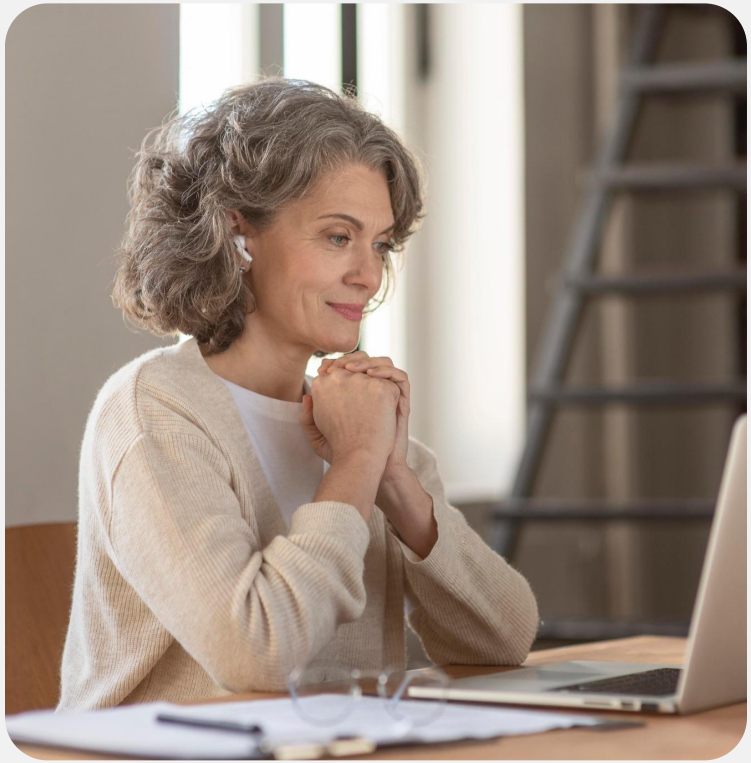
a. `"w"`
b. `"r"`
c. `"x"`
d. `"a"`



Выберите верный вариант ответа

Какой режим работы с файлом приведёт к ошибке, если файл уже существует?

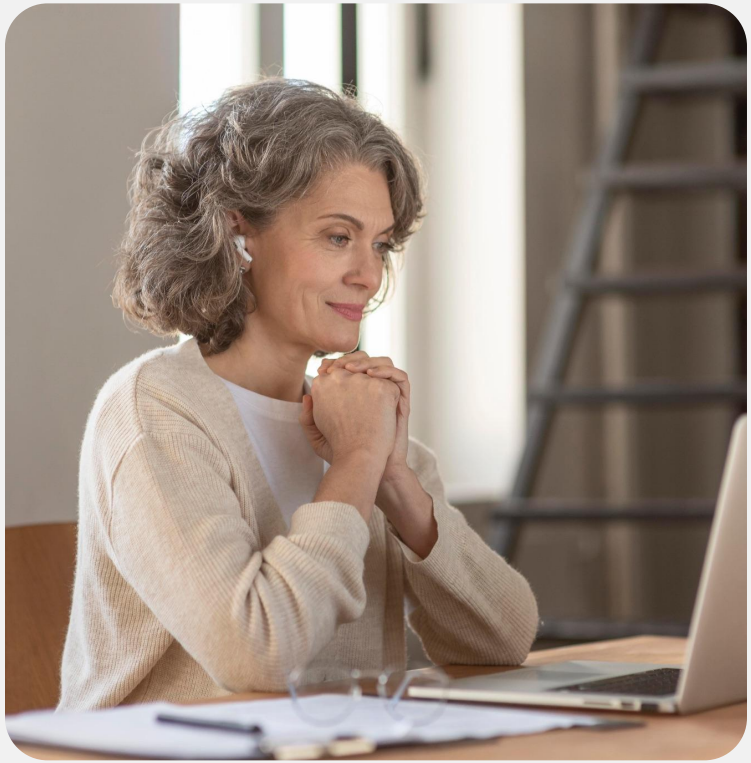
a. `"w"`
b. `"r"`
c. `"x"`
d. `"a"`



Выберите верный вариант ответа

Какой параметр обязателен для open()? Перечислите все варианты.

a. `mode`
b. `encoding`
c. `file`
d. `buffering`



Выберите верный вариант ответа

Какой параметр обязателен для open()? Перечислите все варианты.

a. mode
b. encoding
c. file
d. buffering

IT CAREER HUB

ВОПРОСЫ

ICH IT CAREER HUB

Контекстный менеджер with

Важно

Для работы с файлами используется встроенная функция `open()`, после чего файл необходимо вручную закрыть с помощью метода `close()`. Однако, если забыть закрыть файл, это может привести к утечке ресурсов.

Полезно знать

ICH
IT CAREER HUB

with

Контекстный менеджер `with` используется для безопасной работы с файлами, автоматически закрывая файл после выхода из блока. Это предпочтительный способ работы с файлами в Python

Зачем использовать with open()?

- Автоматическое закрытие файла
- Код становится чище

with open() as

Синтаксис

```
with      open(file,      mode="mode",
encoding="encoding") as variable:
    # Операции с файлом
```

Пояснение

- `with` — ключевое слово для работы с контекстными менеджерами.
- `open` — функция для открытия файла, возвращающая объект для работы с ним.
- `as` — ключевое слово для присваивания объекта файла переменной.
- `variable` — переменная, ссылающаяся на объект файла, через которую можно работать с ним в блоке `with`.

Пример использования with open()

```
with open("example.txt", "r", encoding="utf-8") as file:
    content = file.read()
    print(content)
```

Особенности with open()

- Файл открывается только внутри блока `with`.
- После выхода из блока файл **автоматически закрывается**.

IT CAREER HUB

ВОПРОСЫ

IT CAREER HUB

Чтение из файла

Чтение из файла

Python предоставляет несколько способов чтения данных из файла. Когда файл открывается, указатель устанавливается в начало файла, и по мере чтения он смещается вперёд

Полезно знать

Указатель файла

Это позиция, с которой выполняется следующее чтение или запись

Полезно знать

IT CAREER HUB

Метод read()

Этот метод позволяет считать весь текст из файла в одну строку

Чтение всего содержимого файла

Пример

```
with open("example.txt", "r",
encoding="utf-8") as file:
    content = file.read()
    print(type(content))
    print(content)
```

Особенности

- Читает весь файл целиком
- Если файл большой, может занять много памяти
- Указатель после чтения перемещается в конец файла
- Переносы строк указываются как `\n`

Полезно знать

IT CAREER HUB

Метод read(n)

Этот метод позволяет считать указанное количество символов

Чтение первых n символов

Пример

```
with      open("example.txt",      "r",
encoding="utf-8") as file:
     content = file.read(10)   # Читаем
первые 10 символов
    print(content)
```

Особенности

- Позволяет контролировать объём загружаемых данных
- Полезно при чтении больших файлов по частям

Полезно знать

ICH
IT CAREER HUB

Метод readline()

Этот метод читает одну строку за вызов

Чтение построчно

Пример

```
with open("example.txt", "r",
encoding="utf-8") as file:
    print(file.readline())
    print(file.readline())
    print(file.readline())
```

Особенности

- Читает одну строку из файла
- Каждый вызов `readline()` перемещает указатель дальше
- Если в файле несколько строк, можно вызывать `readline()` повторно

Полезно знать

IT CAREER HUB

Метод readlines()

Этот метод загружает все строки файла в список

Чтение всех строк

Пример

```
with open("example.txt", "r",
encoding="utf-8") as file:
    lines = file.readlines()
    print(type(lines))
    print(lines)
```

Особенности

- Возвращает список, где каждая строка – отдельный элемент
- Учитывает символ \n в конце строк

Чтение файла в цикле

Файл является **итерируемым объектом**, поэтому можно перебирать его содержимое **построчно в цикле** без необходимости загружать весь файл в память.

Чтение файла в цикле

Пример

```
with open("example.txt", "r", encoding="utf-8") as file:
    for line in file:
        print(line, end="")
```

Особенности

- Читает файл по одной строке, не загружая всё в память
- Подходит для обработки больших файлов

IT CAREER HUB

ВОПРОСЫ

IT CAREER HUB

Запись в файл

Полезно знать

ICH
IT CAREER HUB

Метод write()

Этот метод записывает строку в файл

Запись строки в файл: write()

Пример

```
with open("users.txt", "w", encoding="utf-8") as file:
    file.write("ID: 201 | Name: John | Age: 25 | Status: Active\n")
```

Особенности

- Создаёт файл, если его нет
- Удаляет старые данные в файле
- `write()` не добавляет символ `\n` автоматически

Полезно знать

IT CAREER HUB

Метод writelines()

Этот метод записывает список строк

Запись строки в файл: write()

Пример

```
data = [
    "ID: 202 | Name: Alice | Age: 30 |
Status: Inactive\n",
    "ID: 203 | Name: Bob | Age: 27 |
Status: Active\n"
]

with open("users.txt", "w",
encoding="utf-8") as file:
    file.writelines(data)
```

Особенности

- Принимает список строк и записывает их подряд
- \n нужно добавлять вручную, иначе строки сольются

IT CAREER HUB

ВОПРОСЫ

ICH IT CAREER HUB

Другие популярные режимы

Полезно знать

ICH
IT CAREER HUB

Режим "a"

Этот режим позволяет сохранять существующее содержимое файла и дописывать данные в конец

Режим "a" – добавление в файл

Пример

```
with open("users.txt", "a", encoding="utf-8") as file:
    file.write("Дополнительная строка\n")
```

Особенности

- Создаёт файл, если его нет
- Записывает данные в конец файла, а не перезаписывает его

Полезно знать

Режим "r+"

Этот режим позволяет и читать, и записывать данные в файл без удаления содержимого

Режим "r+" – чтение и запись

Пример

```
with open("users.txt", "r+", encoding="utf-8") as file:
    content = file.read()  # Читаем текущие данные
    file.write("\nДобавленный текст")
# Записываем новые данные
```

Особенности

- Файл должен существовать, иначе возникнет ошибка
- Чтение начинается с начала файла
- Запись начинается с текущей позиции указателя, что может привести к частичной перезаписи данных

Полезно знать

IT CAREER HUB

Режим "x"

Этот режим используется, если нужно гарантированно создать новый файл и избежать перезаписи существующих данных

Режим "x" – создание нового файла

Пример

```
try:
    with open("new_file.txt", "x",
encoding="utf-8") as file:
        file.write("Этот файл создан в
режиме 'x'.\n")
except FileExistsError:
    print("Файл уже существует.")
```

Особенности

- Если файл уже существует, возникает `FileExistsError`
- Полезно, если важно не перезаписывать существующие данные

ICH IT CAREER HUB

Контекстный менеджер с несколькими файлами

Контекстный менеджер с несколькими файлами

Можно открывать несколько файлов одновременно

Использование контекстного менеджера с несколькими файлами

Пример

```
with     (open("example.txt",     "r",
encoding="utf-8") as infile,
             open("output.txt",   "w",
encoding="utf-8") as outfile):
    for line in infile:
         outfile.write(line.upper())   #
Записываем в верхнем регистре
```

Особенности

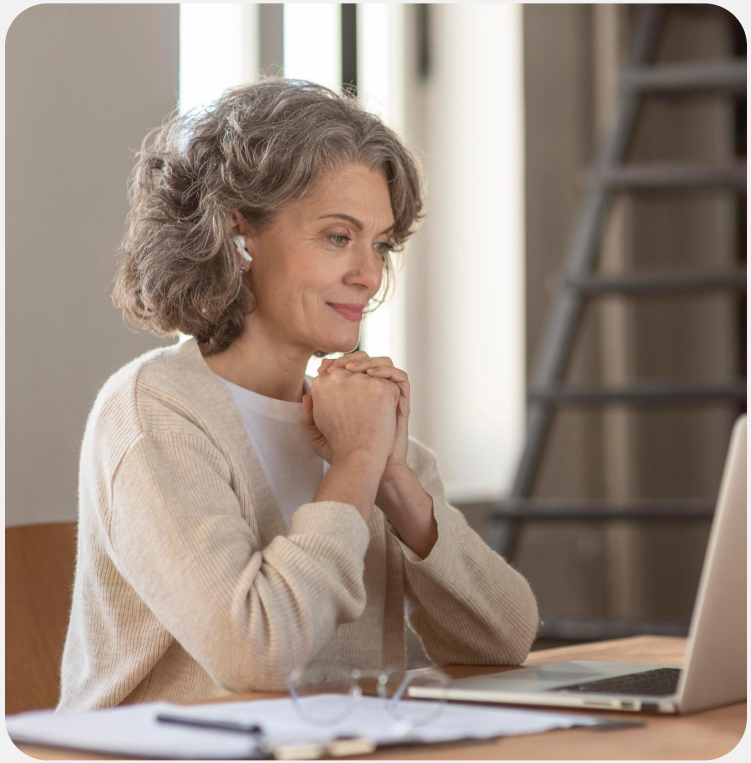
- Открывает сразу два файла.
- Читает из `input.txt` и записывает изменённый текст в `output.txt`

IT CAREER HUB

ВОПРОСЫ

IT CAREER HUB

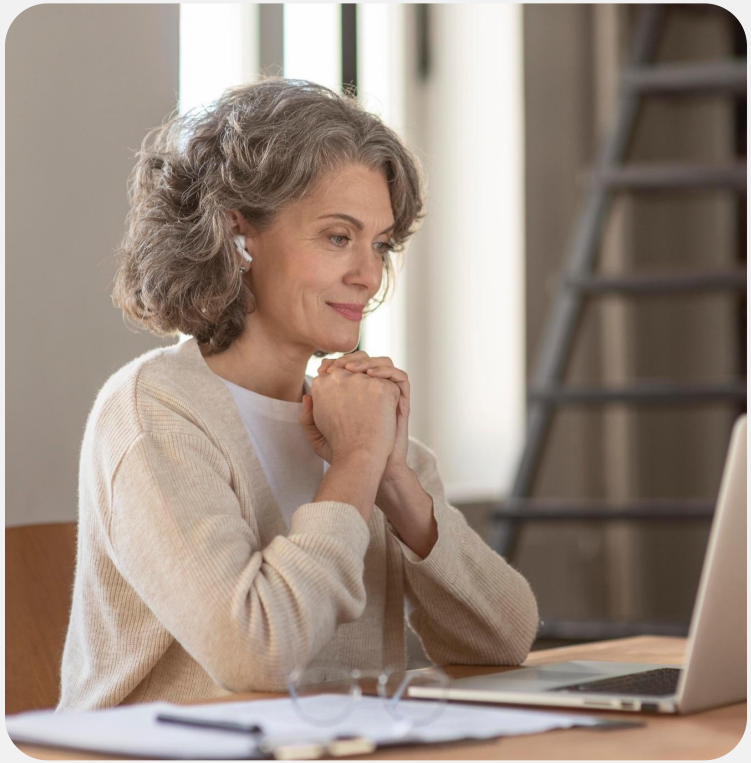
ЗАДАНИЯ



Выберите верный вариант ответа

Какой метод возвращает список строк из файла?

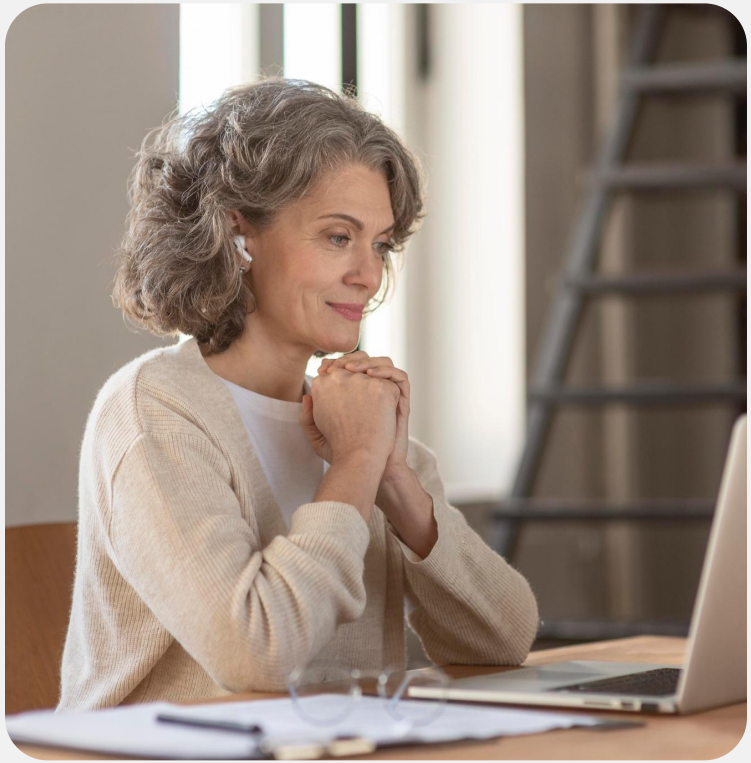
a. `read()`
b. `readline()`
c. `readlines()`
d. `splitlines()`



Выберите верный вариант ответа

Какой метод возвращает список строк из файла?

a. `read()`
b. `readline()`
c. `readlines()`
d. `splitlines()`

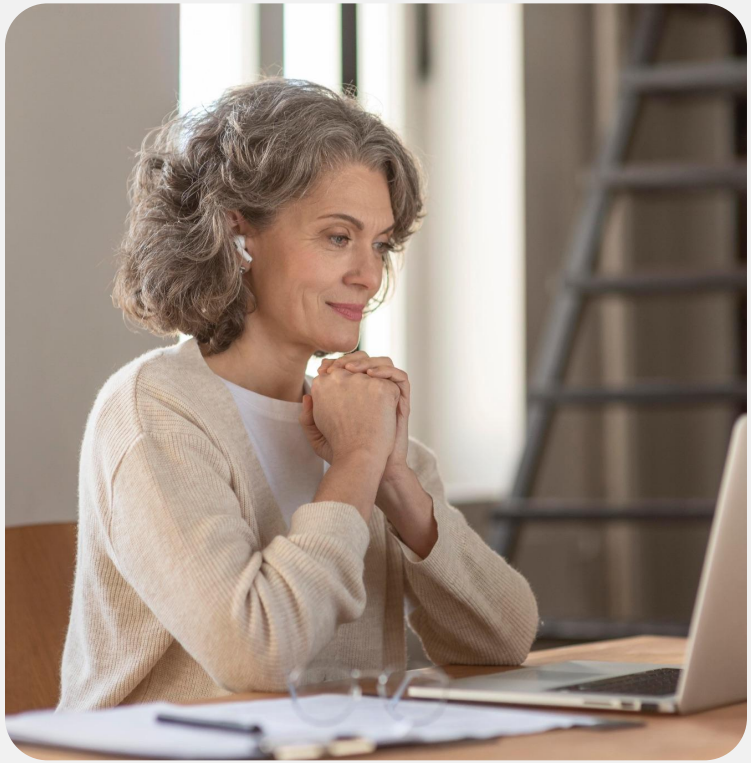


Выберите верный вариант ответа

Что делает следующий код? Перечислите все варианты.

```
with open("data.txt", "a",
encoding="utf-8") as file:
    file.write("Новая строка\n")
```

a. Перезаписывает файл и добавляет "Новая строка"
b. Добавляет "Новая строка" в конец файла
c. Очищает файл перед записью
d. Создаёт новый файл, если его нет



Выберите верный вариант ответа

Что делает следующий код? Перечислите все варианты.

```
with open("data.txt", "a",
encoding="utf-8") as file:
    file.write("Новая строка\n")
```

a. Перезаписывает файл и добавляет "Новая строка"
b. Добавляет "Новая строка" в конец файла
c. Очищает файл перед записью
d. Создаёт новый файл, если его нет

IT CAREER HUB

ВОПРОСЫ

IT CAREER HUB

ПРАКТИЧЕСКАЯ РАБОТА

Подсчёт частоты слов в файле

Напишите программу, которая подсчитывает, сколько раз каждое слово встречается в файле (не учитывая регистр).

- Программа запрашивает имя файла и количество популярных слов для вывода.
- Если указанный файл не существует, программа должна вывести ошибку.

Используйте файл text.txt.

Пример ввода:

```
Введите имя файла: text.txt
Введите количество популярных слов: 3
```

Пример вывода:

```
Популярные слова:
python: 4
is: 3
in: 2
```

Удаление пустых строк

Напишите программу, которая удаляет пустые строки из указанного пользователем файла и записывает результат в новый файл.

Имя нового файла формируется автоматически по шаблону <oldname>_cleaned.txt.

Если указанный файл не существует, программа должна вывести ошибку.

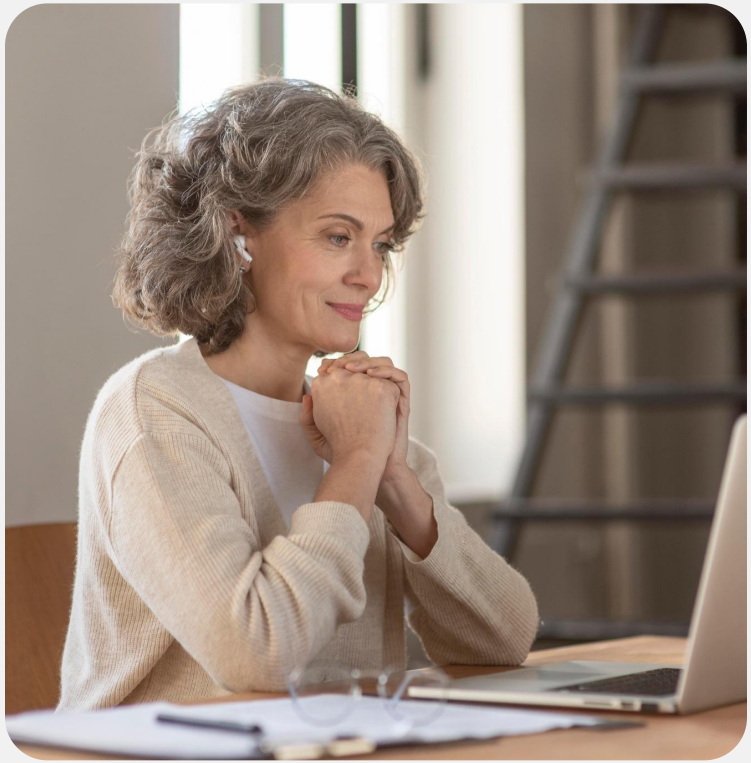
Используйте файл tasks.txt.

Пример ввода:

```
Введите имя файла: tasks.txt
```

Пример вывода:

```
Пустые строки удалены, сохранено в tasks_cleaned.txt.
```



Файлы с заданным расширением

Напишите программу, которая:

- Принимает путь к директории и расширение файлов через аргументы командной строки.
- Ищет файлы с указанным расширением в указанной директории.
- Выводит список найденных файлов

Пример запуска

```
python script.py /home/user/projects .py
```

Пример вывода

```
Найденные файлы в директории '/home/user/projects':
script.py, test.py, main.py
```

IT CAREER HUB

ДОМАШНЕЕ ЗАДАНИЕ

Домашнее задание

1. **Фильтрация по ключевому слову**

Напишите программу, которая ищет в файле все строки, содержащие указанное пользователем слово, и сохраняет их в новый файл.

- Имя нового файла формируется как `<keyword>_<original_filename>`.
- Если файл не существует, программа должна вывести ошибку.
- Если совпадения не найдены, новый файл не создаётся.

Используйте файл system_log.txt.

Пример ввода:

```
Введите имя файла для поиска: system_log.txt
Введите ключевое слово: error
```

Пример вывода:

```
Строки, содержащие 'error',
сохранены в error_system_log.txt.
```

Домашнее задание

Поиск и удаление файлов с указанным расширением

2. **Поиск и удаление дубликатов**

Напишите программу, которая удаляет дублирующиеся строки из файла и сохраняет результат в новый файл.

- Имя нового файла формируется как `unique_<original_filename>`.
- Если файл не существует, программа должна вывести ошибку.
- Исходный порядок строк должен сохраниться.
- Если в файле нет дубликатов, создаётся точная копия файла.

Используйте файл `movies_to_watch.txt`.

Пример ввода:

```
Введите имя файла: movies_to_watch.txt
```

Пример вывода:

```
Дубликаты удалены. Уникальные
строки сохранены в
unique_movies_to_watch.txt.
```

Заключение