

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №1
дисциплины «Анализ данных»

Выполнил:
Говоров Егор Юрьевич
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и
вычислительная техника»,
направленность (профиль)
«Программное обеспечение средств
вычислительной техники и
автоматизированных систем», очная
форма обучения

(подпись)

Руководитель практики:
Воронкин Р. А., канд. технических
наук, доцент кафедры
инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

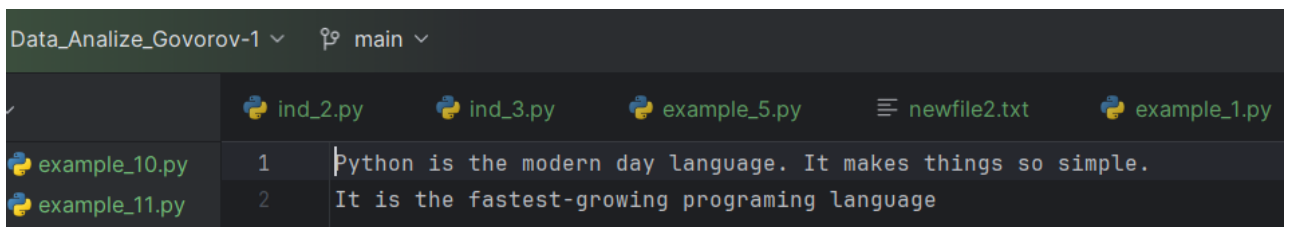
Ставрополь, 2024 г.

Тема: работа с файлами в языке Python

Цель: приобретение навыков по работе с текстовыми файлами при написании программ с помощью языка программирования Python версии 3.x, изучение основных методов модуля os для работы с файловой системой, получение аргументов командной строки.

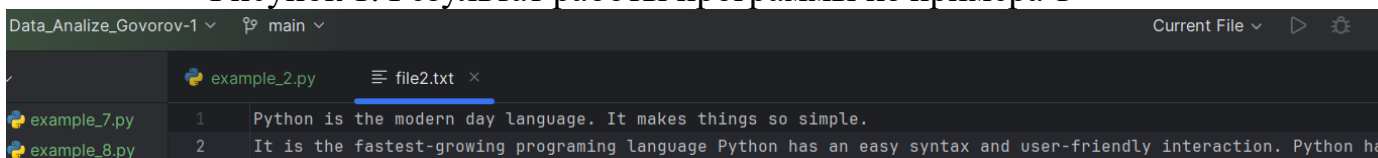
Ход работы

1. Создал общедоступный репозиторий на GitHub, в котором использована лицензия MIT и язык программирования Python. Выполнил клонирование созданного репозитория.
2. Дополнил файл .gitignore необходимыми правилами.
3. Организовал созданный репозиторий в соответствие с моделью ветвления git-flow.
4. Проработал примеры лабораторной работы. Создал для них отдельные модули языка Python. Привел в отчете скриншоты результата выполнения программ примеров при различных исходных данных, вводимых с клавиатуры.



```
Data_Analize_Govorov-1  main
ind_2.py ind_3.py example_5.py newfile2.txt example_1.py
example_10.py 1 Python is the modern day language. It makes things so simple.
example_11.py 2 It is the fastest-growing programming language
```

Рисунок 1. Результат работы программы из примера 1



```
Data_Analize_Govorov-1  main  Current File
example_2.py file2.txt
example_7.py 1 Python is the modern day language. It makes things so simple.
example_8.py 2 It is the fastest-growing programming language Python has an easy syntax and user-friendly interaction. Python ha
```

Рисунок 2. Результат работы программы из примера 2

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 if __name__ == "__main__":
5     # open the file2.txt in read mode. causes error if no such file exists.
6     fileptr = open("file2.txt", "r")
7
8     # stores all the data of the file into the variable content
9     content1 = fileptr.readline()
10    content2 = fileptr.readline()
11
12    # prints the content of the file
13    print(content1)
14    print(content2)
15
16    # closes the opened file
17    fileptr.close()
18
19    # open the file2.txt in read mode. causes error if no such file exists.
20    with open("file2.txt", "r") as fileptr:
21        # stores all the data of the file into the variable content
22        content1 = fileptr.readline()
23        content2 = fileptr.readline()
24
25        # prints the content of the file
26        print(content1)
27        print(content2)
```

Process finished with exit code 0

Рисунок 3. Результат работы программы из примера 3

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 if __name__ == "__main__":
5     # open the file2.txt in read mode. causes error if no such file exists.
6     fileptr = open("file2.txt", "r")
7
8     # stores all the data of the file into the variable content
9     content = fileptr.readlines()
10
11    # prints the content of the file
12    print(content)
13
14    # closes the opened file
15    fileptr.close()
16
17    # open the file2.txt in read mode. causes error if no such file exists.
18    with open("file2.txt", "r") as fileptr:
19        # stores all the data of the file into the variable content
20        content = fileptr.readlines()
21
22        # prints the content of the file
23        print(content)
```

Process finished with exit code 0

Рисунок 4. Результат работы программы из примера 4

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 if __name__ == "__main__":
5     # open the newfile.txt in read mode. causes error if no such file exists
6     fileptr = open("newfile.txt", "x")
7     print(fileptr)
8
9     if fileptr:
10        print("File created successfully")
11
12    # closes the opened file
13    fileptr.close()
14
15    # open the newfile2.txt in read mode. causes error if no such file exists
16    with open("newfile2.txt", "x") as fileptr:
17        print(fileptr)
18
19    if fileptr:
20        print("File created successfully")
```

Рисунок 5. Результат работы программы из примера 5

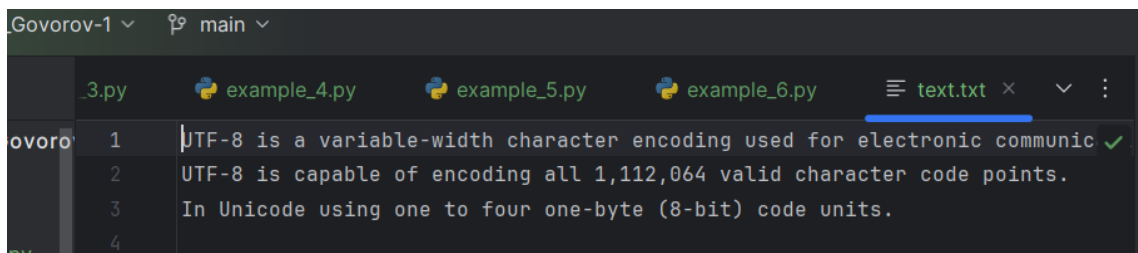


Рисунок 6. Результат работы программы из примера 6

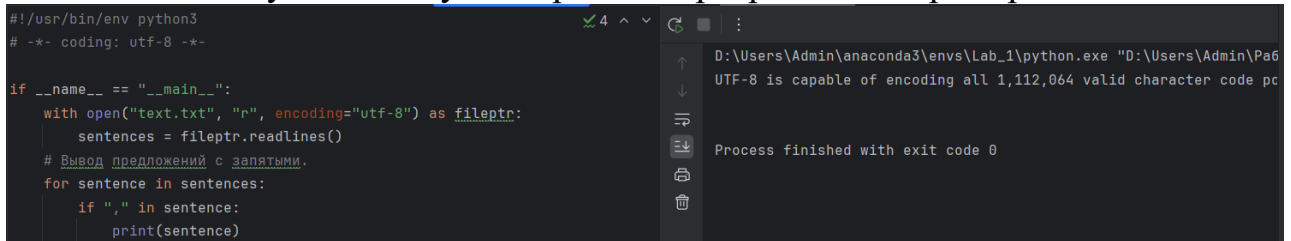


Рисунок 7. Результат работы программы из примера 7

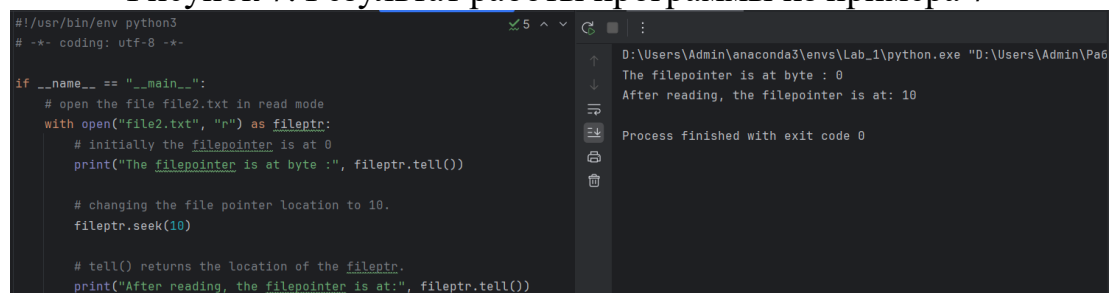


Рисунок 8. Результат работы программы из примера 8

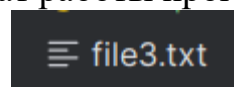


Рисунок 9. Результат работы программы из примера 9

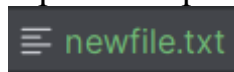


Рисунок 10. Результат работы программы из примера 10

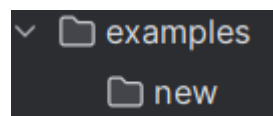


Рисунок 11. Результат работы программы из примера 11

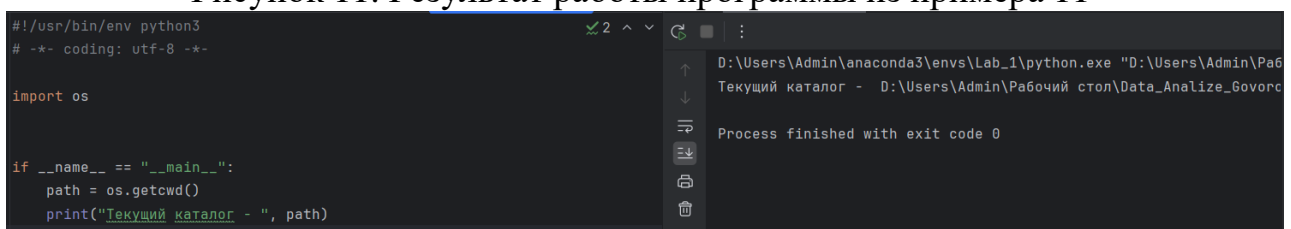


Рисунок 12. Результат работы программы из примера 12

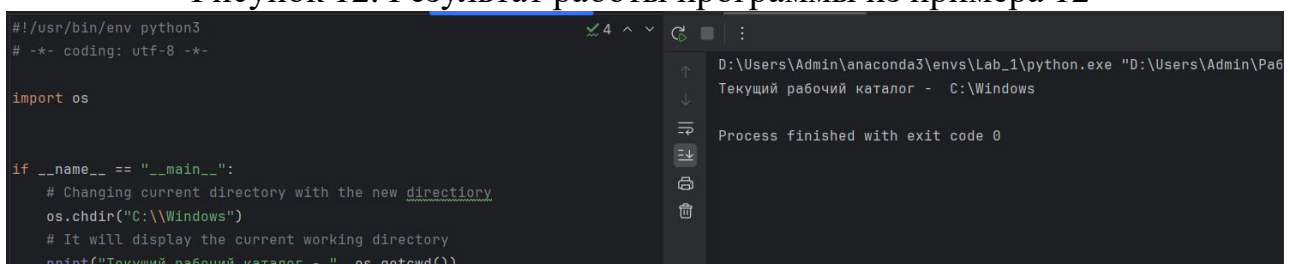


Рисунок 13. Результат работы программы из примера 13

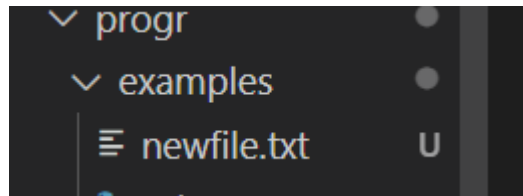


Рисунок 14. Результат работы программы из примера 14

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6
7  if __name__ == "__main__":
8      print("Number of arguments:", len(sys.argv), "arguments")
9      print("Argument List:", str(sys.argv))
```

D:\Users\Admin\anaconda3\envs\Lab_1\python.exe "D:\Users\Admin\Pa6
Number of arguments: 1 arguments
Argument List: ['D:\\Users\\Admin\\Рабочий стол\\Data_Analyze_Govc
Process finished with exit code 0

Рисунок 15. Результат работы программы из примера 15

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import sys

if __name__ == "__main__":
    for idx, arg in enumerate(sys.argv):
        print(f"Argument #{idx} is {arg}")
    print("No. of arguments passed is ", len(sys.argv))
```

D:\Users\Admin\anaconda3\envs\Lab_1\python.exe "D:\Users\Admin\Pa6
Argument #0 is D:\Users\Admin\Рабочий стол\Data_Analyze_Govorov-1\
No. of arguments passed is 1
Process finished with exit code 0

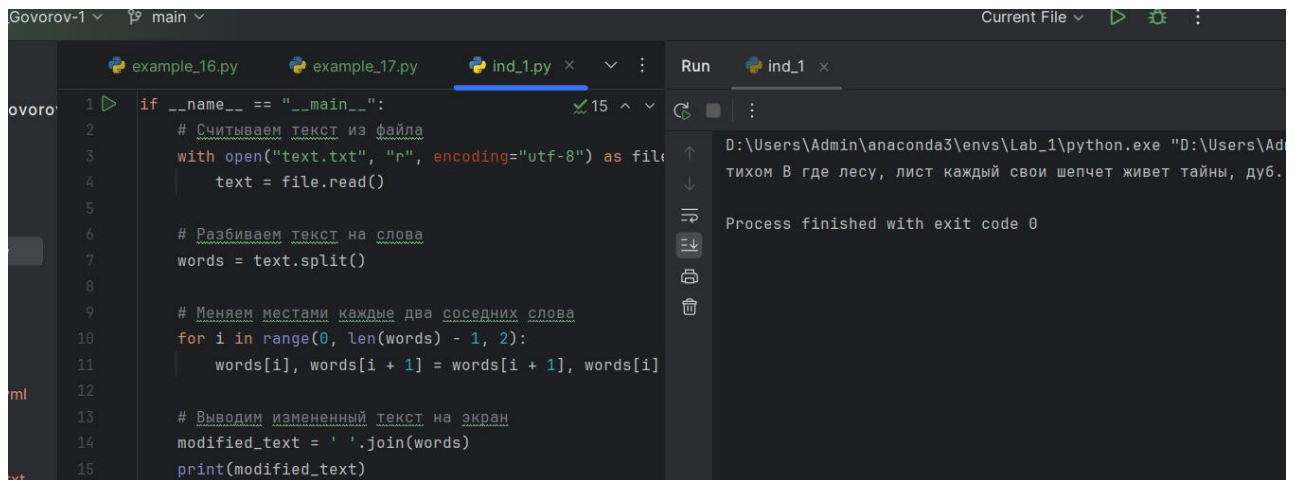
Рисунок 16. Результат работы программы из примера 16

```
Govorov-1  main
example_14.py  example_15.py  example_16.py  example_17.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import os
5  import secrets
6  import string
7  import sys
8
9
10 if __name__ == "__main__":
11     if len(sys.argv) != 2:
12         print("The password length is not given!", file=sys.stderr)
13         sys.exit(1)
14
15     chars = string.ascii_letters + string.punctuation + string.digits
16     length_pwd = int(sys.argv[1])
17
18     result = []
19     for _ in range(length_pwd):
20         idx = secrets.SystemRandom().randrange(len(chars))
21         result.append(chars[idx])
22
23     print(f"Secret Password: {''.join(result)}")
```

Рисунок 17. Результат работы программы из примера 17

5. Выполнил индивидуальные задания, согласно варианту 5. Привёл в отчете скриншоты работы программ.

Задание. Написать программу, которая считывает текст из файла и выводит его на экран, меняя местами каждые два соседних слова.

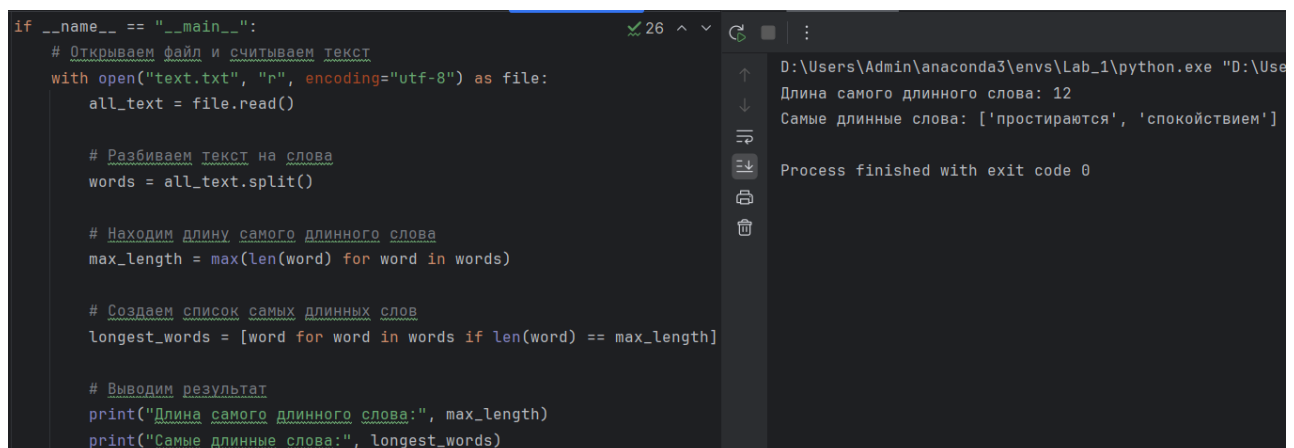


```
1 if __name__ == "__main__":
2     # Считываем текст из файла
3     with open("text.txt", "r", encoding="utf-8") as file:
4         text = file.read()
5
6     # Разбиваем текст на слова
7     words = text.split()
8
9     # Меняем местами каждые два соседних слова
10    for i in range(0, len(words) - 1, 2):
11        words[i], words[i + 1] = words[i + 1], words[i]
12
13    # Выводим измененный текст на экран
14    modified_text = ' '.join(words)
15    print(modified_text)
```

Process finished with exit code 0

Рисунок 18. Результат работы программы из индивидуального задания 1

Задание. В данном упражнении вы должны написать программу, которая будет находить самое длинное слово в файле. В качестве результата программа должна выводить на экран длину самого длинного слова и все слова такой длины. Для простоты принимайте за значимые буквы любые непробельные символы, включая цифры и знаки препинания.



```
if __name__ == "__main__":
    # Открываем файл и считываем текст
    with open("text.txt", "r", encoding="utf-8") as file:
        all_text = file.read()

    # Разбиваем текст на слова
    words = all_text.split()

    # Находим длину самого длинного слова
    max_length = max(len(word) for word in words)

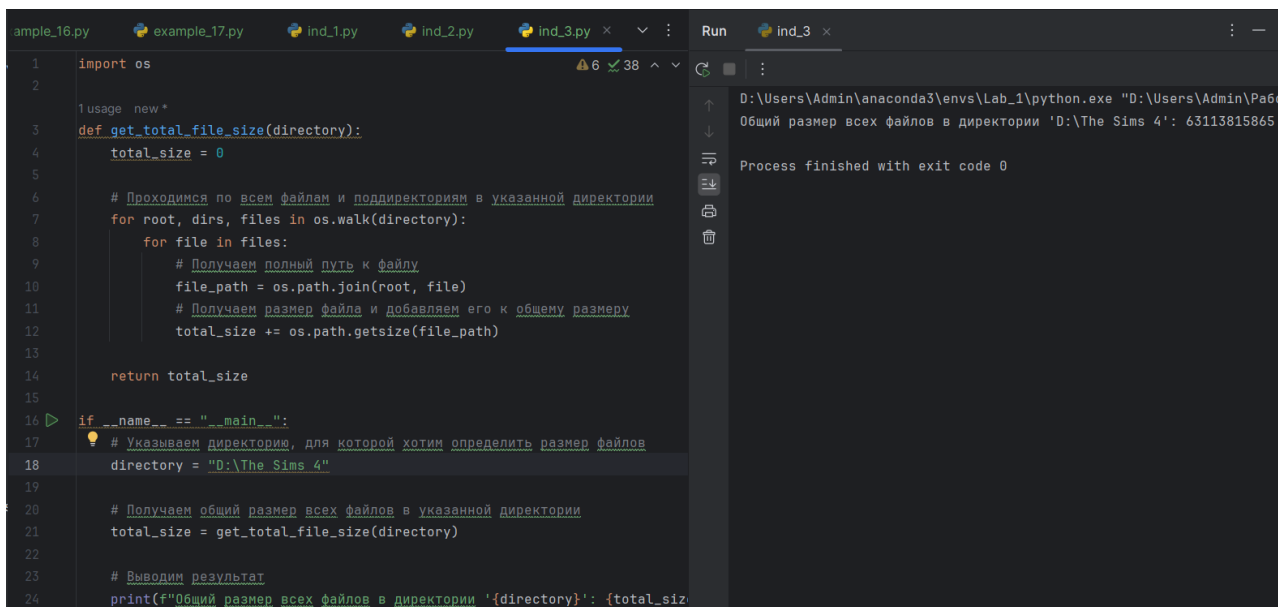
    # Создаем список самых длинных слов
    longest_words = [word for word in words if len(word) == max_length]

    # Выводим результат
    print("Длина самого длинного слова:", max_length)
    print("Самые длинные слова:", longest_words)
```

Process finished with exit code 0

Рисунок 19. Результат работы программы из индивидуального задания 2

Задача: Определение размера всех файлов в указанной директории и ее поддиректориях.



```
1 import os
2
3 usage new *
4 def get_total_file_size(directory):
5     total_size = 0
6     # Проходимся по всем файлам и поддиректориям в указанной директории
7     for root, dirs, files in os.walk(directory):
8         for file in files:
9             # Получаем полный путь к файлу
10            file_path = os.path.join(root, file)
11            # Получаем размер файла и добавляем его к общему размеру
12            total_size += os.path.getsize(file_path)
13
14    return total_size
15
16 if __name__ == "__main__":
17     # Указываем директорию, для которой хотим определить размер файлов
18     directory = "D:\The Sims 4"
19
20     # Получаем общий размер всех файлов в указанной директории
21     total_size = get_total_file_size(directory)
22
23     # Выводим результат
24     print(f"Общий размер всех файлов в директории '{directory}': {total_size}")
```

Run ind_3 x

D:\Users\Admin\anaconda3\envs\Lab_1\python.exe "D:\Users\Admin\Pa6c
Общий размер всех файлов в директории 'D:\The Sims 4': 63113815865

Process finished with exit code 0

Рисунок 20. Файл после запуска программы из работы 3

Контрольные вопросы

1. Как открыть файл в языке Python только для чтения?

В Python для открытия файла только для чтения используется функция `open()` с указанием режима доступа `r`.

2. Как открыть файл в языке Python только для записи?

Для открытия файла только для записи в Python используется функция `open()` с указанием режима доступа `'w'`. Если файл с указанным именем не существует, он будет создан. Если файл уже существует, его содержимое будет перезаписано.

3. Как прочитать данные из файла в языке Python?

Для чтения данных из файла в языке Python вы можете использовать встроенную функцию `open()` в сочетании с методами чтения, такими как `read()`, `readline()`, или `readlines()`.

4. Как записать данные в файл в языке Python?

В языке Python для записи данных в файл можно воспользоваться функцией `write()` объекта файла или методом `writelines()`.

5. Как закрыть файл в языке Python?

В языке Python закрыть файл можно с помощью метода `close()`. Этот метод следует вызывать после завершения всех операций с файлом, чтобы освободить ресурсы операционной системы. Однако, более предпочтительным способом управления файлом является использование

конструкции `with`, которая автоматически закрывает файл после завершения работы с ним.

6. Изучите самостоятельно работу конструкции `with ... as`. Каково ее назначение в языке Python? Где она может быть использована еще, помимо работы с файлами?

Конструкция `with ... as` в языке Python используется для создания контекстных менеджеров. Ее основное назначение - обеспечить правильное управление ресурсами в блоке кода, гарантируя выполнение определенных действий до и после выполнения блока кода.

В контексте работы с файлами, конструкция `with ... as` используется для автоматического закрытия файла после завершения работы с ним. Но помимо работы с файлами, она может быть использована в различных сценариях, где требуется управление ресурсами или выполнение каких-то действий до и после выполнения определенного блока кода.

Кроме работы с файлами, конструкцию `with ... as` можно использовать для работы с другими ресурсами, которые требуют явного закрытия, например, сетевыми соединениями или базами данных. При использовании конструкции `with ... as` с такими ресурсами, можно быть уверенным, что они будут корректно закрыты, даже при возникновении исключений или ошибок.

7. Изучите самостоятельно документацию Python по работе с файлами. Какие помимо рассмотренных существуют методы записи/чтения информации из файла?

`writelines()`: Этот метод используется для записи списка строк в файл. Он принимает список строк в качестве аргумента и записывает каждую строку в конец файла.

`readline()`: Этот метод используется для чтения одной строки из файла. При каждом вызове метода он возвращает следующую строку файла.

`readlines()`: Этот метод используется для чтения всех строк из файла и возвращает список строк, где каждая строка представляет собой элемент списка.

Метод `seek()` используется для изменения позиции указателя файла. Он позволяет переместить указатель на определенное смещение `offset`

относительно начала, текущей позиции или конца файла в зависимости от значения аргумента `whence`.

Аргументы `offset` и `whence` являются необязательными:

- `offset` - целочисленное значение, указывающее смещение в байтах.

Значение `offset` может быть положительным, отрицательным или нулем.

- `whence` - указывает, как интерпретировать аргумент `offset`. Доступные значения `whence`:

- 0 (по умолчанию): смещение относительно начала файла.

- 1: смещение относительно текущей позиции.

- 2: смещение относительно конца файла.

`tell()`: Метод `tell()` используется для получения текущей позиции указателя файла. Он возвращает целочисленное значение, представляющее текущую позицию указателя в байтах от начала файла.

`truncate([size])`: Метод `truncate()` используется для изменения размера файла до указанного размера `size`. Если аргумент `size` не указан, то размер файла обрезается до текущей позиции указателя. Метод возвращает новый размер файла после изменения.

8. Какие существуют, помимо рассмотренных, функции модуля `os` для работы с файловой системой?

Функция `listdir()` возвращает список файлов и директорий в указанном пути `path`. Она принимает один аргумент - путь к директории. Возвращается список строк, представляющих имена файлов и директорий в указанной директории.

Функция `makedirs()` создает все директории в указанном пути `path`. Если указан аргумент `mode`, то устанавливает права доступа для всех созданных директорий в соответствии с указанным значением `mode`.

Функция `rmdir()` используется для удаления директории по указанному пути `path` и всех пустых поддиректорий. Если какая-либо поддиректория не является пустой, будет вызвано исключение `OSError`.

Функция `join()` объединяет один или несколько компонентов пути в один путь. Она автоматически добавляет разделитель пути, соответствующий операционной системе.

Функция `exists()` проверяет, существует ли файл или директория по указанному пути `path`. Возвращает `True`, если файл или директория существуют, и `False` в противном случае.

Функция `isdir()` проверяет, является ли заданный путь директорией. Возвращает `True`, если путь является директорией, и `False` в противном случае.