

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ  
Федеральное государственное автономное  
образовательное учреждение высшего образования  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций

Отчет по лабораторной работе № 2.15  
по дисциплине «Основы программной инженерии»

Выполнил студент группы ПИЖ-б-о-22-1

Душин Александр Владимирович.

Подпись студента \_\_\_\_\_

Работа защищена « » \_\_\_\_\_ 20\_\_ г.

Проверил Воронкин Р.А. \_\_\_\_\_  
(подпись)

Ставрополь 2024

Тема: Работа с файлами в языке Python.

Цель работы: 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 выполнения работы:


1. Создать общедоступный репозиторий на GitHub с использованием лицензии MIT и язык программирования Python:

## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (\*).

Owner \*

 MrPlatynum ▾

Repository name \*

/ ProgrammEngineering2.1! ▾

✔ ProgrammEngineering2.15 is available.

Great repository names are short and memorable. Need inspiration? How about [automatic-lamp](#) ?

Description (optional)



Public

Anyone on the internet can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

Initialize this repository with:



Add a README file

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: Python ▾

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: MIT License ▾

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

 You are creating a public repository in your personal account.

Create repository

Рисунок 1 – Создание общедоступного репозитория на GitHub с заданными настройками

```
Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents
$ git clone https://github.com/MrPlatynum/ProgrammEngineering18.git
Cloning into 'ProgrammEngineering18'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.
```

Рисунок 2 – Клонирование созданного репозитория на локальный компьютер

```
ENV/
env.bak/
venv.bak/

# Spyder project settings
.spyderproject
.spyproject

# Rope project settings
.ropeproject

# mkdocs documentation
/site

# mypy
.mypy_cache/
.dmypy.json
dmypy.json

# Pyre type checker
.pyre/

# pytype static type analyzer
.pytype/

# Cython debug symbols
cython_debug/

# PyCharm
# JetBrains specific template is maintained in a separate JetBrains.gitignore that can
# be found at https://github.com/github/gitignore/blob/main/Global/JetBrains.gitignore
# and can be added to the global gitignore or merged into this file. For a more nuclear
# option (not recommended) you can uncomment the following to ignore the entire idea folder.
.idea/
```

Рисунок 3 – файл .gitignore

```
Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering18 (main)
$ git checkout -b develop
Switched to a new branch 'develop'

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering18 (develop)
$
```

Рисунок 4 – организация репозитория в соответствии с моделью ветвления git flow

2. Проработайте примеры лабораторной работы:

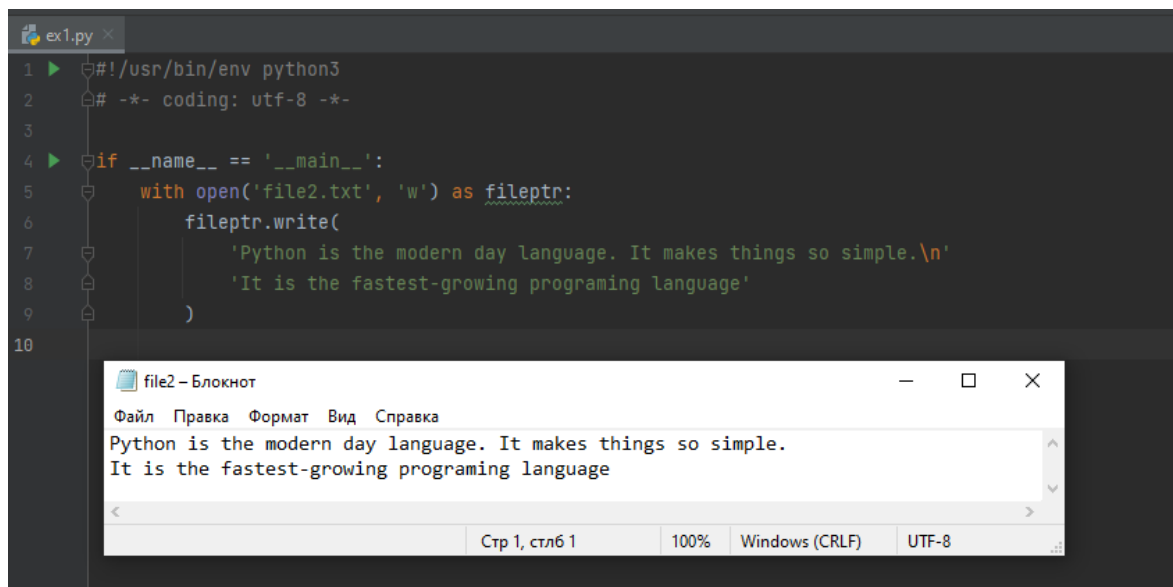


Рисунок 5 – Запись текста в файл file2.txt

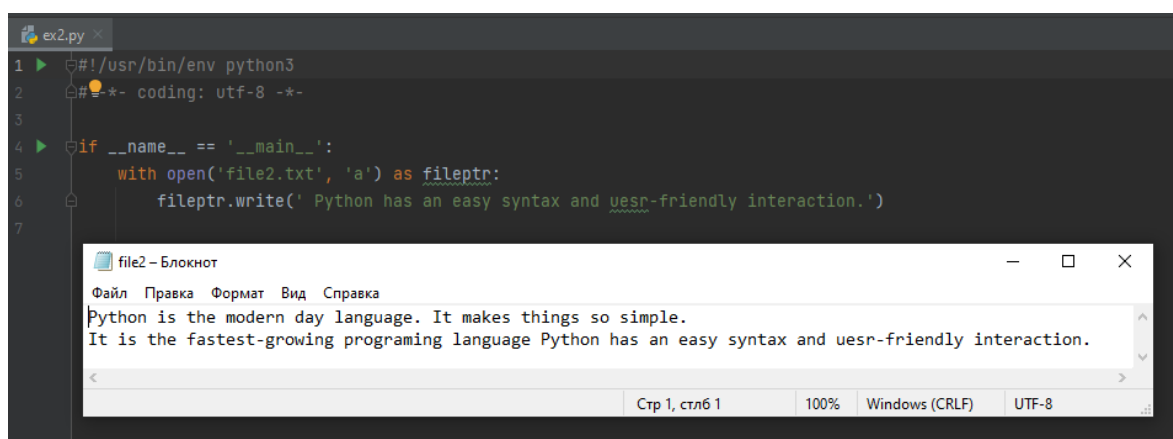


Рисунок 6 – Запись другого текста в файл file2.txt

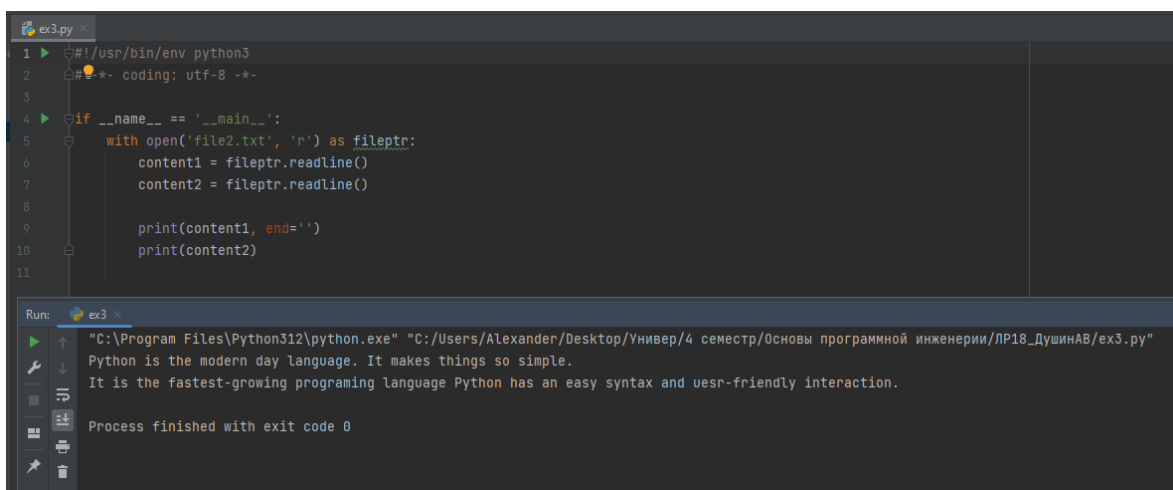


Рисунок 7 – Чтение первых двух строк файла

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 if __name__ == '__main__':
5     with open('file2.txt', 'r') as fileptr:
6         content = fileptr.readlines()
7         print(content)
8
```

Run: ex4

"C:\Program Files\Python312\python.exe" "C:/Users/Alexander/Desktop/Универ/4 семестр/Основы программной инженерии/ЛР18\_ДушинАВ/ex4.py"

['Python is the modern day language. It makes things so simple.\n', 'It is the fastest-growing programing language Python has an easy syntax and uesr-friendly interaction.']

Process finished with exit code 0

Рисунок 8 – Чтение всех строк файла с помощью метода readlines()

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 if __name__ == '__main__':
5     with open('newfile.txt', 'x') as fileptr:
6         print(fileptr)
7
8     if fileptr:
9         print('File create successfully')
10
```

Run: ex5

"C:\Program Files\Python312\python.exe" "C:/Users/Alexander/Desktop/Универ/4 семестр/Основы программной инженерии/ЛР18\_ДушинАВ/ex5.py"

<\_io.TextIOWrapper name='newfile.txt' mode='x' encoding='cp1251'>

File create successfully

Рисунок 9 – Создание нового файла с помощью режима доступа x

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 if __name__ == '__main__':
5     with open('text.txt', 'w', encoding='utf-8') as fileptr:
6         print('UTF-8 is a variable-width character encoding used for electronic communication', file=fileptr)
7         print('UTF-8 is capable of encoding all 1,112,064 valid character code points.', file=fileptr)
8         print('In Unicode using one to four one-byte (8-bit) code units.', file=fileptr)
9
```

text – Блокнот

Файл Правка Формат Вид Справка

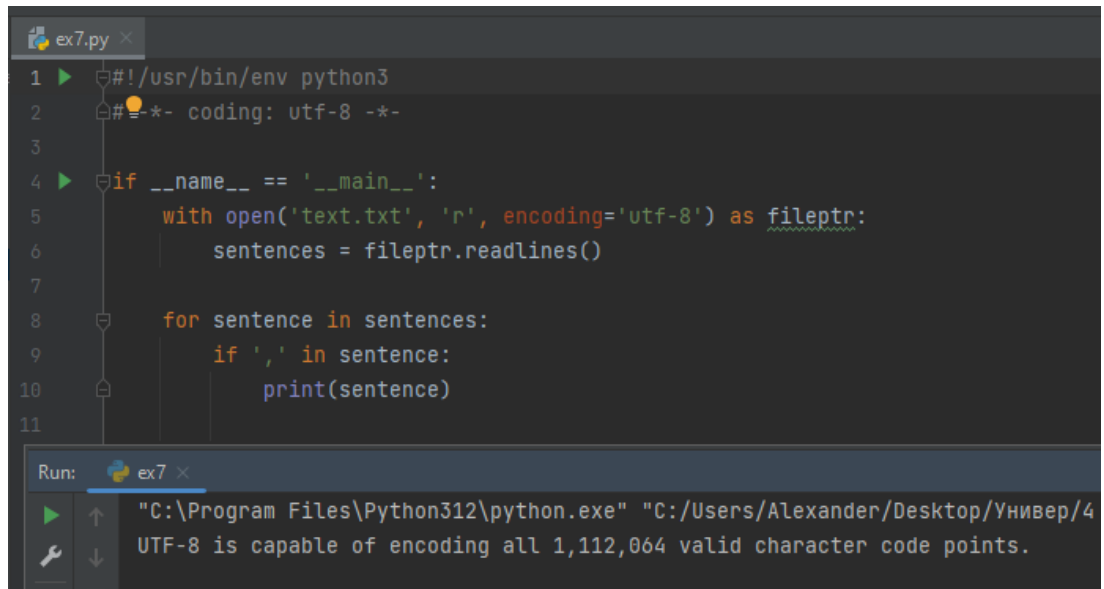
UTF-8 is a variable-width character encoding used for electronic communication

UTF-8 is capable of encoding all 1,112,064 valid character code points.

In Unicode using one to four one-byte (8-bit) code units.

Стр 1, столб 1 100% Windows (CRLF) UTF-8

Рисунок 10 – Создание и запись в файл text.txt с использованием кодировки UTF-8



```
ex7.py
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 if __name__ == '__main__':
5     with open('text.txt', 'r', encoding='utf-8') as fileptr:
6         sentences = fileptr.readlines()
7
8         for sentence in sentences:
9             if ',' in sentence:
10                 print(sentence)
11
```

Run: ex7

"C:\Program Files\Python312\python.exe" "C:/Users/Alexander/Desktop/Универ/4  
UTF-8 is capable of encoding all 1,112,064 valid character code points.

Рисунок 11 – Вывод предложений, в которых содержится запятая



```
ex8.py
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 if __name__ == '__main__':
5     with open('file2.txt', 'r') as fileptr:
6         print('The filepointer is at byte: ', fileptr.tell())
7
8         fileptr.seek(10)
9
10        print('After reading, the filepointer is at: ', fileptr.tell())
11
```

Run: ex8

"C:\Program Files\Python312\python.exe" "C:/Users/Alexander/Desktop/Универ  
The filepointer is at byte: 0  
After reading, the filepointer is at: 10

Рисунок 12 – Смещение указателя файла на 10 байт

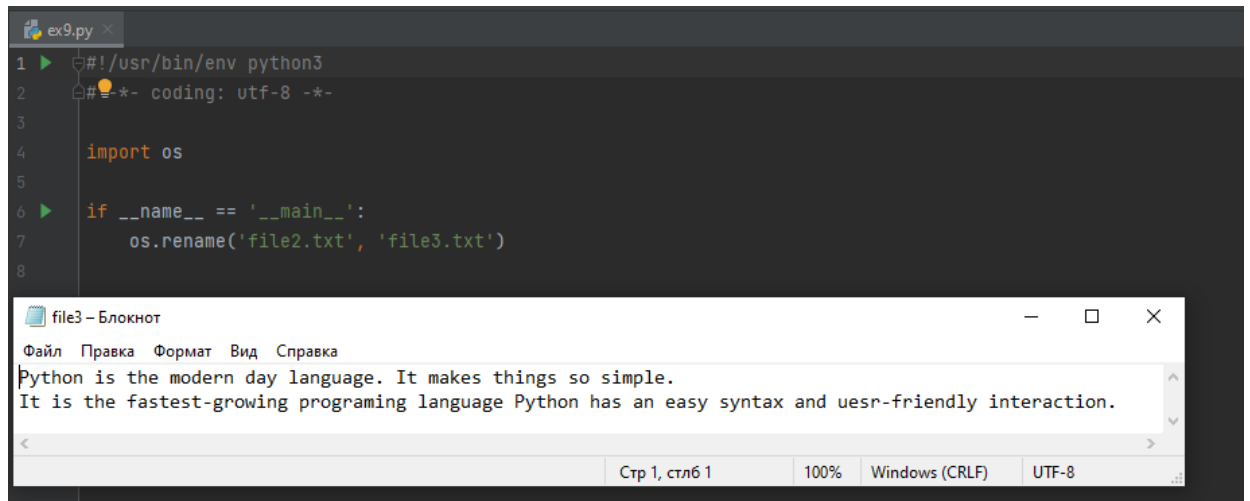


Рисунок 13 – Переименование файла и результат

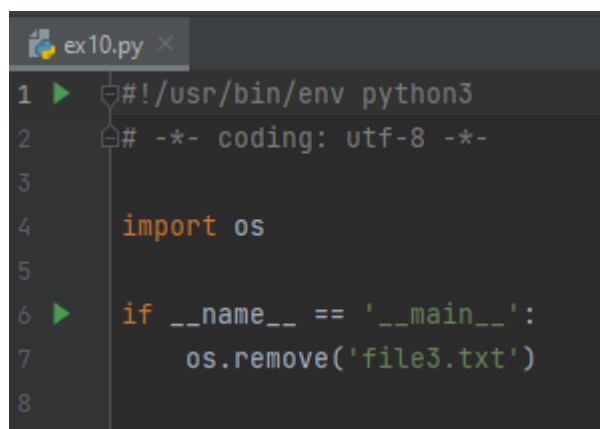


Рисунок 14 – Удаление файла

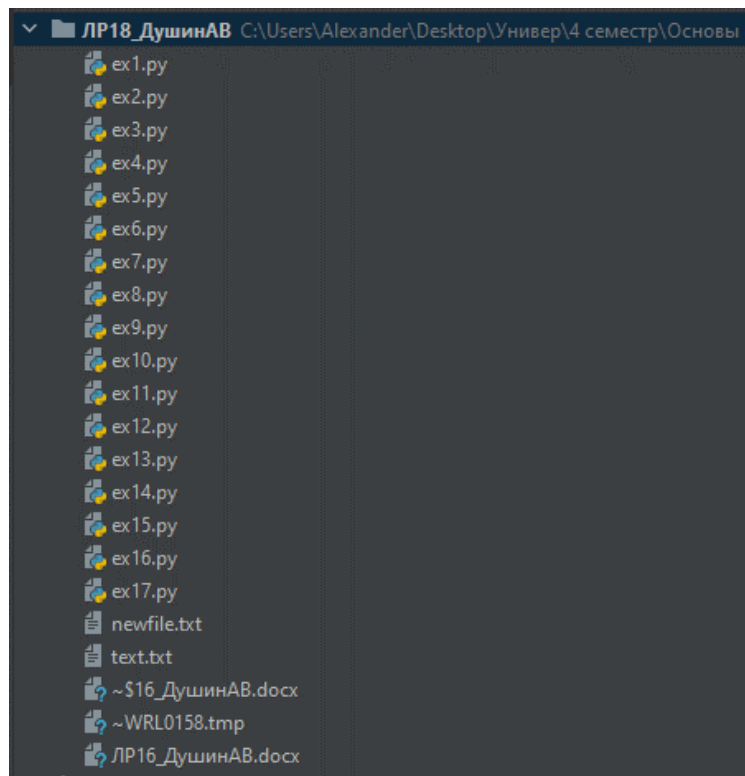
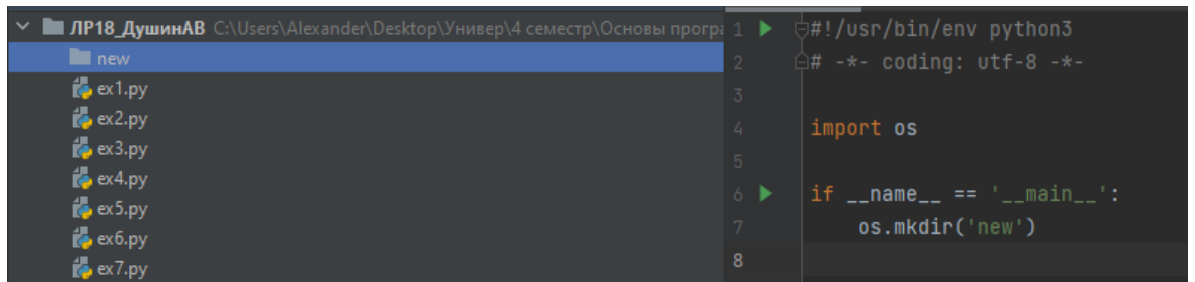
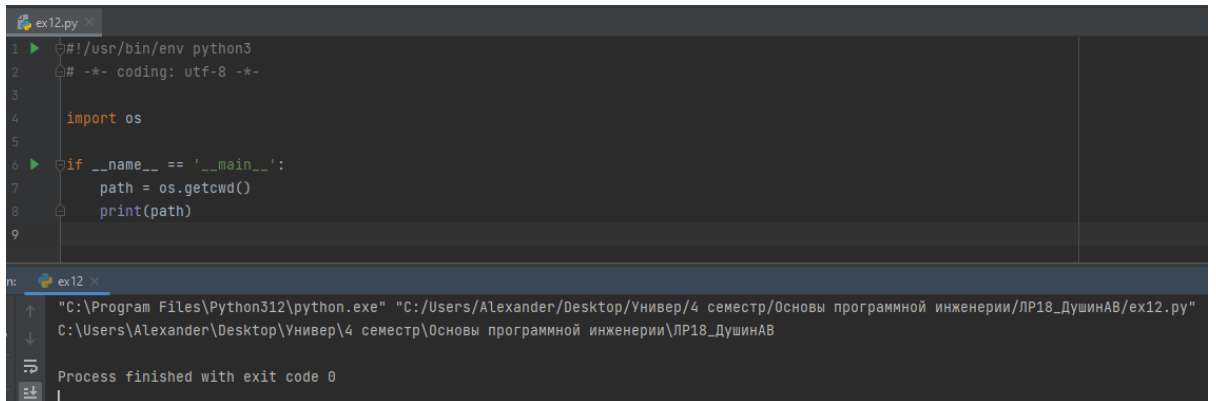


Рисунок 15 – Каталог после удаления файла



```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import os
5
6 if __name__ == '__main__':
7     os.mkdir('new')
```

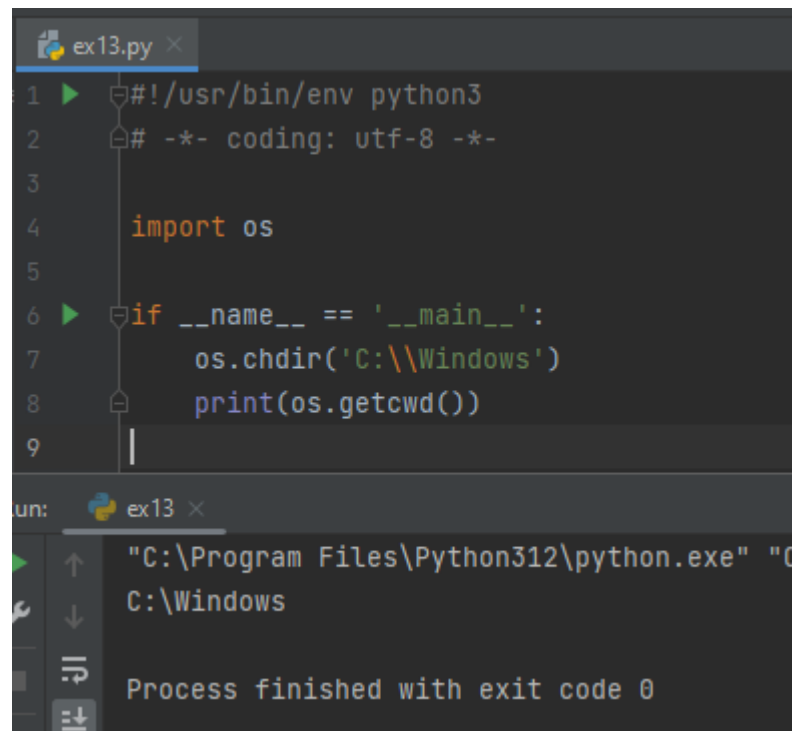
Рисунок 16 – Создание нового каталога



```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import os
5
6 if __name__ == '__main__':
7     path = os.getcwd()
8     print(path)
```

"C:\Program Files\Python312\python.exe" "C:/Users/Alexander/Desktop/Универ/4 семестр/Основы программной инженерии/ЛР18\_ДушинАВ/ex12.py"  
C:\Users\Alexander\Desktop\Универ\4 семестр\Основы программной инженерии\ЛР18\_ДушинАВ  
Process finished with exit code 0

Рисунок 17 – Вывод пути текущего каталога



```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import os
5
6 if __name__ == '__main__':
7     os.chdir('C:\\Windows')
8     print(os.getcwd())
```

un: ex13 x  
"C:\Program Files\Python312\python.exe" "C:/Users/Alexander/Desktop/Универ/4 семестр/Основы программной инженерии/ЛР18\_ДушинАВ/ex13.py"  
C:\Windows  
Process finished with exit code 0

Рисунок 18 – Изменение текущего рабочего каталога



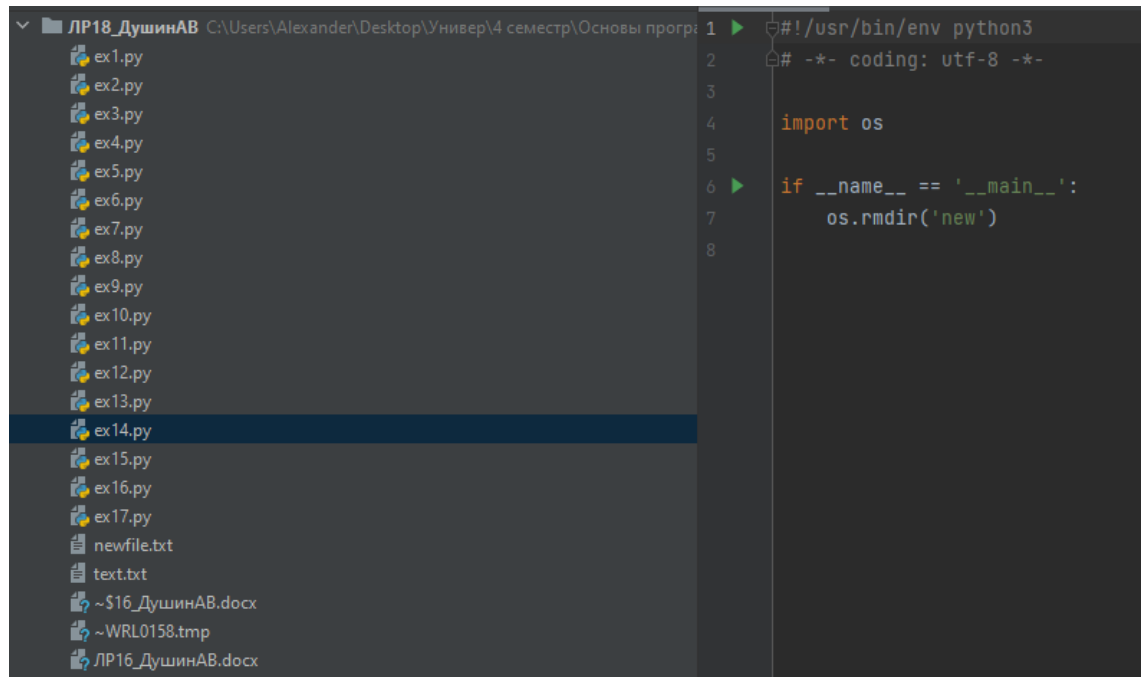


Рисунок 19 – Удаление каталога «new»

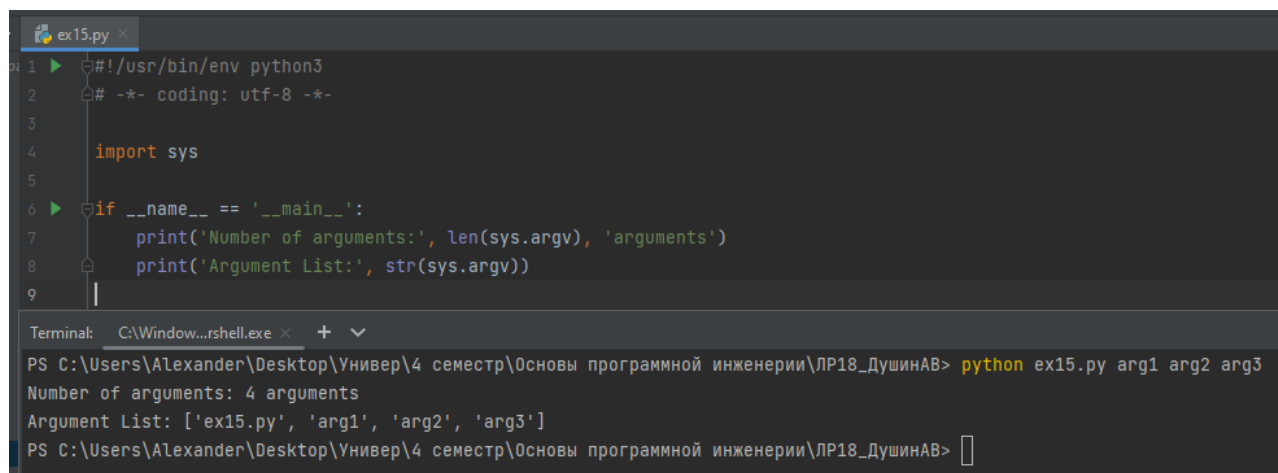


Рисунок 20 – Вывод списка аргументов и его длины

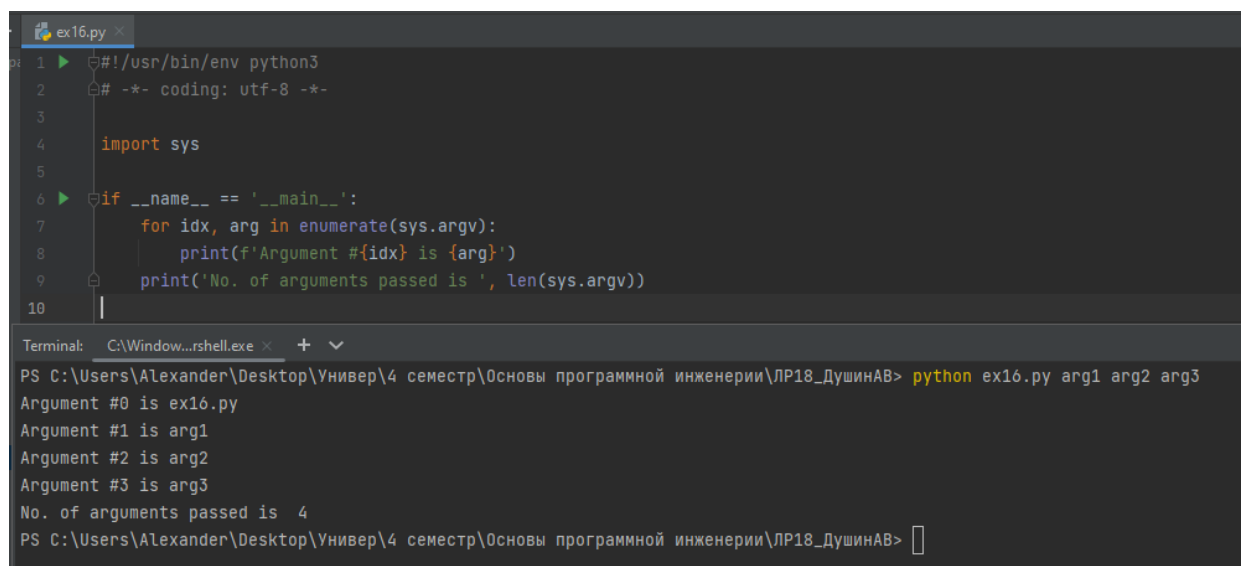
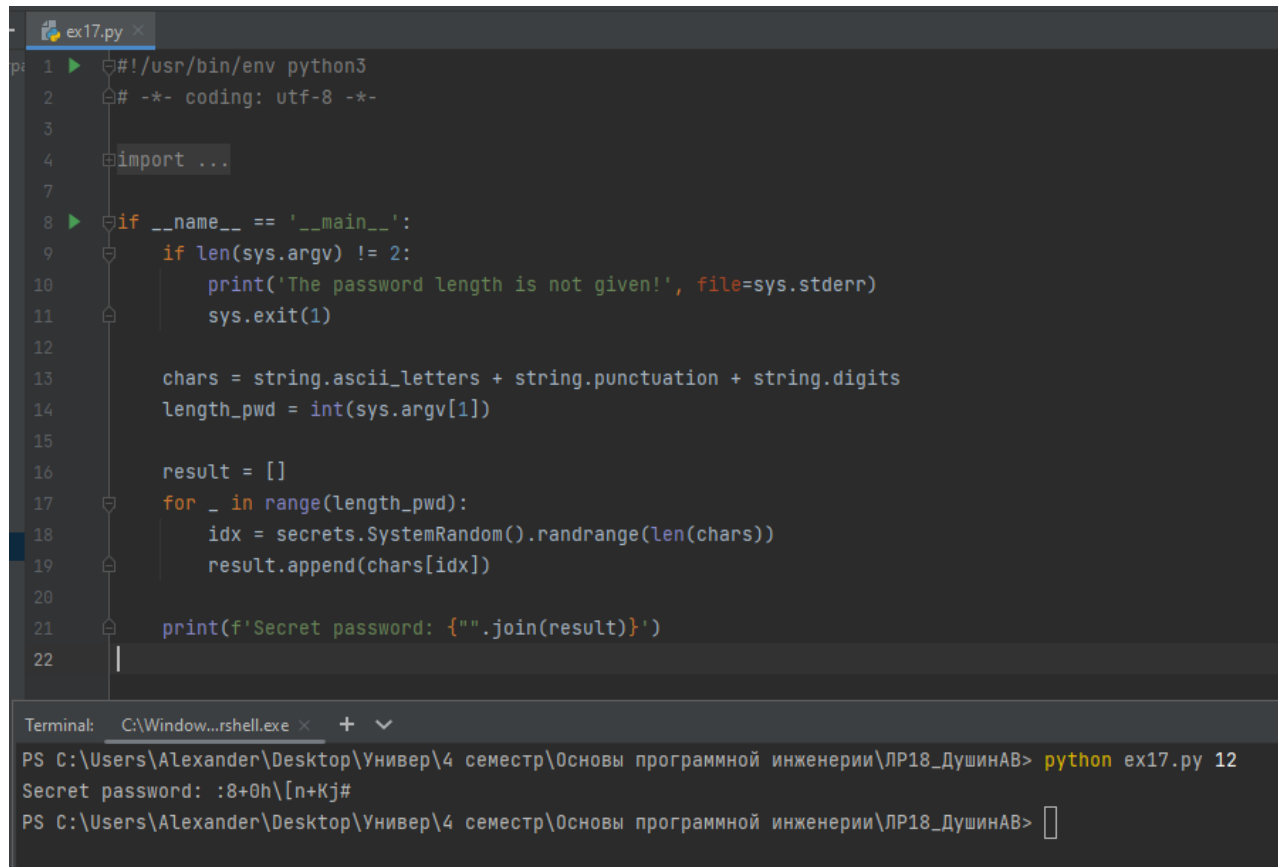


Рисунок 21 – Вывод аргументов и их позиций в списке

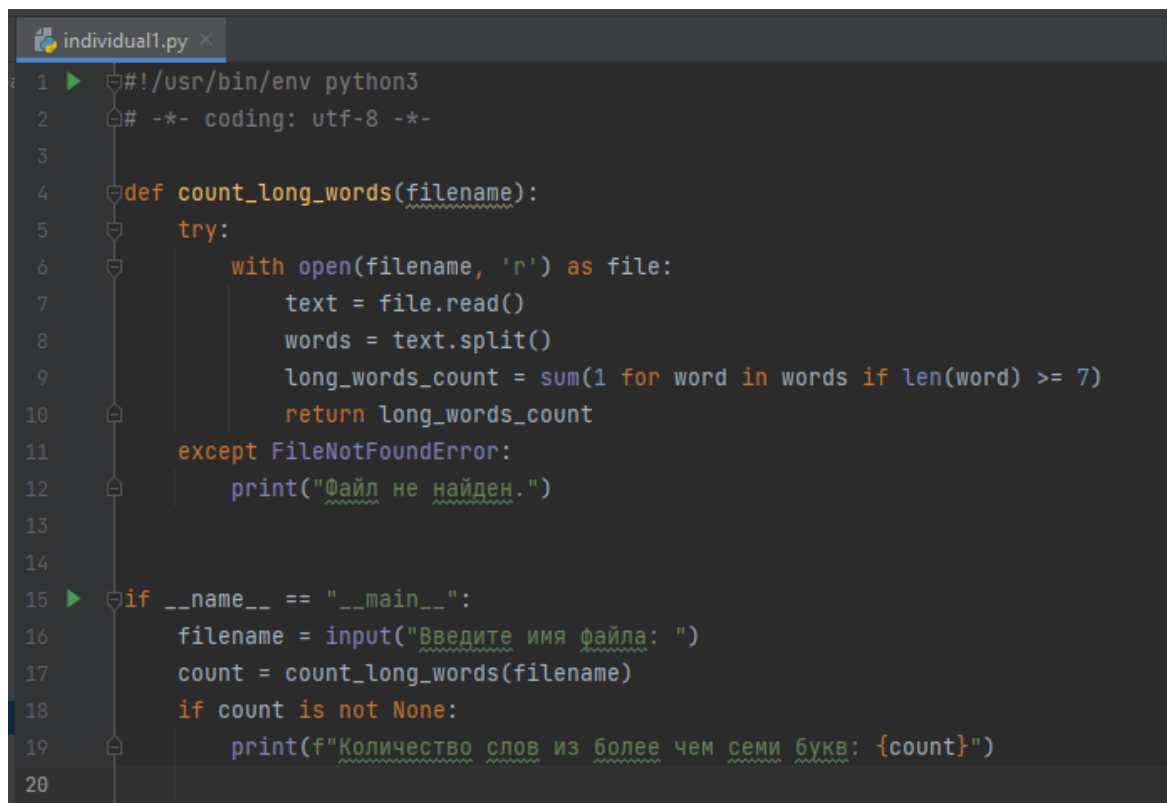


```
ex17.py x
1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3
4      import ...
5
6
7
8  ▶  if __name__ == '__main__':
9      if len(sys.argv) != 2:
10         print('The password length is not given!', file=sys.stderr)
11         sys.exit(1)
12
13         chars = string.ascii_letters + string.punctuation + string.digits
14         length_pwd = int(sys.argv[1])
15
16         result = []
17         for _ in range(length_pwd):
18             idx = secrets.SystemRandom().randrange(len(chars))
19             result.append(chars[idx])
20
21         print(f'Secret password: {"".join(result)}')
22
Terminal: C:\Window...rshell.exe x + v
PS C:\Users\Alexander\Desktop\Универ\4 семестр\Основы программной инженерии\ЛР18_ДушинАВ> python ex17.py 12
Secret password: :8+0h\[n+Kj#
PS C:\Users\Alexander\Desktop\Универ\4 семестр\Основы программной инженерии\ЛР18_ДушинАВ> 
```

Рисунок 22 – Генерация пароля заданной длины

### 3. Выполнить индивидуальные задания (вариант 7):

Задание 1. Написать программу, которая считывает текст из файла и определяет, сколько в нем слов, состоящих из не менее чем семи букв.

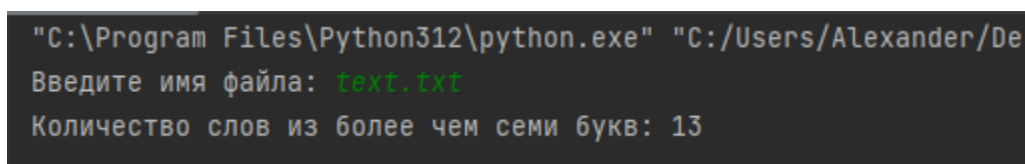


```

1  #!/usr/bin/env python3
2  #- coding: utf-8 -*-
3
4  def count_long_words(filename):
5      try:
6          with open(filename, 'r') as file:
7              text = file.read()
8              words = text.split()
9              long_words_count = sum(1 for word in words if len(word) >= 7)
10             return long_words_count
11     except FileNotFoundError:
12         print("Файл не найден.")
13
14
15  if __name__ == "__main__":
16     filename = input("Введите имя файла: ")
17     count = count_long_words(filename)
18     if count is not None:
19         print(f"Количество слов из более чем семи букв: {count}")
20

```

Рисунок 23 – Индивидуальное задание №1



```

"C:\Program Files\Python312\python.exe" "C:/Users/Alexander/De
Введите имя файла: text.txt
Количество слов из более чем семи букв: 13

```

Рисунок 24 – Вывод программы

Задание 2. Разработайте программу, которая будет показывать слово (или слова), чаще остальных встречающееся в текстовом файле. Сначала пользователь должен ввести имя файла для обработки. После этого вы должны открыть файл и проанализировать его построчно, разделив при этом строки по словам и исключив из них пробелы и знаки препинания. Также при подсчете частоты появления слов в файле вам стоит игнорировать регистры.

```
individual2.py x
1  ▶  #!/usr/bin/env python3
2    # -*- coding: utf-8 -*-
3
4    from collections import Counter
5    import string
6
7
8    def clean_text(text):
9        for punctuation in string.punctuation:
10            text = text.replace(punctuation, "")
11        words = text.lower().split()
12        return words
13
14
15    def most_common_words(filename):
16        try:
17            with open(filename, 'r') as file:
18                text = file.read()
19                words = clean_text(text)
20                word_counts = Counter(words)
21                most_common = word_counts.most_common(1)
22            return most_common
23        except FileNotFoundError:
24            print("Файл не найден.")
25
26
27    ▶ if __name__ == "__main__":
28        filename = input("Введите имя файла: ")
29        most_common_word = most_common_words(filename)
30        if most_common_word:
31            print(f"Самое часто встречающееся слово: {most_common_word[0][0]}")
32        else:
33            print("В файле нет текста.")
34
```

Рисунок 25 – Индивидуальное задание №2

```
individual2.py x  text.txt x
1  UTF-8 is a variable-width character encoding used fo
2  UTF-8 is capable of encoding all 1,112,064 valid cha
3  In Unicode using one to four one-byte (8-bit) code u
4
Run:  individual2 x
▶  "C:\Program Files\Python312\python.exe" "C:/Users/A
Введите имя файла: text.txt
Самое часто встречающееся слово: utf8
```

Рисунок 26 – Вывод программы

4. Зафиксируем сделанные изменения, сольем ветки и отправим на удаленный репозиторий:

```
Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering18 (develop)
$ git log --oneline
cceb74 (HEAD -> develop) Upload SRC
42de0dc (origin/main, origin/HEAD, main) Initial commit
```

Рисунок 27 – Коммиты ветки develop во время выполнения лабораторной работы

```
Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering18 (develop)
$ git checkout main
Switched to branch 'main'
M .gitignore
Your branch is up to date with 'origin/main'.

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering18 (main)
$ git merge develop
Updating 42de0dc..cceb74
Fast-forward
 ex1.py      | 9 ++++++
 ex10.py     | 7 ++++++
 ex11.py     | 7 ++++++
 ex12.py     | 8 ++++++
 ex13.py     | 8 ++++++
 ex14.py     | 7 ++++++
 ex15.py     | 8 ++++++
 ex16.py     | 9 ++++++
 ex17.py     | 21 ++++++
 ex2.py      | 6 ++++++
 ex3.py      | 10 ++++++
 ex4.py      | 7 ++++++
 ex5.py      | 9 ++++++
 ex6.py      | 8 ++++++
 ex7.py      | 10 ++++++
 ex8.py      | 10 ++++++
 ex9.py      | 7 ++++++
 individual1.py | 19 ++++++
 individual2.py | 33 ++++++
 newfile.txt  | 0
 text.txt    | 3 +++
21 files changed, 206 insertions(+)
 create mode 100644 ex1.py
 create mode 100644 ex10.py
 create mode 100644 ex11.py
 create mode 100644 ex12.py
 create mode 100644 ex13.py
 create mode 100644 ex14.py
 create mode 100644 ex15.py
 create mode 100644 ex16.py
 create mode 100644 ex17.py
 create mode 100644 ex2.py
 create mode 100644 ex3.py
 create mode 100644 ex4.py
 create mode 100644 ex5.py
 create mode 100644 ex6.py
 create mode 100644 ex7.py
 create mode 100644 ex8.py
 create mode 100644 ex9.py
 create mode 100644 individual1.py
 create mode 100644 individual2.py
 create mode 100644 newfile.txt
 create mode 100644 text.txt

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering18 (main)
$ |
```

Рисунок 28 – Слияние ветки develop в ветку main

```
Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering18 (main)
$ git push origin main
Enumerating objects: 24, done.
Counting objects: 100% (24/24), done.
Delta compression using up to 12 threads
Compressing objects: 100% (22/22), done.
Writing objects: 100% (23/23), 4.18 KiB | 1.39 MiB/s, done.
Total 23 (delta 5), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (5/5), done.
To https://github.com/MrPlatinum/ProgrammEngineering18.git
    42de0dc..cceb74  main -> main

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering18 (main)
$
```

Рисунок 29 – Отправка на удаленный репозиторий

Ответы на контрольные вопросы:

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

Чтобы открыть файл в языке Python только для чтения, используйте функцию `open()` с режимом доступа `'r'`, `'rb'`. Например: `open('file.txt', 'r')`.

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

Для открытия файла только для записи в Python используйте функцию `open()` с режимом доступа `'w'`, `'wb'`. Например: `open('file.txt', 'w')`.

3. Как прочитать данные из файла в языке Python?

Для чтения данных из файла в Python можно использовать методы `read()`, `readline()` или `readlines()` объекта файла, который возвращается функцией `open()`.

4. Как записать данные в файл в языке Python?

Запись данных в файл в Python можно осуществить с помощью метода `write()` объекта файла, который возвращается функцией `open()`.

5. Как закрыть файл в языке Python?

Чтобы закрыть файл в Python, используйте метод `close()` объекта файла.

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 Python используется для создания контекстного менеджера, который автоматически управляет ресурсами (например, файлами), освобождая их после использования. Она может быть также использована, например, для работы с сетевыми соединениями или базами данных.

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

1) `file.read(size)`: Читает определенное количество байтов из файла. Если аргумент `size` не указан, читает весь файл.

2) `file.readline()`: Читает одну строку из файла.

3) `file.readlines()`: Читает все строки из файла и возвращает список строк.

4) `file.write(string)`: Записывает строку в файл.

5) `file.writelines(lines)`: Записывает список строк в файл.

6) `file.seek(offset, whence)`: Изменяет текущую позицию в файле на указанное смещение. Аргумент `whence` определяет базовую позицию для смещения (0 - начало файла, 1 - текущая позиция, 2 - конец файла).

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

- `os.path`: Этот модуль предоставляет функции для работы с путями к файлам и директориям, такие как `os.path.exists()`, `os.path.abspath()`, `os.path.dirname()`, `os.path.basename()`, `os.path.join()` и другие.

- `os.listdir(path)`: Возвращает список файлов и директорий в указанной директории.

- `os.getcwd()`: Возвращает текущий рабочий каталог.

- `os.chdir(path)`: Изменяет текущий рабочий каталог.

- `os.path.isfile(path)`: Проверяет, является ли путь к файлу.

- `os.path.isdir(path)`: Проверяет, является ли путь к директории.



