

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 №2.15
дисциплины
«Программирование на языке Python»

Выполнил:
Быковская Стефания Станиславовна
2 курс, группа ИТС-б-о-22-1,
11.03.02 «Инфокоммуникационные
технологии и системы связи», очная
форма обучения

(подпись)

Проверил:

Воронкин Р. А., доцент кафедры
инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

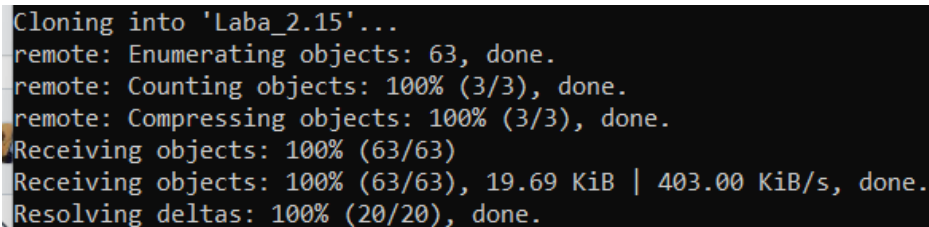
Ставрополь, 2023 г.

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:

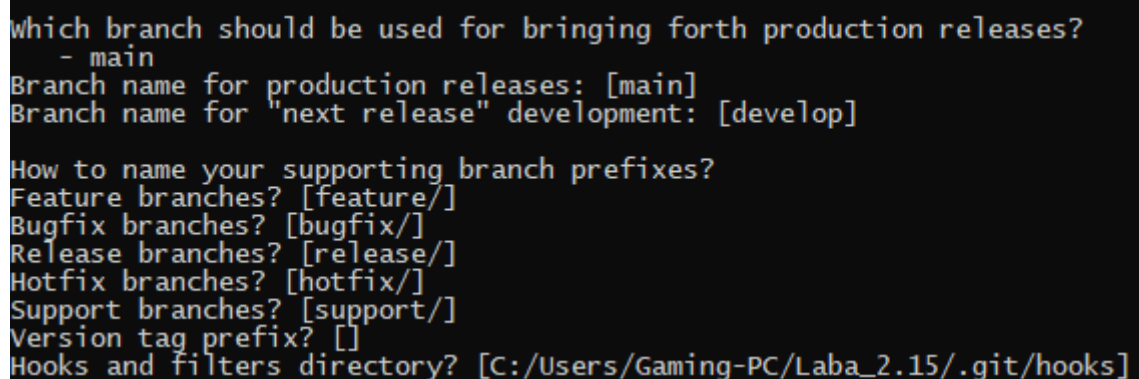
Задание 1. Создала общедоступный репозиторий на GitHub, в котором использована лицензия MIT и язык программирования Python, также добавила файл .gitignore с необходимыми правилами. Клонировала свой репозиторий на свой компьютер.



```
Cloning into 'Laba_2.15'...
remote: Enumerating objects: 63, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (3/3), done.
Receiving objects: 100% (63/63)
Receiving objects: 100% (63/63), 19.69 KiB | 403.00 KiB/s, done.
Resolving deltas: 100% (20/20), done.
```

Рисунок 1. Клонирование репозитория

Задание 2. Организовала свой репозиторий в соответствии с моделью ветвления git-flow, появилась новая ветка develop в которой буду выполнять дальнейшие задачи.



```
which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [C:/Users/Gaming-PC/Laba_2.15/.git/hooks]
```

Рисунок 2. Модель ветвления git-flow

Задание 3. Создала виртуальное окружение conda и активировала его, также установила необходимые пакеты isort, black, flake8.

```
(base) PS C:\Users\Gaming-PC\Laba_2.15> conda activate 2.15
(2.15) PS C:\Users\Gaming-PC\Laba_2.15> conda install -c conda-forge isort
Collecting package metadata (current_repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
  current version: 23.1.0
  latest version: 23.9.0

Please update conda by running

  $ conda update -n base -c defaults conda

Or to minimize the number of packages updated during conda update use

  conda install conda=23.9.0
```

Рисунок 3. Создание виртуального окружения и установка пакетов

Задание 4. Создаем проект PyCharm в папке репозитория. Приступила к работе с примером. Добавила новый файл Пример 1.py.

Условие примера: файл file2.txt не существует, необходимо создать программу которая будет создавать новый файл и записывать его содержимое с помощью функции write() .

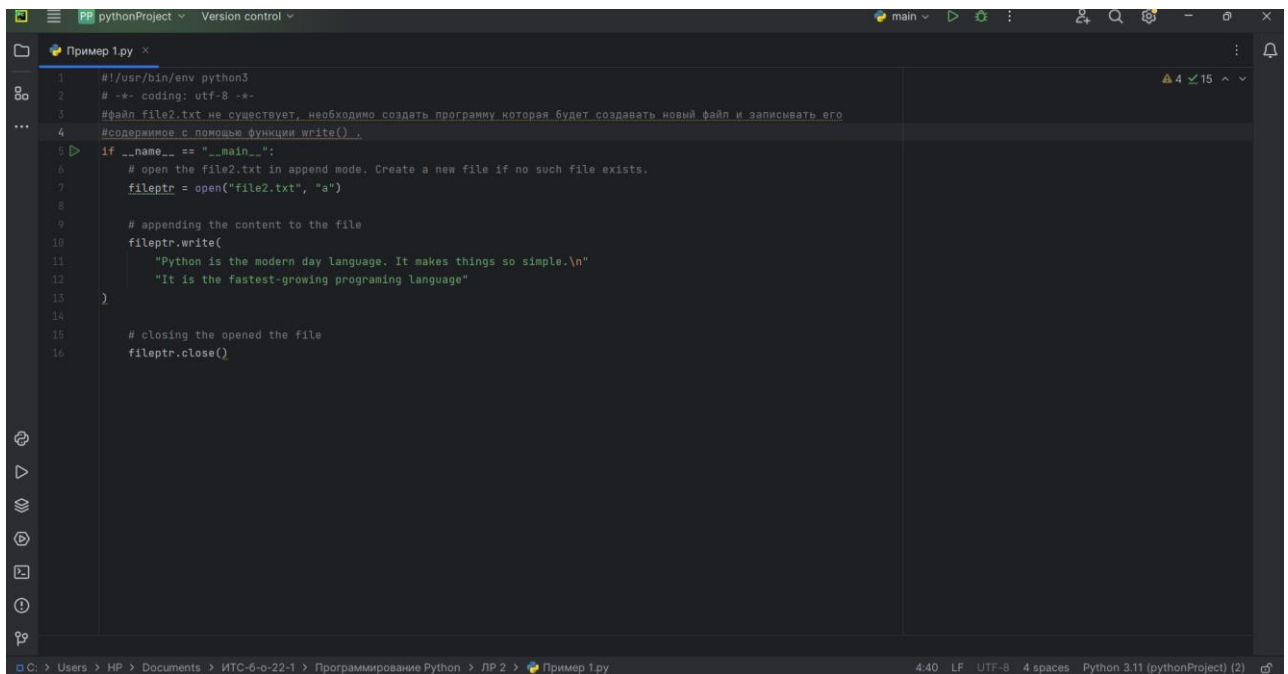


Рисунок 4. Реализация примера 1

file2 – Блокнот

Файл Правка Формат Вид Справка

```
Python is the modern day language. It makes things so simple.
It is the fastest-growing programming language
```

Рисунок 5. Результат примера 1

Создала новый файл под названием Пример2.ру.

Условие примера: необходимо открыть файл в режиме и добавить содержимое в существующий файл file2.txt.

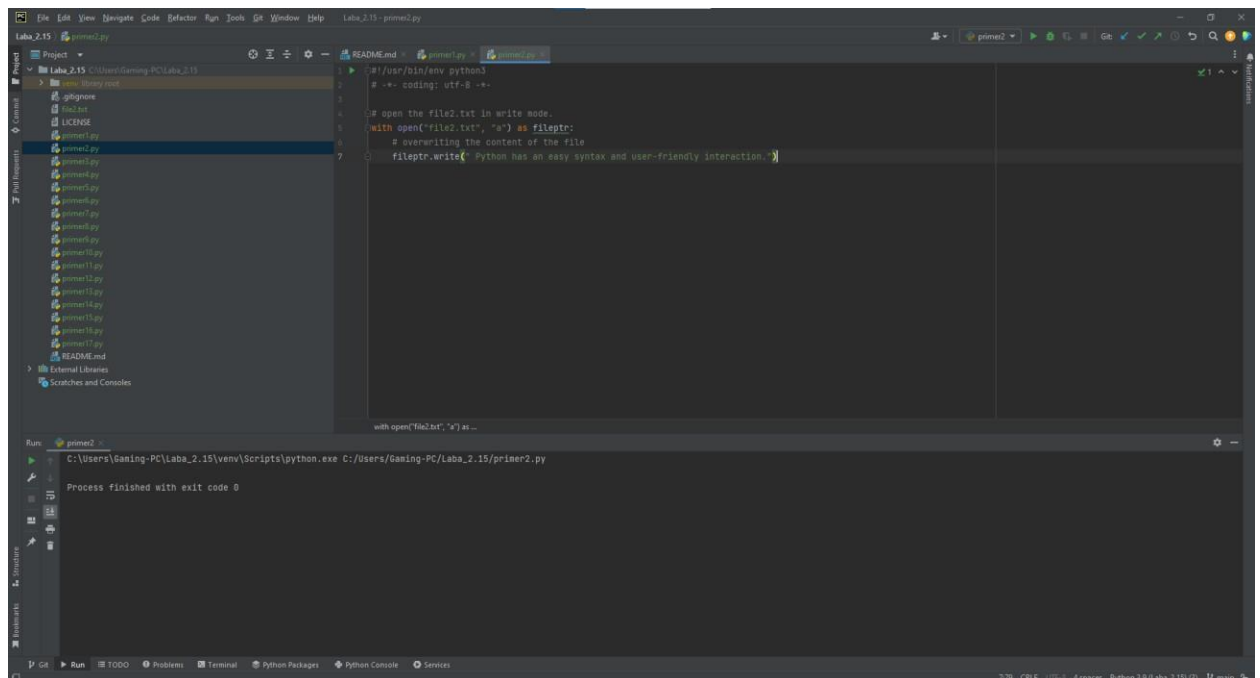


Рисунок 6. Реализация второго примера

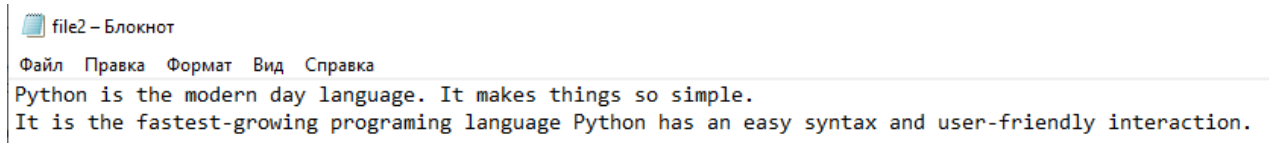


Рисунок 7. Результат второго примера

Создала новый файл под названием primer3.py

Условие примера: чтение строк с помощью метода readline()

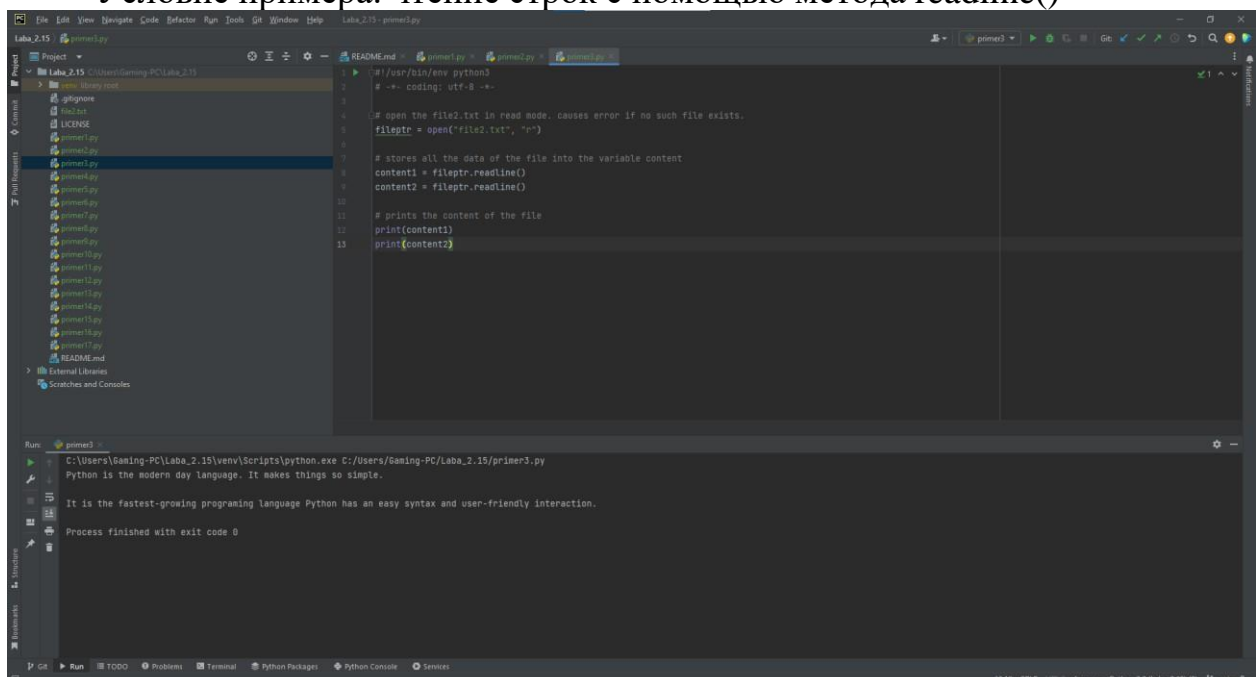


Рисунок 8. Реализация третьего примера

Создала новый файл под названием Пример 4.py

Условие примера: чтение строк с помощью функции readlines()

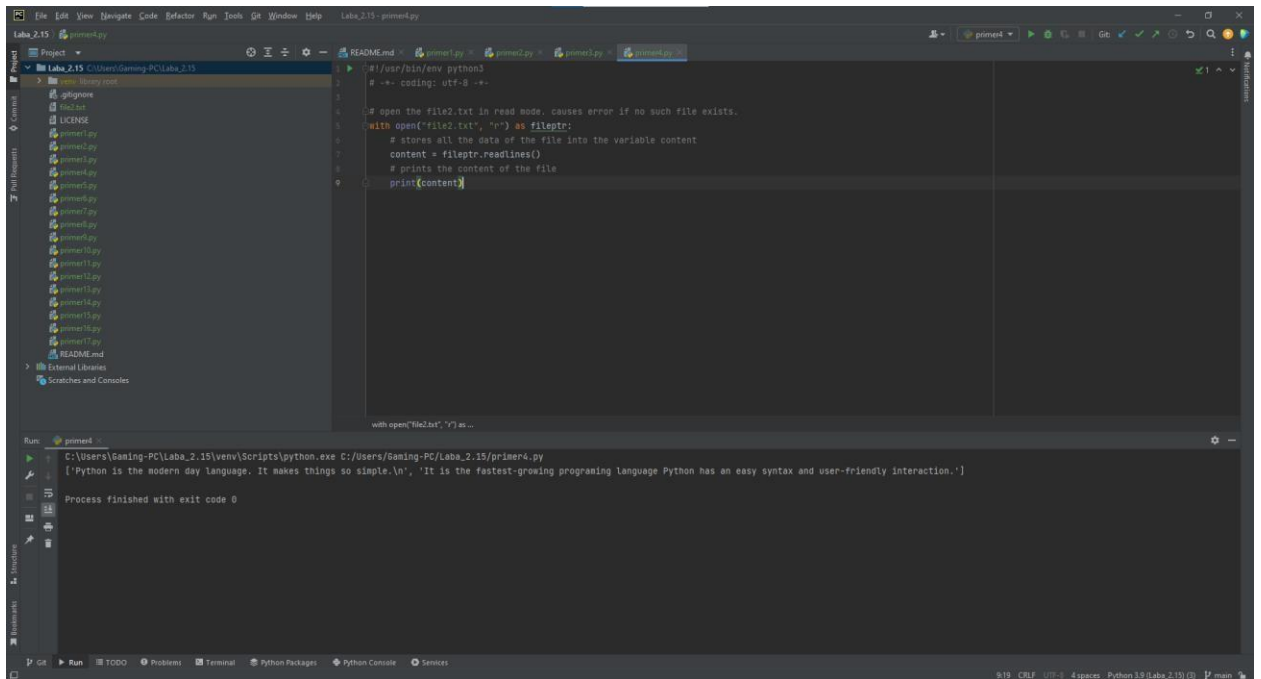


Рисунок 9. Реализация четвертого примера

Создала новый файл под названием Пример 5.py

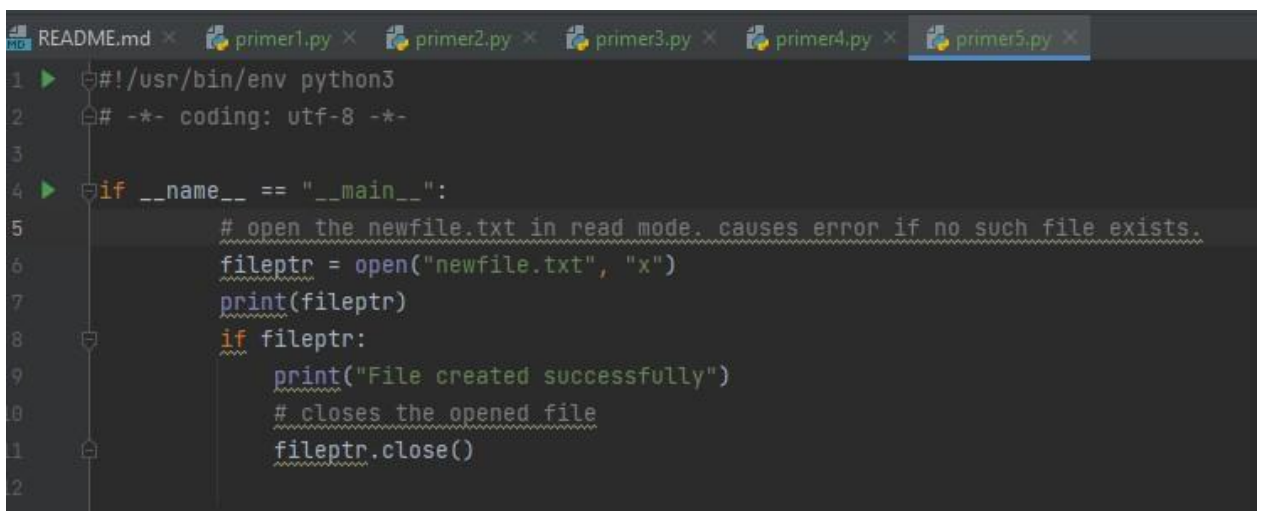


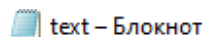
Рисунок 10. Реализация пятого примера

Создала новый файл под названием Пример 6.py

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

▶ if __name__ == "__main__":
    # open the text.txt in append mode. Create a new file if no such file exists.
    with open("text.txt", "a", encoding="utf-8") as fileptr:
        # appending the content to the file
        print(
            "UTF-8 is a variable-width character encoding used for electronic communication.",
            file=fileptr,
        )
        print(
            "UTF-8 is capable of encoding all 1,112,064 valid character code points.",
            file=fileptr,
        )
        print("In Unicode using one to four one-byte (8-bit) code units.", file=fileptr)
```

Рисунок 11. Реализация шестого примера



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.

Рисунок 12. Результат шестого примера

Создала новый файл под названием Пример 7.py

Условие примера: Написать программу, которая считывает текст из файла и выводит на экран только предложения, содержащие запятые. Каждое предложение в файле записано на отдельной строке.

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

Run: primer7.py
C:\Users\Gaming-PC\Lab2.15\Scripts\python.exe C:\Users\Gaming-PC\Lab2.15\primer7.py
UTF-8 is capable of encoding all 1,112,064 valid character code points.
Process finished with exit code 0

Рисунок 13. Реализация седьмого примера

Создала новый файл под названием Пример 8.py

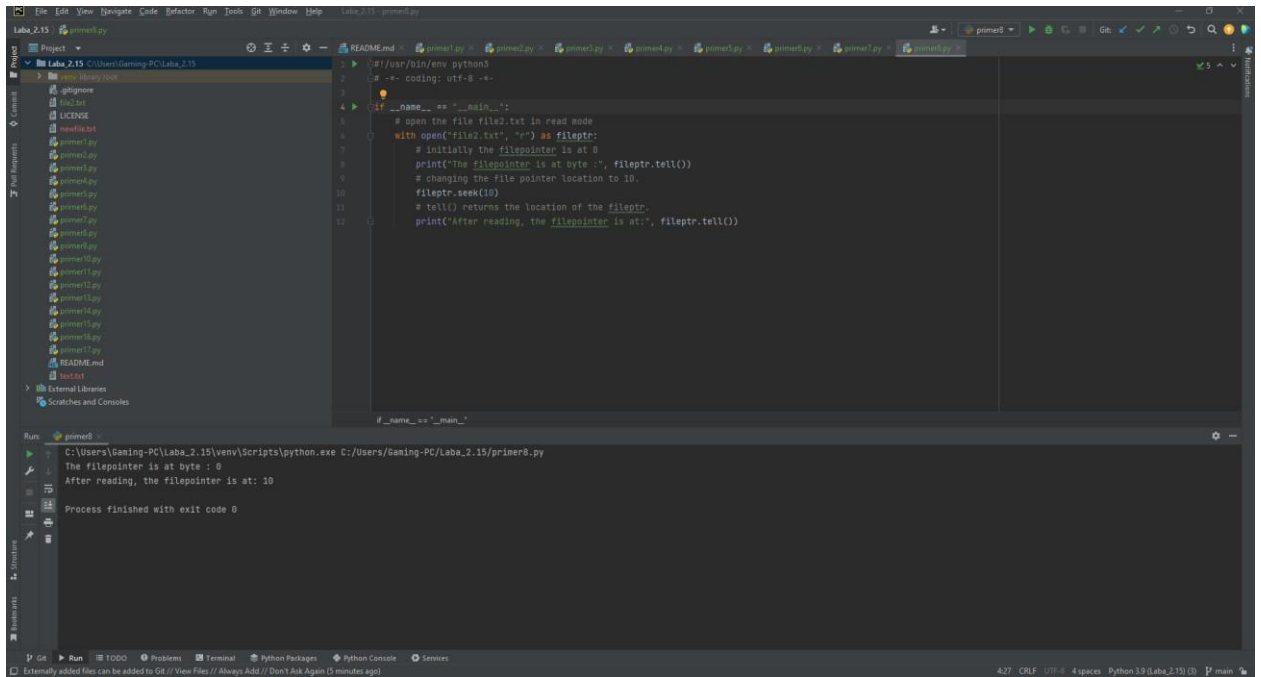


Рисунок 14. Реализация восьмого примера

Создал новый файл под названием Пример 9.py

Условие примера: Приведенный выше код переименует файл file2.txt в file3.txt в текущей рабочей папке.

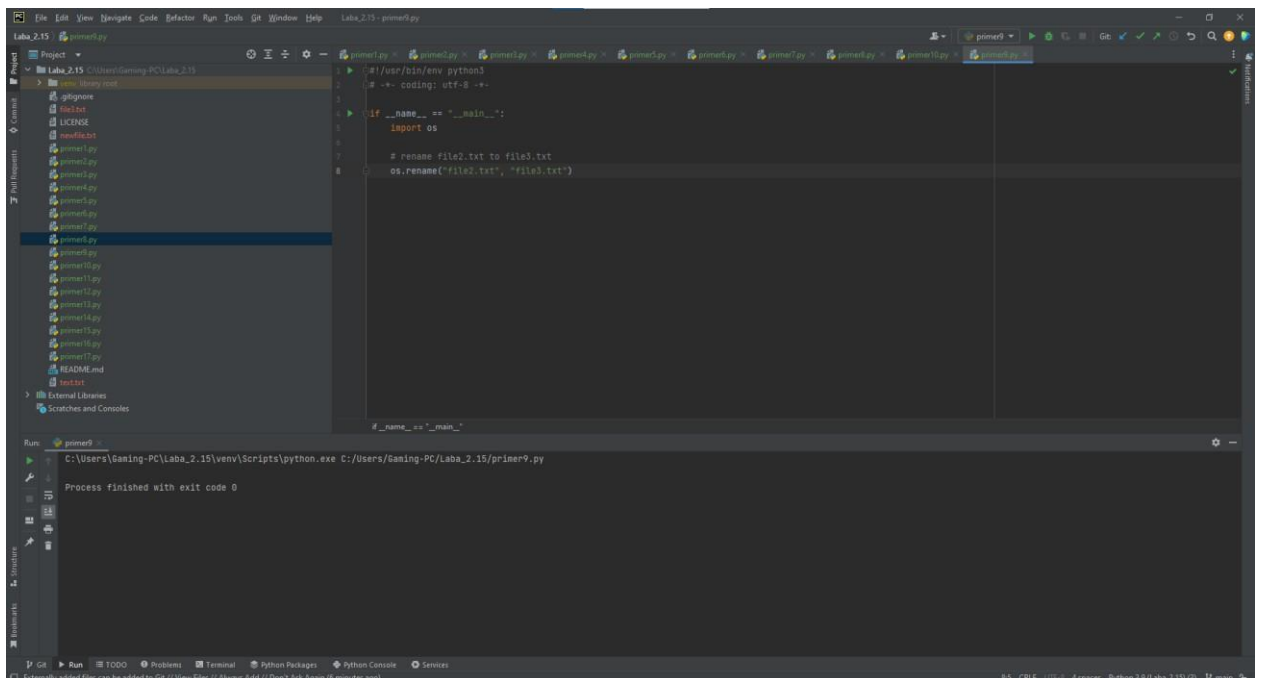


Рисунок 15. Реализация девятого примера

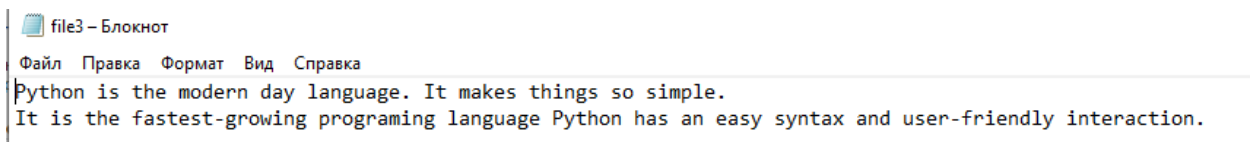


Рисунок 16. Результат девятого примера

Создала новый файл под названием Пример10.py

Условие примера: Приведенный выше код удалит файл file3.txt в текущей рабочей папке.

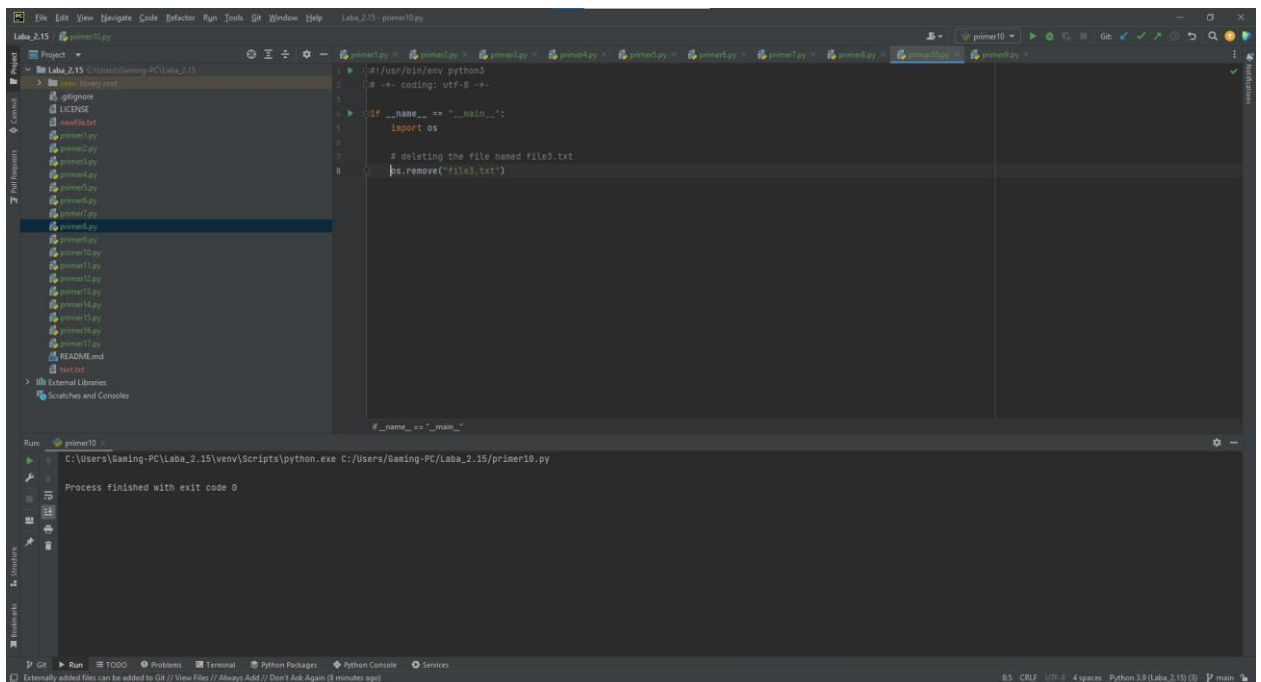


Рисунок 16. Реализация десятого примера

Создала новый файл под названием primer11.py

Условие примера: Приведенный выше код создаст каталог new в текущей рабочей папке.

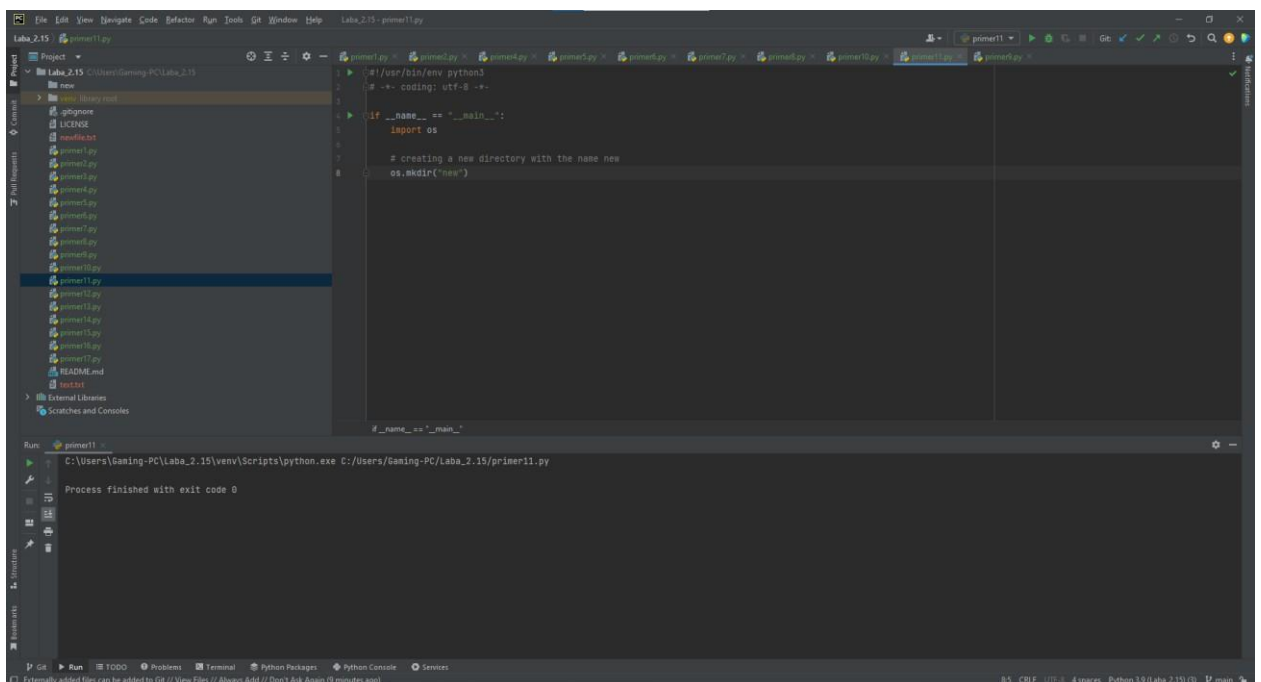


Рисунок 18. Реализация одиннадцатого примера




















Имя	Дата изменения	Тип	Размер
 .idea	17.12.2023 22:07	Папка с файлами	
 new	17.12.2023 22:06	Папка с файлами	
 venv	17.12.2023 22:07	Папка с файлами	
 .gitignore	29.10.2023 20:27	Исходный файл G...	4 КБ
 envirement	29.10.2023 20:27	Исходный файл Y...	2 КБ
 idz1	29.10.2023 20:27	Python File	1 КБ
 idz1	29.10.2023 20:27	Текстовый докум...	1 КБ
 idz2	29.10.2023 20:27	Python File	4 КБ
 idz2	29.10.2023 20:27	Текстовый докум...	2 КБ
 Lab2.15	17.12.2023 22:25	Документ Microso...	1 487 КБ
 newfile	29.10.2023 20:27	Текстовый докум...	0 КБ
 os_idz	29.10.2023 20:27	Python File	1 КБ
 Пример 1	18.12.2023 10:34	Python File	1 КБ
 Пример 2	29.10.2023 20:27	Python File	1 КБ
 Пример 3	29.10.2023 20:27	Python File	1 КБ
 Пример 4	29.10.2023 20:27	Python File	1 КБ
 Пример 5	29.10.2023 20:27	Python File	1 КБ
 Пример 6	29.10.2023 20:27	Python File	1 КБ
 Пример 7	29.10.2023 20:27	Python File	1 КБ

Рисунок 19. Результат программы

Создали новый файл под названием Пример 12.py

Условие примера: Приведенный код выведет имя текущего рабочего каталога.

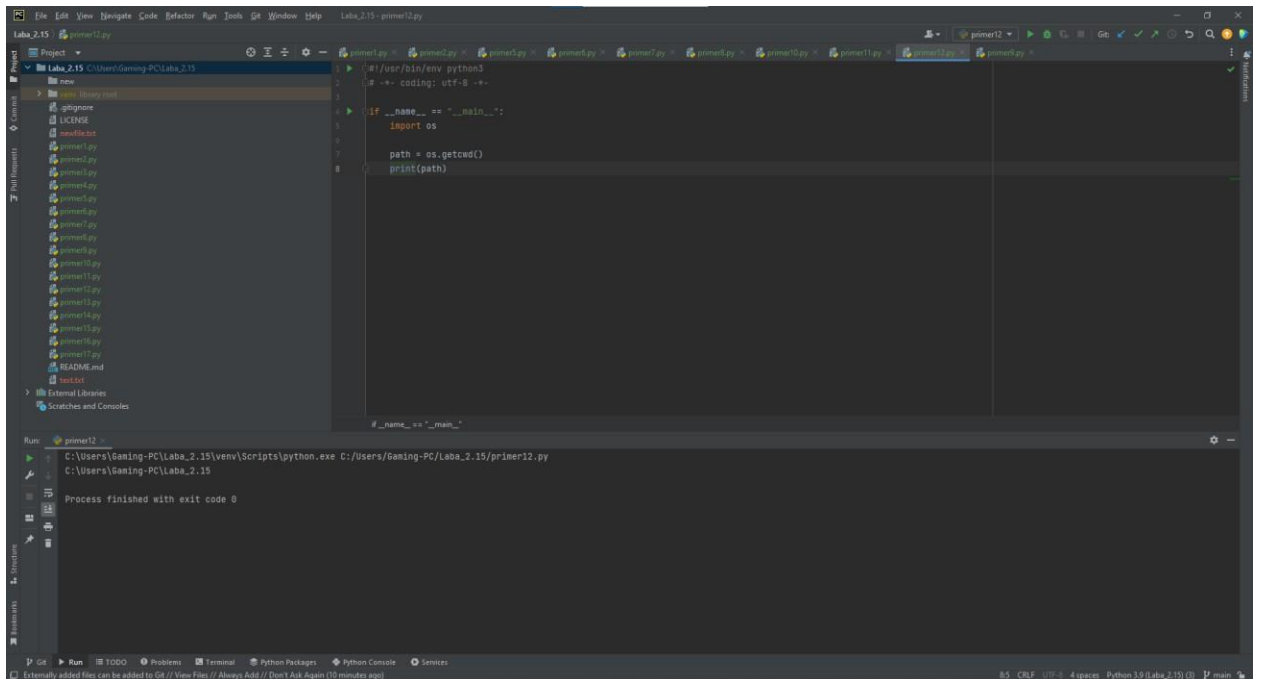


Рисунок 20. Реализация двенадцатого примера

Создала новый файл под названием Пример 13.py

Условие примера: Приведенный код для операционной системы Windows сменит текущий каталог на C:\Windows и выведет новое имя текущего каталога.

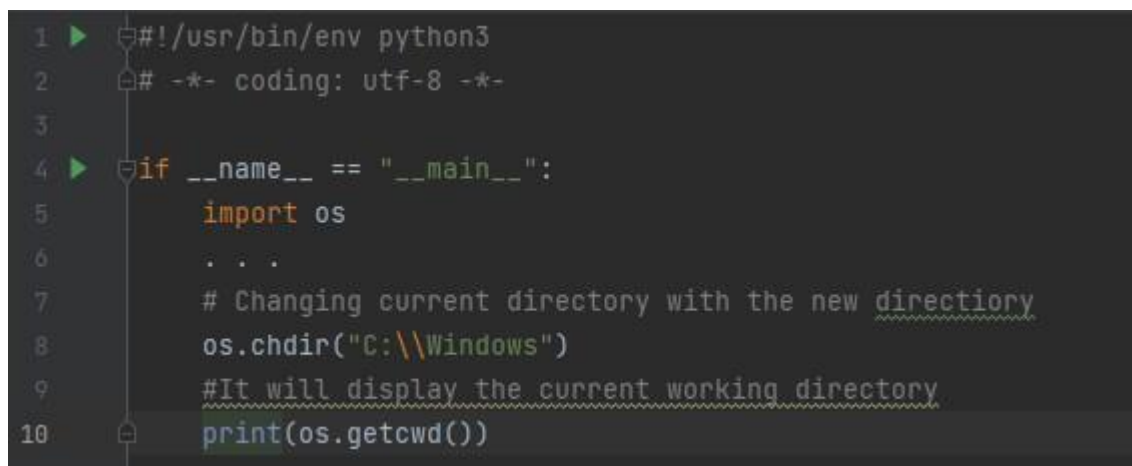


Рисунок 21. Реализация тринадцатого примера

Создала новый файл под названием Пример 14.py

Условие примера: Данный код удалит ранее созданный каталог new при условии, что он не пуст и находится в текущей рабочей папке.

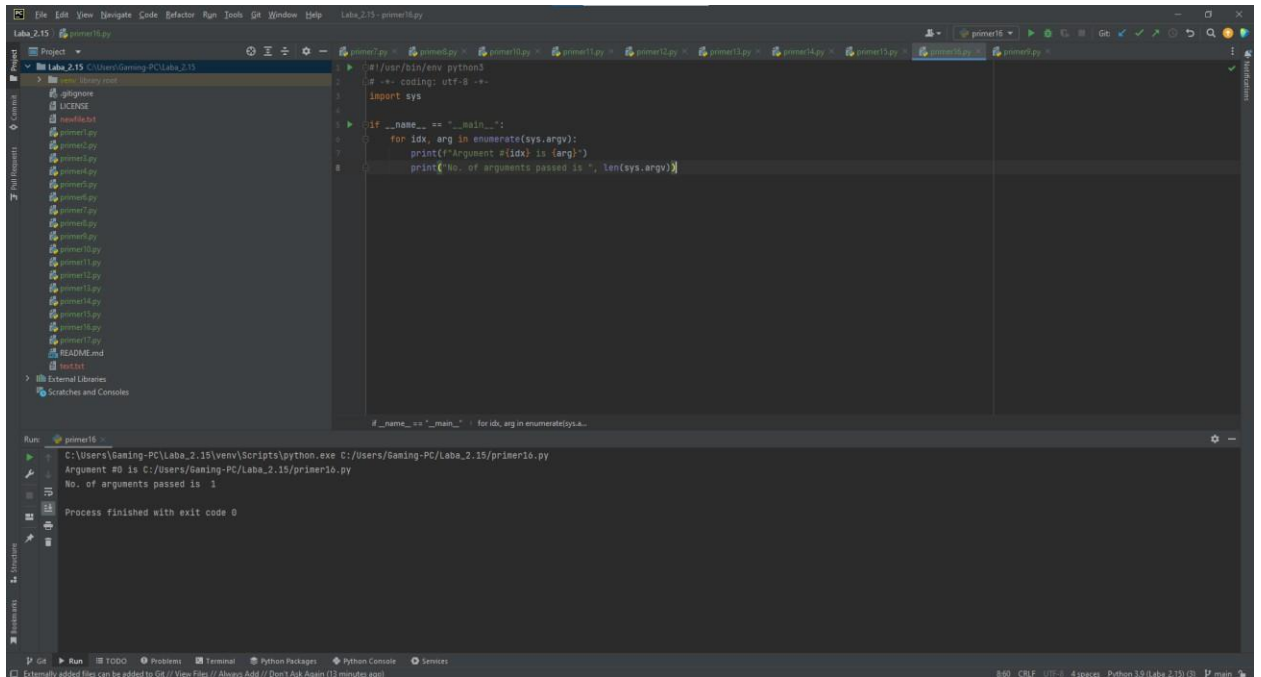


Рисунок 24. Реализация шестнадцатого примера

Создала новый файл под названием Пример 17.py

Условие примера: Написать программу для генерации пароля заданной длины. Длина пароля должна передаваться как аргумент командной строки сценария.

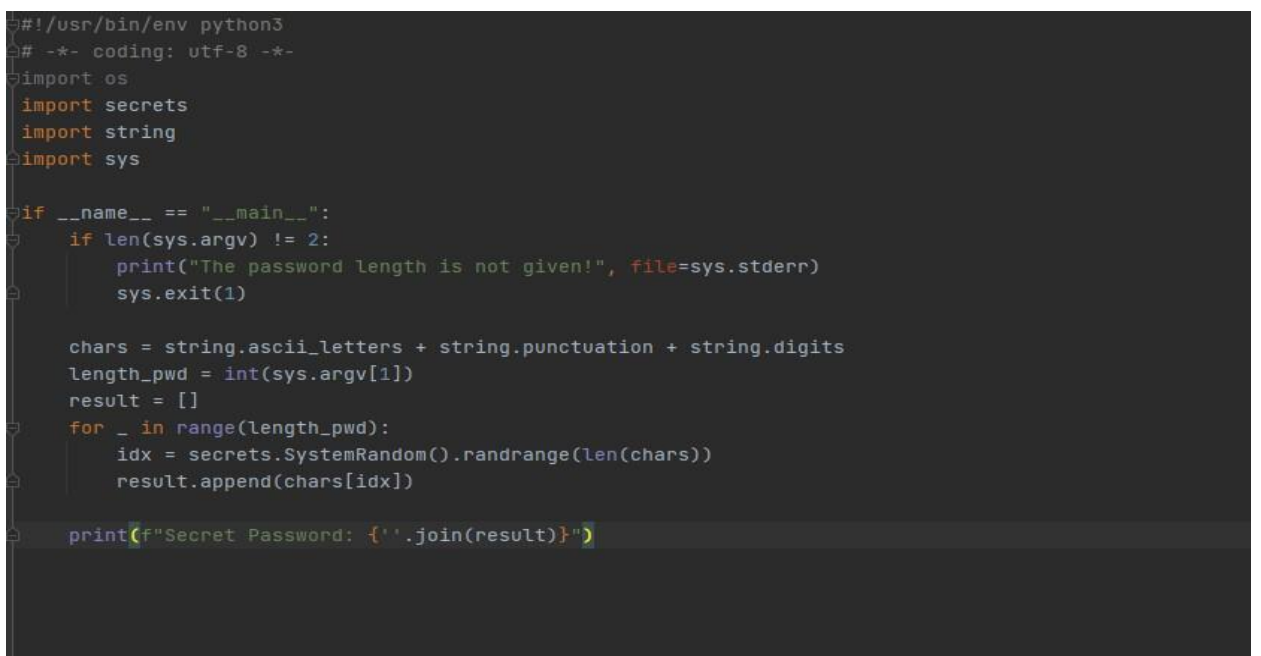


Рисунок 25. Реализация семнадцатого примера

Задание 5.

Создала новый файл под названием os_idz.py

Условие примера: самостоятельно подобрать или придумать задачу для работы с изученными функциями модуля os. Привести решение этой задачи.

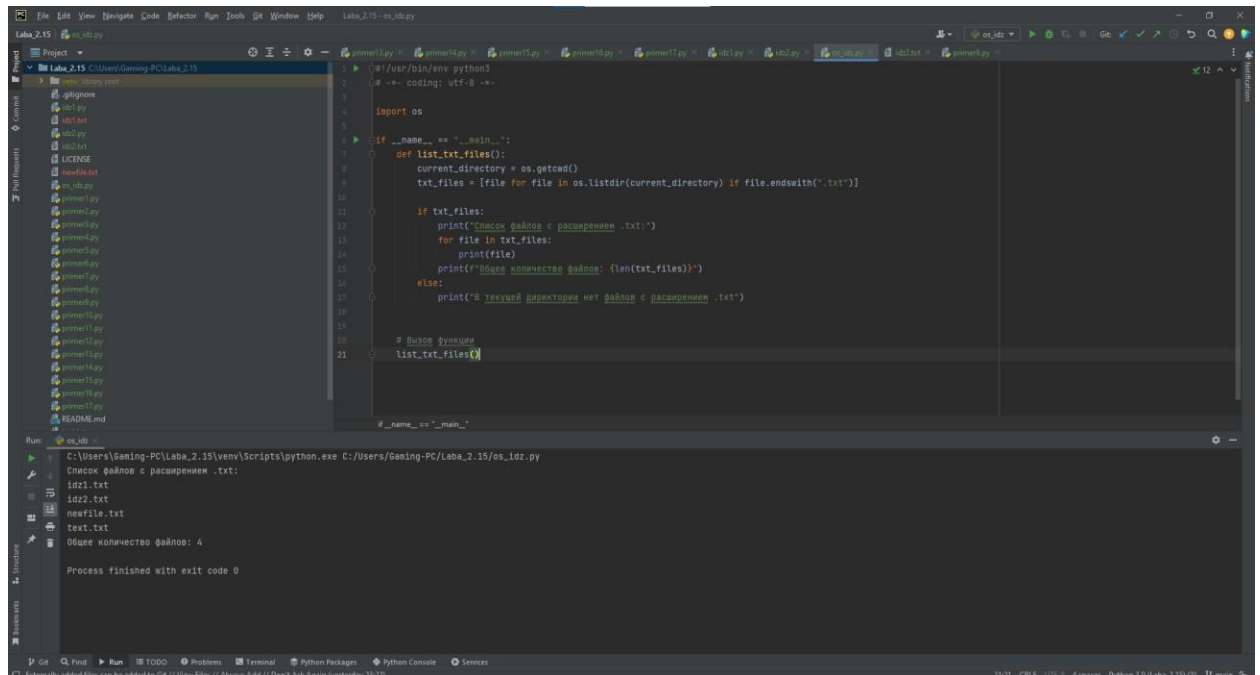


Рисунок 26. Программа индивидуального задания

Индивидуальное задание

Вариант 2

Создали новый файл под названием Индив 1.py.

Условие задания: Написать программу, которая считывает текст из файла и выводит на экран только предложения, содержащие введенное с клавиатуры слово.

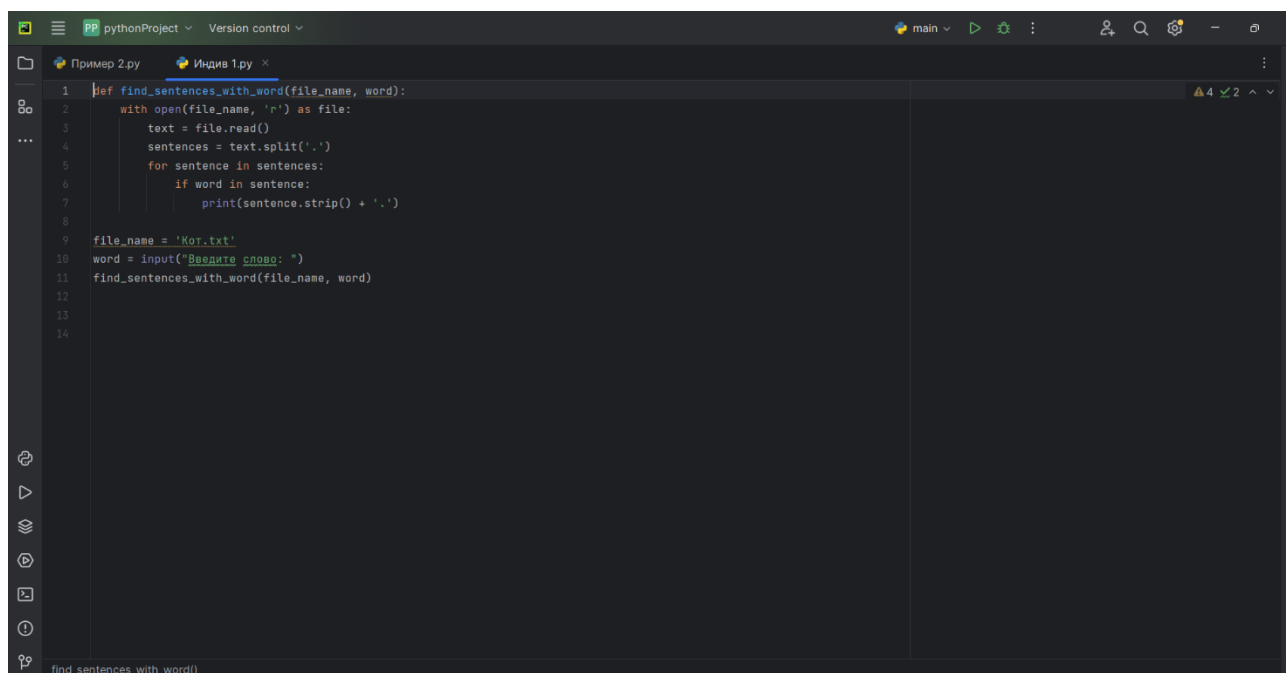
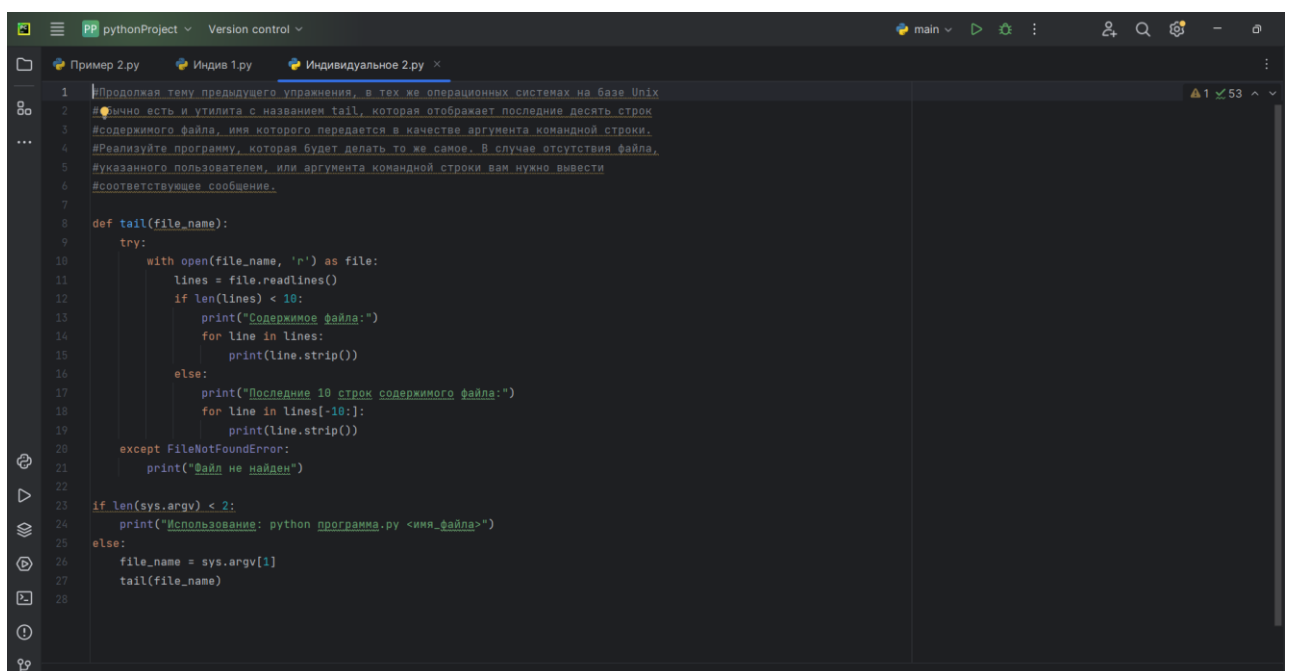


Рисунок 27. Реализация первого индивидуального задания

Создали новый файл под названием idz2.py.

Условие задания: Продолжая тему предыдущего упражнения, в тех же операционных системах на базе Unix обычно есть и утилита с названием tail, которая отображает последние десять строк содержимого файла, имя которого передается в качестве аргумента командной строки. Реализуйте программу, которая будет делать то же самое. В случае отсутствия файла, указанного пользователем, или аргумента командной строки вам нужно вывести соответствующее сообщение.



```
1 #Продолжая тему предыдущего упражнения, в тех же операционных системах на базе Unix
2 #обычно есть и утилита с названием tail, которая отображает последние десять строк
3 #содержимого файла, имя которого передается в качестве аргумента командной строки.
4 #Реализуйте программу, которая будет делать то же самое. В случае отсутствия файла,
5 #указанного пользователем, или аргумента командной строки вам нужно вывести
6 #соответствующее сообщение.
7
8 def tail(file_name):
9     try:
10         with open(file_name, 'r') as file:
11             lines = file.readlines()
12             if len(lines) < 10:
13                 print("Содержимое файла:")
14                 for line in lines:
15                     print(line.strip())
16             else:
17                 print("Последние 10 строк содержимого файла:")
18                 for line in lines[-10:]:
19                     print(line.strip())
20     except FileNotFoundError:
21         print("@файл не найден")
22
23 if len(sys.argv) < 2:
24     print("Использование: python программа.py <имя_файла>")
25 else:
26     file_name = sys.argv[1]
27     tail(file_name)
28
```

Рисунок 29. Реализация второго индивидуального задания

Ссылка: https://github.com/stefa-b/Laba_2.15.git

Ответы на контрольные вопросы:

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

'r' - Открывает файл только для чтения.

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

'w' – Открывает файл только для записи.

3. Как прочитать данные из файла в языке Python?

Функция read() используется для чтения содержимого файла после открытия его в режиме чтения (r).

4. Как записать данные в файл в языке Python?

open('file.txt', 'w') открывает файл для записи. Параметр 'w' указывает режим открытия файла (в данном случае, для записи).

Контекстный менеджер with автоматически закрывает файл после использования блока кода внутри него.

write() используется для записи строки в файл.

writelines() записывает список строк в файл.

Метод print() с параметром file=file записывает текст в файл.

json.dump() записывает данные в формате JSON в файл.

5. Как закрыть файл в языке Python?

Метод файла file. close() закрывает открытый файл. Закрытый файл больше не может быть прочитан или записан. Любая операция, которая требует, чтобы файл был открыт, вызовет исключение ValueError после того, как файл был закрыт.

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 предназначена для управления контекстами ресурсов, таких как файлы, сетевые соединения, базы данных и другие, чтобы гарантировать их корректное открытие, использование и закрытие. Она гарантирует, что ресурсы будут правильно освобождены после завершения блока кода. Конструкция with ... as обычно используется с контекстными менеджерами, которые определяют методы __enter

_____ и _____ `exit` для выполнения действий до и после использования ресурсов.

Кроме работы с файлами, `with ... as` может быть использована для работы с сетевыми соединениями (например, с помощью библиотеки `socket`), управления транзакциями в базах данных (с библиотекой `sqlite3`), манипуляций с ресурсами, требующими блокировки, и многими другими ситуациями, где важна правильная инициализация и освобождение ресурсов.

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

Помимо рассмотренных методов (`read()`, `write()`, `readline()`, `writelines()` и т. д.) для чтения и записи информации в файлы, существует ряд других методов в Python для более специфических задач:

- `readlines()`: Этот метод читает все строки из файла и возвращает их в виде списка строк.
- `seek()`: Метод `seek()` используется для перемещения указателя файла в заданную позицию. Это полезно, например, чтобы перейти к определенному месту в файле.
- `tell()`: Метод `tell()` возвращает текущую позицию указателя файла.
- `flush()`: Метод `flush()` записывает непрошедшие данные на диск, но не закрывает файл.
- `truncate()`: Метод `truncate()` урезает файл до указанного размера.

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

Модуль `os` в Python предоставляет множество функций для работы с файловой системой. Помимо рассмотренных ранее функций, некоторые другие функции модуля `os` включают:

- `os.rename(src, dst)`: Используется для переименования файла или директории.
- `os.remove(path)`: Удаляет файл.
- `os.rmdir(path)`: Удаляет пустую директорию.
- `os.removedirs(path)`: Удаляет директории рекурсивно.

- `os.listdir(path)`: Возвращает список файлов и директорий в указанной директории.
- `os.makedirs(path)`: Создает директорию или директории, включая промежуточные.
- `os.path`: Этот модуль предоставляет функции для работы с путями к файлам и директориям.

Это лишь несколько примеров функций модуля `os`. Модуль `os` предоставляет множество других функций для работы с файловой системой и системными вызовами в операционной системе.

Вывод: приобрела навыки по работе с текстовыми файлами при написании программ с помощью языка программирования Python версии 3.x, изучила основные методы модуля `os` для работы с файловой системой, получил аргументы командной строки.