

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

Кафедра инфокоммуникаций

**Отчет по лабораторной работе №10**

**Работа с файлами в языке Python.**

**По дисциплине «Технологии программирования и алгоритмизация»**

Выполнил студент группы ИВТ-б-о-20-1

Бобров Н. В. « » \_\_\_\_\_ 20\_\_ г.

Подпись студента \_\_\_\_\_

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

Проверил Воронкин Р. А. \_\_\_\_\_

(подпись)

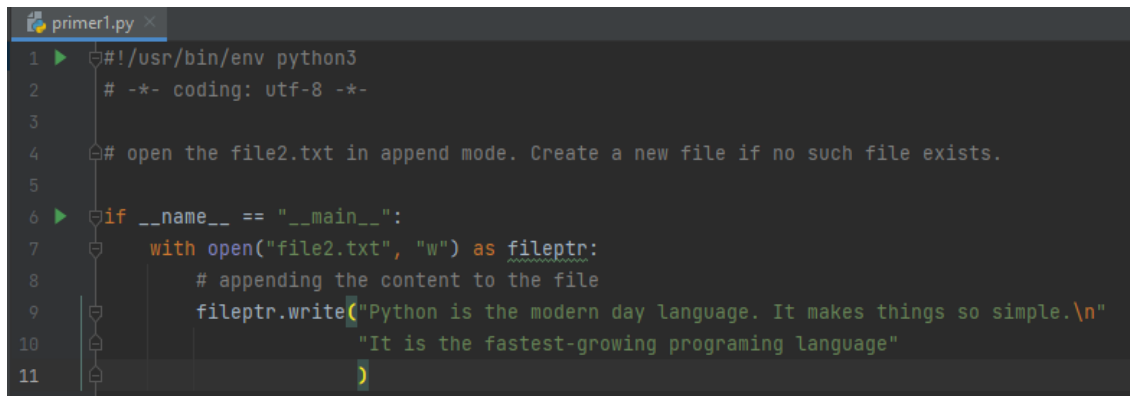
Ставрополь 2021

**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 работы:

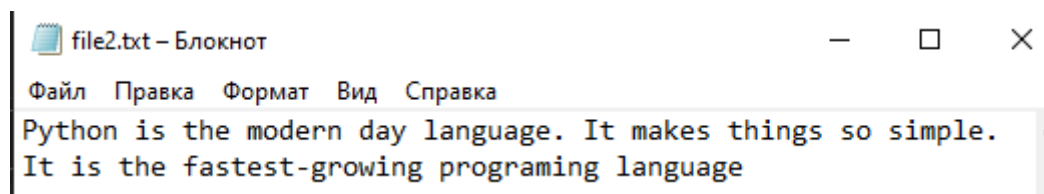
Ссылка на репозиторий: <https://github.com/nbobrov8/laba17>

1. Создал общедоступный репозиторий и клонировал его на локальный сервер.
2. Изучив теорию, приступил к проработке примеров из методического материала.



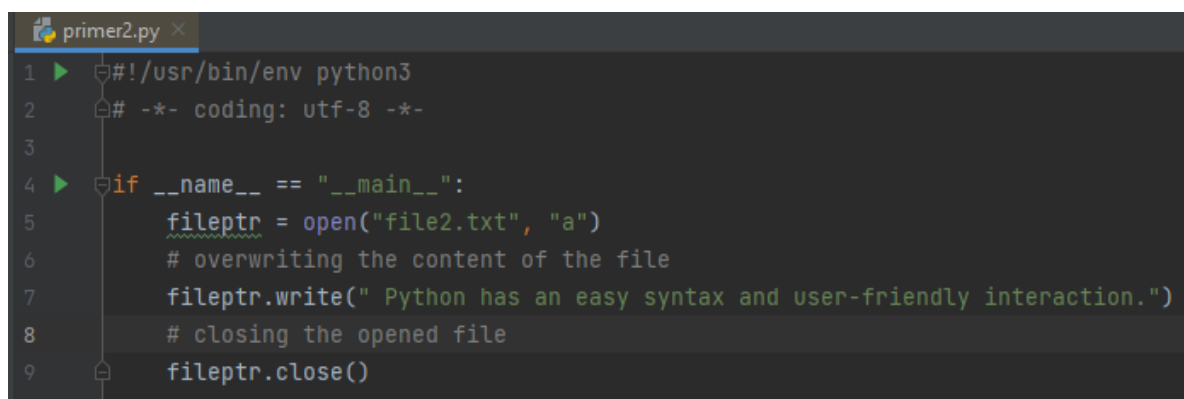
```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 # open the file2.txt in append mode. Create a new file if no such file exists.
5
6 if __name__ == "__main__":
7     with open("file2.txt", "a") as fileptr:
8         # appending the content to the file
9         fileptr.write("Python is the modern day language. It makes things so simple.\n"
10                      "It is the fastest-growing programming language")
11
```

Рисунок 1 – Код первого примера



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

Рисунок 2 – Результат записи в file2.txt



```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 if __name__ == "__main__":
5     fileptr = open("file2.txt", "a")
6     # overwriting the content of the file
7     fileptr.write(" Python has an easy syntax and user-friendly interaction.")
8     # closing the opened file
9     fileptr.close()
```

Рисунок 3 – Код второго примера

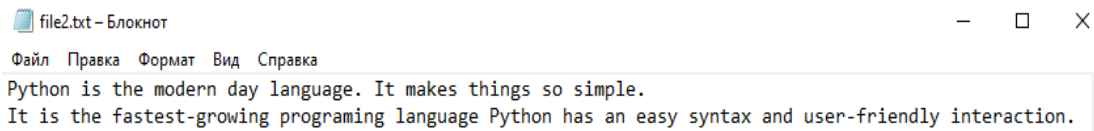


Рисунок 4 – Результат записи в файл

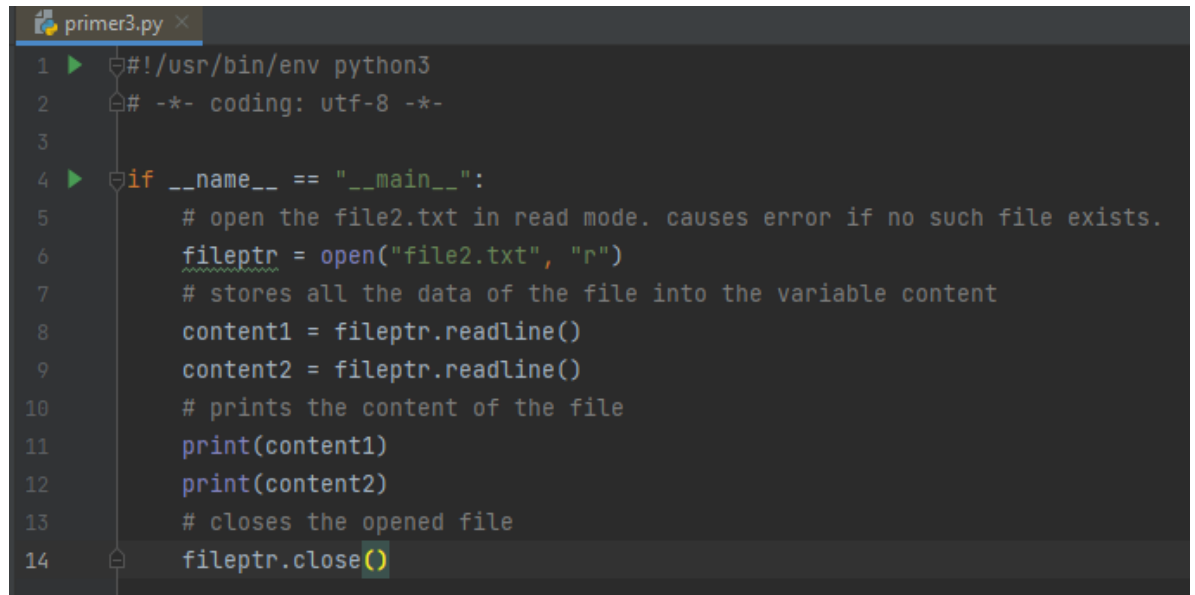


Рисунок 5 – Проработка 3 примера с помощью метода readline()

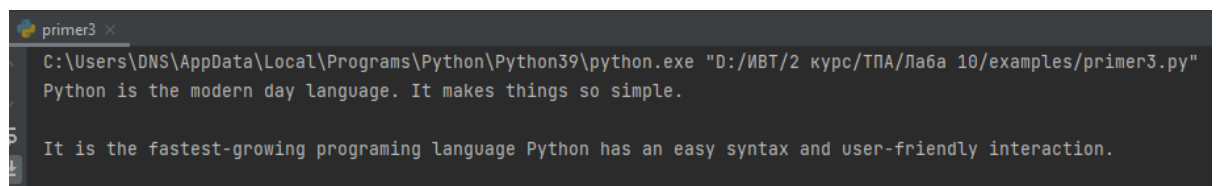


Рисунок 6 – Результат выполнения кода

3. Проработал остальные 14 примеров и приступил к выполнению индивидуальных задания согласно моему варианту.

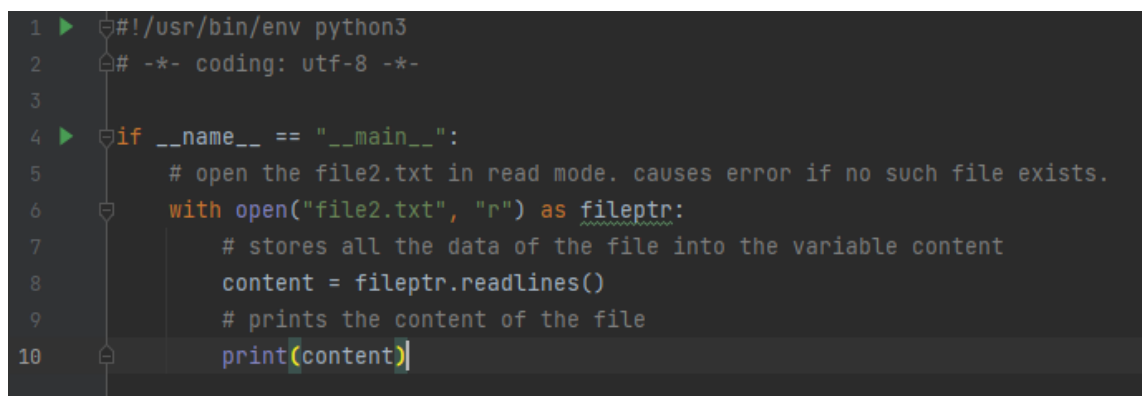


Рисунок 7 – Четвертый пример

```

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     if fileptr:
9         print("File created successfully")
10    # closes the opened file
11    fileptr.close()

```

Рисунок 8 – Пятый пример

```

5 ▶ if __name__ == "__main__":
6     # open the text.txt in append mode. Create a new file if no such file
7     # exists.
8     with open("text.txt", "w", encoding="utf-8") as fileptr:
9         # appending the content to the file
10        print(
11            "UTF-8 is a variable-width character encoding used for electronic communication.",
12            file=fileptr
13        )
14        print(
15            "UTF-8 is capable of encoding all 1,112,064 valid character code points.",
16            file=fileptr
17        )
18        print(
19            "In Unicode using one to four one-byte (8-bit) code units.",
20            file=fileptr
21        )
22

```

Рисунок 9 – Шестой пример

```

4 ▶ if __name__ == "__main__":
5     with open("text.txt", "r", encoding="utf-8") as fileptr:
6         sentences = fileptr.readlines()
7
8     # Вывод предложений с запятыми.
9     for sentence in sentences:
10        if "," in sentence:
11            print(sentence)

```

Рисунок 10 – Седьмой пример

```

4 ► if __name__ == "__main__":
5     # open the file file2.txt in read mode
6     with open("file2.txt", "r") as fileptr:
7         # initially the filepointer is at 0
8         print("The filepointer is at byte :", fileptr.tell())
9         # changing the file pointer location to 10.
10        fileptr.seek(10)
11        # tell() returns the location of the fileptr.
12        print("After reading, the filepointer is at:", fileptr.tell())

```

Рисунок 11 – Восьмой пример

```

1 ► #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import os
5
6 ► if __name__ == "__main__":
7     os.rename("file2.txt", "file3.txt")

```

Рисунок 12 – Девятый пример

```

1 ► #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import os
5
6 ► if __name__ == "__main__":
7     os.remove("file3.txt")

```

Рисунок 13 – Десятый пример

```

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

```

Рисунок 14 – Одиннадцатый пример

```

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)

```

Рисунок 15 – Двенадцатый пример

```

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())

```

Рисунок 16 – Тринадцатый пример

```

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

```

Рисунок 17 – Четырнадцатый пример

```

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

```

Рисунок 18 – Пятнадцатый пример

```

1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import sys
5
6
7 ▶ if __name__ == "__main__":
8     for idx, arg in enumerate(sys.argv):
9         print(f"Argument #{idx} is {arg}")
10    print("No. of arguments passed is ", len(sys.argv))

```

Рисунок 19 – Шестнадцатый пример

```

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)}")

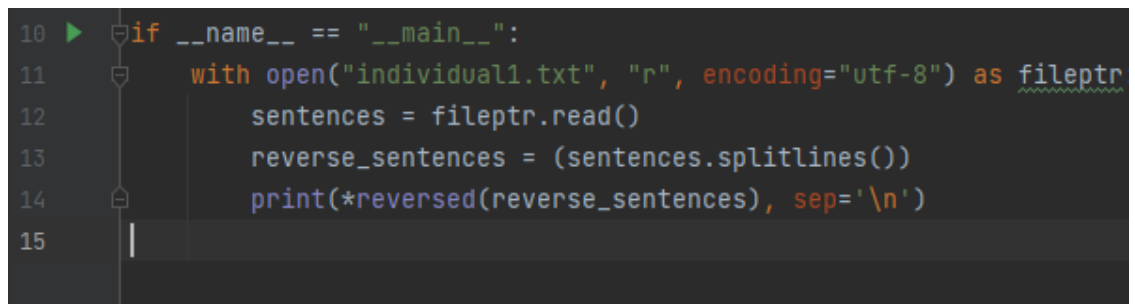
```

Рисунок 20 – Семнадцатый пример

### Индивидуальные задания. Вариант 1.

**Задание 1.** Написать программу, которая считывает из текстового файла три предложения и выводит их в обратном порядке.

1. Написал код для решения задачи.



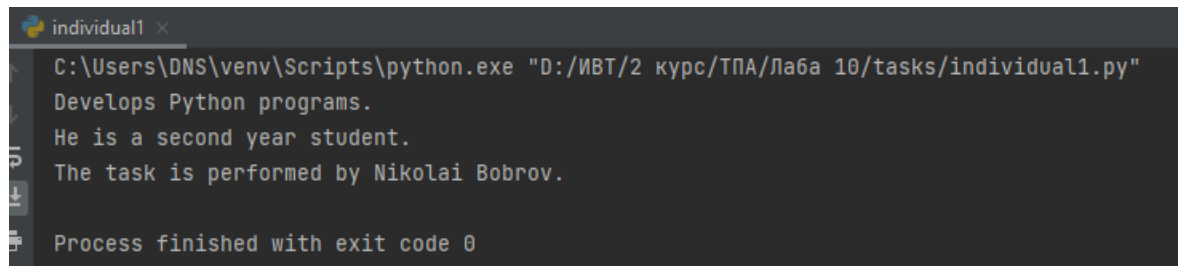
```

10 if __name__ == "__main__":
11     with open("individual1.txt", "r", encoding="utf-8") as fileptr:
12         sentences = fileptr.read()
13         reverse_sentences = (sentences.splitlines())
14         print(*reversed(reverse_sentences), sep='\n')
15

```

Рисунок 21 – Код для выполнения задания

2. Проверил работоспособность кода.



```

individual1 x
C:\Users\DNS\venv\Scripts\python.exe "D:/ИВТ/2 курс/ТПА/Лаба 10/tasks/individual1.py"
Develops Python programs.
He is a second year student.
The task is performed by Nikolai Bobrov.

Process finished with exit code 0

```

Рисунок 22 – Результат выполнения кода

**Задание 2.** В операционных системах на базе Unix обычно присутствует утилита с названием `head`. Она выводит первые десять строк содержимого файла, имя которого передается в качестве аргумента командной строки. Напишите программу на Python, имитирующую поведение этой утилиты. Если файла, указанного пользователем, не существует, или не задан аргумент командной строки, необходимо вывести соответствующее сообщение об ошибке.

1. Написал код для решения задачи.



```

14  if __name__ == "__main__":
15      ten_lines = 10
16
17      if len(sys.argv) != 2:
18          print("В качестве аргумента командной строки передайте имя файла!", file=sys.stderr)
19          sys.exit(1)
20
21      try:
22          # открываем файл на чтение
23          with open(sys.argv[1], "r", encoding="utf-8") as file:
24
25              # читаем первую строку из файла
26              line = file.readlines()
27
28              # создаем список
29              list_lines = []
30
31              # запуск цикла, читаем строки пока не дойдём до 10
32              count = 0
33              for rows in line:
34                  if count < ten_lines:
35                      list_lines.append(rows)
36                      count = count + 1
37
38                  print(*list_lines, end="")
39
40      except IOError:
41          # если возникнут проблемы с чтением файла, отображаем ошибку
42          print("Ошибка при доступе к файлу", file=sys.stderr)

```

Рисунок 23 – Код для решения второго задания

2. Проверил работоспособность кода для второго задания через терминал.

```

PS D:\ИВТ\2 курс\ТПА\Лаба 10\tasks> py individual2.py individual2.txt
Красивое лучше уродливого.
Явное лучше неявного.
Простое лучше сложного.
Сложное лучше запутанного.
Развернутое лучше вложенного.
Разреженное лучше плотного.
Читаемость имеет значение.
Особые случаи не настолько особые, чтобы нарушать правила.
При этом практичность важнее безупречности.
Ошибки не должны замалчиваться.
PS D:\ИВТ\2 курс\ТПА\Лаба 10\tasks>

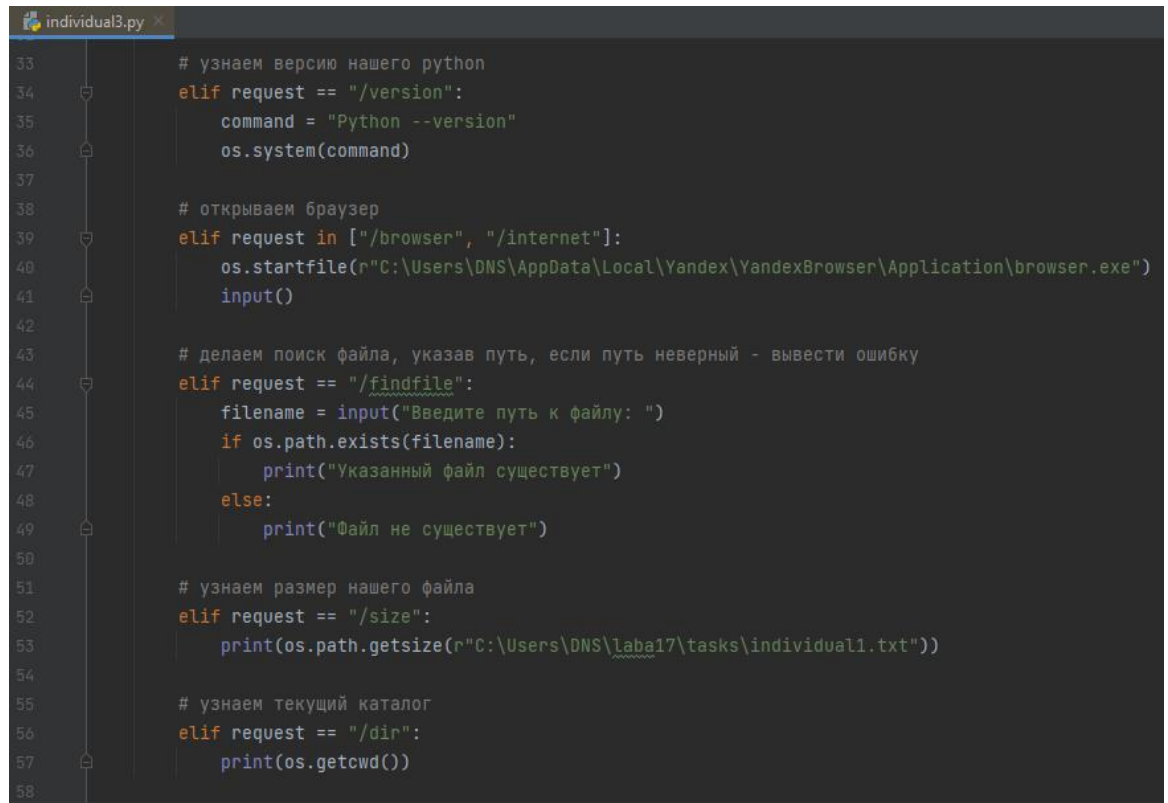
```

Рисунок 24 – Результат работы кода второго задания

**Задание 3.** Самостоятельно подберите или придумайте задачу для работы с изученными функциями модуля os. Приведите решение этой задачи.

Условие: разработать терминал, принимающий команды от пользователя. Использовать модуль os.

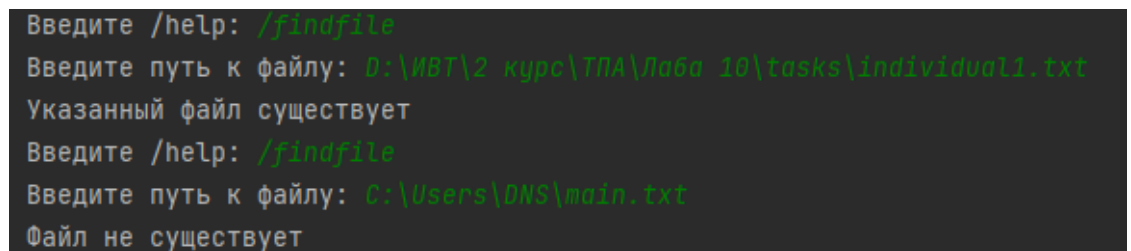
1. Написал код для реализации задачи.



```
33 # узнаем версию нашего python
34 elif request == "/version":
35     command = "Python --version"
36     os.system(command)
37
38 # открываем браузер
39 elif request in ["/browser", "/internet"]:
40     os.startfile(r"C:\Users\DNS\AppData\Local\Yandex\YandexBrowser\Application\browser.exe")
41     input()
42
43 # делаем поиск файла, указав путь, если путь неверный - вывести ошибку
44 elif request == "/findfile":
45     filename = input("Введите путь к файлу: ")
46     if os.path.exists(filename):
47         print("Указанный файл существует")
48     else:
49         print("Файл не существует")
50
51 # узнаем размер нашего файла
52 elif request == "/size":
53     print(os.path.getsize(r"C:\Users\DNS\lab17\tasks\individual1.txt"))
54
55 # узнаем текущий каталог
56 elif request == "/dir":
57     print(os.getcwd())
58
```

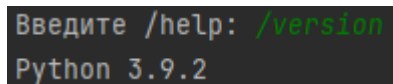
Рисунок 25 – Код для реализации задачи

2. Проверил работоспособность кода.



```
Введите /help: /findfile
Введите путь к файлу: D:\ИИТ\2 курс\ТРА\Лаба 10\tasks\individual1.txt
Указанный файл существует
Введите /help: /findfile
Введите путь к файлу: C:\Users\DNS\main.txt
Файл не существует
```

Рисунок 26 – Проверка команды findfile



```
Введите /help: /version
Python 3.9.2
```

Рисунок 27 – Проверка команды version

**Контрольные вопросы:**

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

Используя функцию open(), после ввода имени файла через запятую указать режим "r".

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

Используя функцию `open()`, после ввода имени файла через запятую указать режим “w”.

3. Как прочитать данные из файла в языке Python?

Сначала необходимо открыть файл, вызвав функцию `open()`, затем использовать метод `read()`.

4. Как записать данные в файл в языке Python?

Сначала необходимо открыть файл, вызвав функцию `open()`, затем использовать метод `write()`.

5. Как закрыть файл в языке Python?

Использовать метод `close()` или открывать файл при помощи оператора `with`, который закрывает файл, после окончания 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` гарантирует, что критические функции выполнятся в любом случае. В основном она используется для работы с файлами разного типа, но также может использоваться для фиксации или отката транзакции базы данных, для перенаправления стандартного вывода однопоточных программ.

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

Метод `writelines()` – добавляет последовательность строк в файл.

Метод `tell()` - возвращает текущую позицию “условного курсора” в файле.

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

**os.name** - имя операционной системы. **os.environ** - словарь переменных окружения. **os.getpid()** - текущий id процесса. **os.uname()** - информация об ОС.

**os.access** () - проверка доступа к объекту у текущего пользователя.

**os.chdir** (path) - смена текущей директории. **os.chmod** () - смена прав доступа к объекту. **os.link** () - создаёт жёсткую ссылку.

**os.listdir** () - список файлов и директорий в папке.

**os.makedirs** () - создаёт директорию, создавая при этом промежуточные директории.

**os.symlink** () - создаёт символическую ссылку на объект.

**os.truncate** () - обрезает файл до длины length.

**os.utime** () - модификация времени последнего доступа и изменения файла.

**os.walk** () - генерация имён файлов в дереве каталогов.

**os.system** () - исполняет системную команду, возвращает код её завершения.

**os.urandom** (n) - n случайных байт.

**os.path** - модуль, реализующий некоторые полезные функции на работы с путями.

**Вывод:** в ходе выполнения лабораторной работы приобрел навыки по работе с текстовыми файлами при написании программ с помощью языка программирования Python версии 3.x, изучил основные методы модуля os для работы с файловой системой и получения аргументов командной строки.