

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 ПРАКТИЧЕСКОЙ РАБОТЕ №1
дисциплины «Анализ данных»

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

(подпись)

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

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

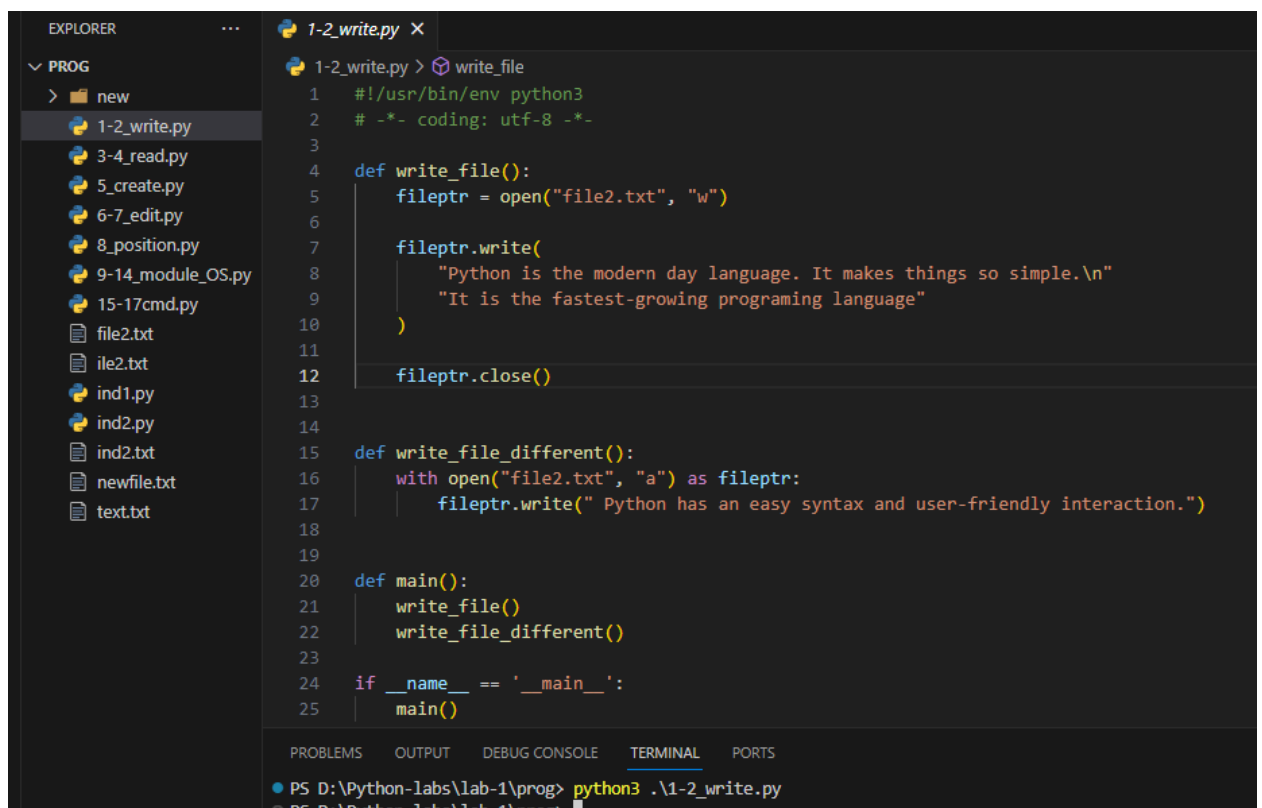
Тема: Работа с файлами в языке Python

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:

1. Проработка примеров данной лабораторной работы:

1) Написал программу (1-2_write.py), в которой проработал примеры раздела методических указаний «Запись файла»:

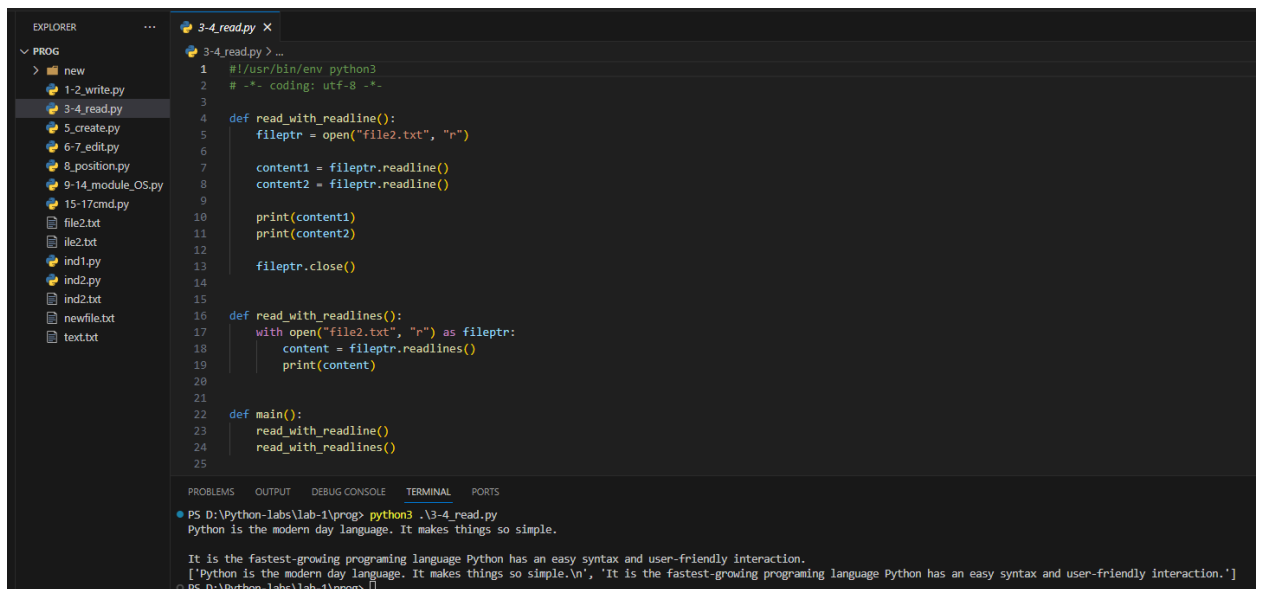


```
1-2_write.py X
1-2_write.py > write_file
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  def write_file():
5      fileptr = open("file2.txt", "w")
6
7      fileptr.write(
8          "Python is the modern day language. It makes things so simple.\n"
9          "It is the fastest-growing programming language"
10     )
11
12     fileptr.close()
13
14
15 def write_file_different():
16     with open("file2.txt", "a") as fileptr:
17         fileptr.write(" Python has an easy syntax and user-friendly interaction.")
18
19
20 def main():
21     write_file()
22     write_file_different()
23
24 if __name__ == '__main__':
25     main()

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\Python-labs\lab-1\prog> python3 .\1-2_write.py
```

Рисунок 1 – Результат выполнения программы 1-2_write.py

2) Написал программу (3-4_read.py), в которой проработал примеры раздела методических указаний «Построчное чтение содержимого файла с помощью методов файлового объекта»:



```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 def read_with_readline():
5     fileptr = open("file2.txt", "r")
6
7     content1 = fileptr.readline()
8     content2 = fileptr.readline()
9
10    print(content1)
11    print(content2)
12
13    fileptr.close()
14
15
16 def read_with_readlines():
17     with open("file2.txt", "r") as fileptr:
18         content = fileptr.readlines()
19         print(content)
20
21
22 def main():
23     read_with_readline()
24     read_with_readlines()
25
26
27 if __name__ == '__main__':
28     main()
```

PS D:\Python-labs\lab-1\prog> python3 3-4_read.py

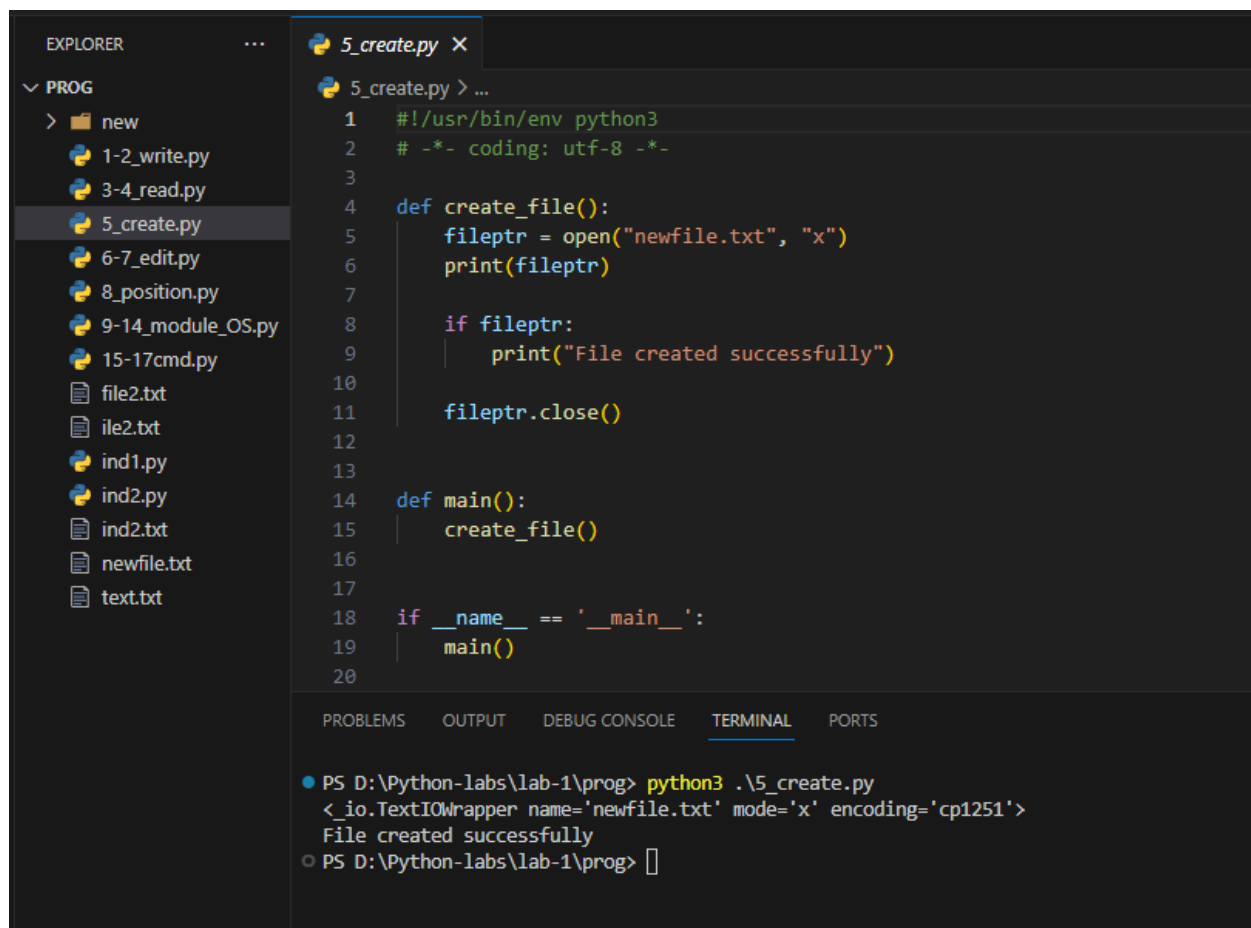
Python is the modern day language. It makes things so simple.

It is the fastest-growing programming language Python has an easy syntax and user-friendly interaction.

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

Рисунок 2 – Результат выполнения программы 3-4_read.py

3) Написал программу (5_create.py), в которой проработал примеры раздела методических указаний «Создание нового файла»:



```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 def create_file():
5     fileptr = open("newfile.txt", "x")
6     print(fileptr)
7
8     if fileptr:
9         print("File created successfully")
10
11     fileptr.close()
12
13
14 def main():
15     create_file()
16
17
18 if __name__ == '__main__':
19     main()
20
21
22 if __name__ == '__main__':
23     main()
```

PS D:\Python-labs\lab-1\prog> python3 5_create.py

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

File created successfully

PS D:\Python-labs\lab-1\prog>

Рисунок 3 – Результат выполнения программы 5_create.py

4) Написал программу (6-7_edit.py), в которой проработал примеры раздела методических указаний «Изменение кодировки файла»:

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 def edit_utf():
5     with open("text.txt", "w", encoding="utf-8") as fileptr:
6         print(
7             "UTF-8 is a variable-width character encoding used for electronic communication.",
8             file=fileptr
9         )
10        print(
11            "UTF-8 is capable of encoding all 1,112,064 valid character code points.",
12            file=fileptr
13        )
14        print(
15            "In Unicode using one to four one-byte (8-bit) code units.",
16            file=fileptr
17        )
18
19
20 def find_sentences():
21     with open("text.txt", "r", encoding="utf-8") as fileptr:
22         sentences = fileptr.readlines()
23
24     for sentence in sentences:
25         if ',' in sentence:
26             print(sentence)
27
28
29 def main():
30     edit_utf()
31     find_sentences()
32
33
34 if __name__ == '__main__':
35     main()
36
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS

● PS D:\Python-labs\lab-1\prog> python3 .\6-7_edit.py
UTF-8 is capable of encoding all 1,112,064 valid character code points.

○ PS D:\Python-labs\lab-1\prog> []

Рисунок 4 – Результат выполнения программы 6-7_edit.py

5) Написал программу (8_position.py), в которой проработал примеры раздела методических указаний «Позиция указателя файла»:

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 def pointer():
5     with open("file2.txt", "r") as fileptr:
6         print("The filepointer is at byte :", fileptr.tell())
7         fileptr.seek(10);
8         print("After reading, the filepointer is at:", fileptr.tell())
9
10
11 def main():
12     pointer()
13
14
15 if __name__ == '__main__':
16     main()
17
```

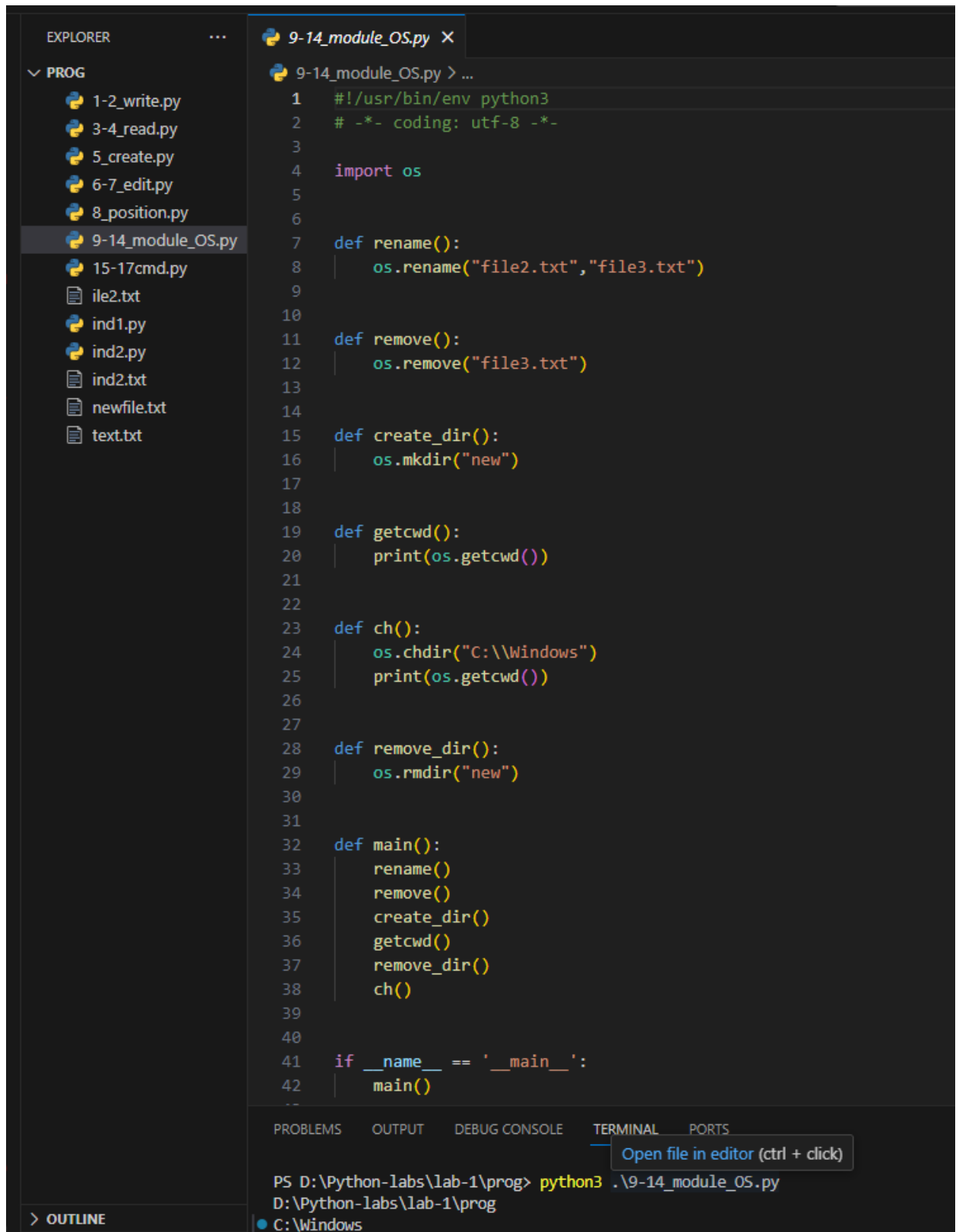
PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS

● PS D:\Python-labs\lab-1\prog> python3 .\8_position.py
The filepointer is at byte : 0
After reading, the filepointer is at: 10

○ PS D:\Python-labs\lab-1\prog> []

Рисунок 5 – Результат выполнения программы 8_position.py

6) Написал программу (9-14_module_OS.py), в которой проработал примеры раздела методических указаний «Модуль OS»:



The screenshot displays a code editor with a dark theme. On the left, the 'EXPLORER' sidebar shows a project structure under 'PROG' with files like 1-2_write.py, 3-4_read.py, 5_create.py, 6-7_edit.py, 8_position.py, 9-14_module_OS.py (selected), 15-17cmd.py, and various text files. The main editor window shows the code for 9-14_module_OS.py, which includes imports, function definitions for file operations, and a main function. The bottom panel shows the 'TERMINAL' tab with the command to run the script and the current directory path.

```
9-14_module_OS.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import os
5
6
7  def rename():
8      os.rename("file2.txt", "file3.txt")
9
10
11 def remove():
12     os.remove("file3.txt")
13
14
15 def create_dir():
16     os.mkdir("new")
17
18
19 def getcwd():
20     print(os.getcwd())
21
22
23 def ch():
24     os.chdir("C:\\Windows")
25     print(os.getcwd())
26
27
28 def remove_dir():
29     os.rmdir("new")
30
31
32 def main():
33     rename()
34     remove()
35     create_dir()
36     getcwd()
37     remove_dir()
38     ch()
39
40
41 if __name__ == '__main__':
42     main()
```

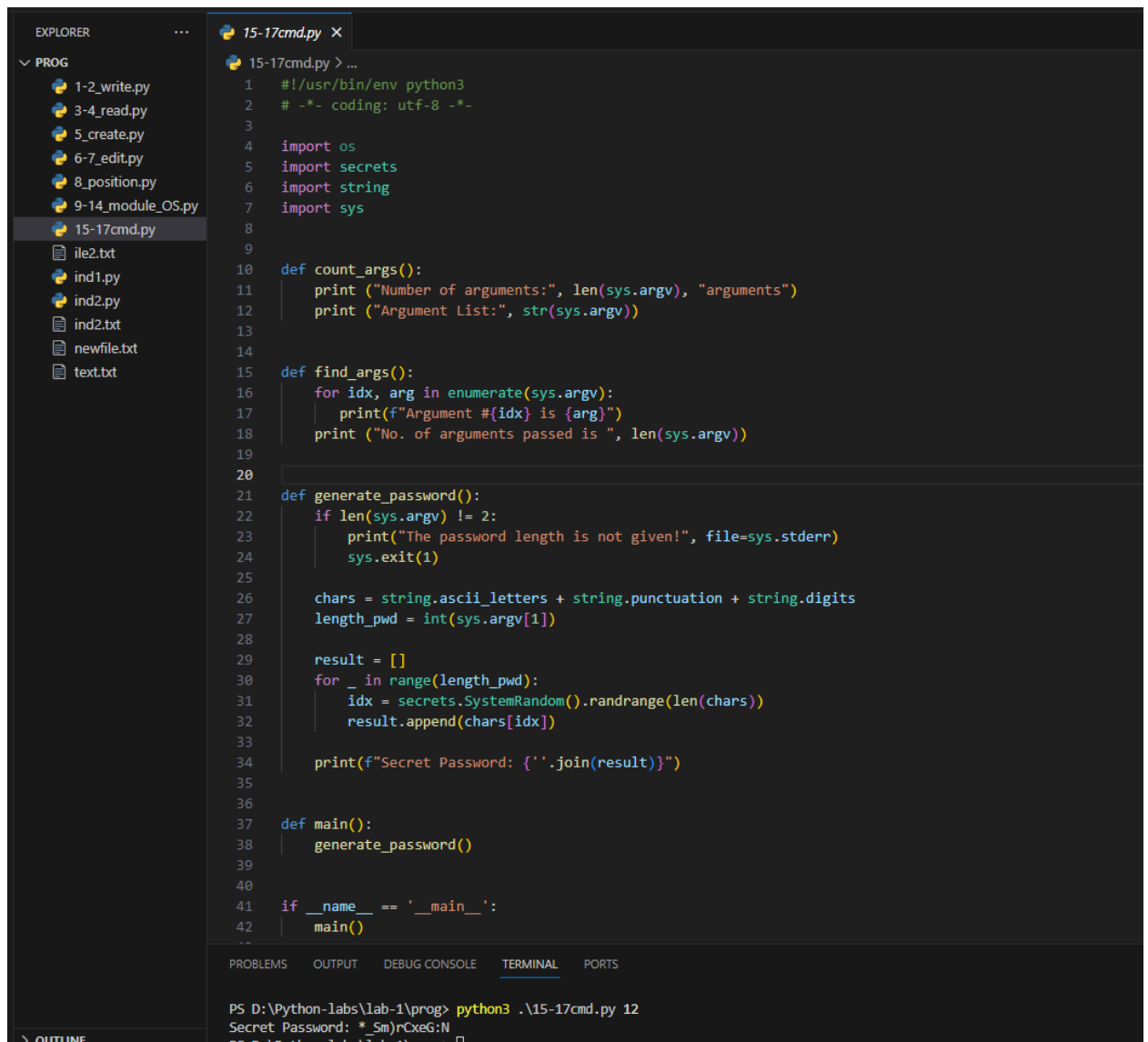
PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS

Open file in editor (ctrl + click)

PS D:\Python-labs\lab-1\prog> python3 .\9-14_module_OS.py
D:\Python-labs\lab-1\prog
● C:\Windows

Рисунок 6 – Результат выполнения программы 9-14_module_OS.py

7) Написал программу (15-17cmd.py), в которой проработал примеры раздела методических указаний «Доступ к элементам командной строки в языке программирования Python»:



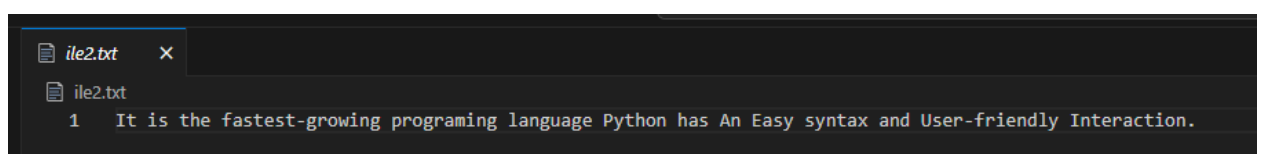
```
EXPLORER
PROG
1-2_write.py
3-4_read.py
5_create.py
6-7_edit.py
8_position.py
9-14_module_OS.py
15-17cmd.py
ile2.txt
ind1.py
ind2.py
ind2.txt
newfile.txt
text.txt

15-17cmd.py > ...
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 def count_args():
11     print ("Number of arguments:", len(sys.argv), "arguments")
12     print ("Argument List:", str(sys.argv))
13
14
15 def find_args():
16     for idx, arg in enumerate(sys.argv):
17         print(f"Argument #{idx} is {arg}")
18     print ("No. of arguments passed is ", len(sys.argv))
19
20
21 def generate_password():
22     if len(sys.argv) != 2:
23         print("The password length is not given!", file=sys.stderr)
24         sys.exit(1)
25
26     chars = string.ascii_letters + string.punctuation + string.digits
27     length_pwd = int(sys.argv[1])
28
29     result = []
30     for _ in range(length_pwd):
31         idx = secrets.SystemRandom().randrange(len(chars))
32         result.append(chars[idx])
33
34     print(f"Secret Password: {''.join(result)}")
35
36
37 def main():
38     generate_password()
39
40
41 if __name__ == '__main__':
42     main()
43
44
45 PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
46
47 PS D:\Python-labs\lab-1\prog> python3 .\15-17cmd.py 12
48 Secret Password: *_5m)rCxeG:N
49 PS D:\Python-labs\lab-1\prog> []
```

Рисунок 7 – Результат выполнения программы

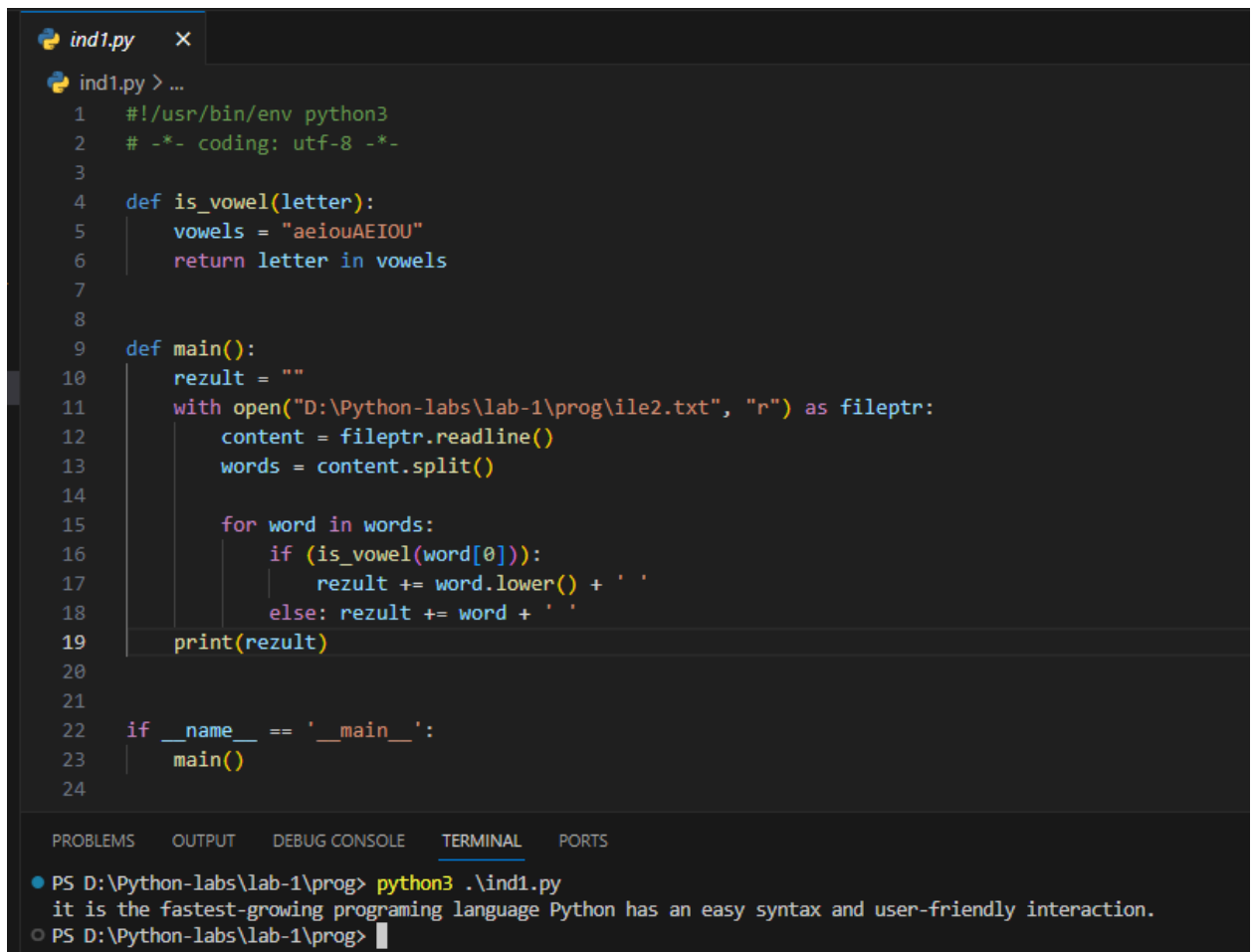
Индивидуальное задание (12 Вариант):

1 задание. Написать программу, которая считывает английский текст из файла и выводит его на экран, заменив каждую первую букву слов, начинающихся с гласной буквы, на прописную.



```
ile2.txt
ile2.txt
1 It is the fastest-growing programing language Python has An Easy syntax and User-friendly Interaction.
```

Рисунок 8 – Входные данные



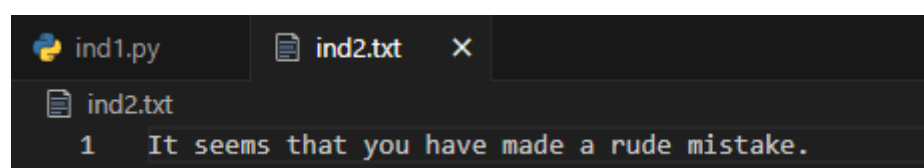
```
ind1.py x
ind1.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  def is_vowel(letter):
5      vowels = "aeiouAEIOU"
6      return letter in vowels
7
8
9  def main():
10     rezult = ""
11     with open("D:\Python-labs\lab-1\prog\ile2.txt", "r") as fileptr:
12         content = fileptr.readline()
13         words = content.split()
14
15         for word in words:
16             if (is_vowel(word[0])):
17                 rezult += word.lower() + ' '
18             else: rezult += word + ' '
19     print(rezult)
20
21
22 if __name__ == '__main__':
23     main()
24
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS D:\Python-labs\lab-1\prog> python3 .\ind1.py
it is the fastest-growing programing language Python has an easy syntax and user-friendly interaction.
PS D:\Python-labs\lab-1\prog>

Рисунок 9 – Результат выполнения программы ind1.py

2 Задание. Автоматическая проверка орфографии не помешала бы многим из нас. В данном упражнении мы напишем простую программу, сверяющую слова из текстового файла со словарем. Неправильно написанными будем считать все слова, которых не нашлось в словаре. Имя файла, в котором требуется выполнить орфографическую проверку, пользователь должен передать при помощи аргумента командной строки. В случае отсутствия аргумента должна выдаваться соответствующая ошибка. Сообщение об ошибке также должно появляться, если не удастся открыть указанный пользователем файл. Также Вам следует игнорировать регистр символов при выполнении проверки.



```
ind1.py ind2.txt x
ind2.txt
1  It seems that you have made a rude mistake.
```

Рисунок 10 – Входные данные

```
ind1.py ind2.txt ind2.py x
ind2.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6
7  def main():
8      if len(sys.argv) != 2:
9          print("Не указан путь!", file=sys.stderr)
10         sys.exit(1)
11
12         dictionary = ["it", "seems", "a", "rude"]
13         result = []
14         with open(sys.argv[1], "r") as fileptr:
15             content = fileptr.readline()
16             words = content.split()
17
18             for word in words:
19                 if (word.lower() not in dictionary):
20                     result.append(word)
21         print(result)
22
23
24 if __name__ == '__main__':
25     main()
26
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
● PS D:\Python-labs\lab-1\prog> python3 .\ind2.py .\ind2.txt
['that', 'you', 'have', 'made', 'mistake.']
○ PS D:\Python-labs\lab-1\prog>
```

Рисунок 11 – Результат выполнения программы ind2.py

Вывод: в ходе выполнения работы были приобретены навыки по работе с текстовыми файлами при написании программ с помощью языка программирования Python версии 3.x, также были изучены основные методы модуля os для работы с файловой системой, получение аргументов командной строки.

Контрольная работа

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

Чтобы открыть файл только для чтения в Python, вы можете использовать функцию open() с режимом 'r'. Пример кода:

```
f = open('text.txt', 'r')
```


Здесь 'text.txt' - это имя файла, который вы хотите открыть, а 'r' - это режим, указывающий, что файл открывается только для чтения.

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

Чтобы открыть файл только для записи в Python, вы можете использовать функцию `open()` с режимом 'w'. Пример кода:

```
f = open('text.txt', 'w')
```

Здесь 'text.txt' - это имя файла, который вы хотите открыть, а 'w' - это режим, указывающий, что файл открывается только для записи.

3. Как прочитать данные из файла в языке Python?

Чтобы прочитать данные из файла в Python, вы можете использовать метод `read()`. Пример кода:

```
f = open('text.txt', 'r')
data = f.read()
print(data)
f.close()
```

Здесь `f.read()` читает все данные из файла, а `print(data)` выводит эти данные на экран.

4. Как записать данные в файл в языке Python?

Чтобы записать данные в файл в Python, вы можете использовать метод `write()`. Пример кода:

```
f = open('text.txt', 'w')
f.write("Это пример текста")
f.close()
```

Здесь `f.write("Это пример текста")` записывает строку "Это пример текста" в файл.

5. Как закрыть файл в языке Python?

Чтобы закрыть файл в Python, вы можете использовать метод `close()`.
Пример кода:

```
f = open('text.txt', 'r')
# выполните здесь операции с файлом
```

```
f.close()
```

Здесь `f.close()` закрывает файл после выполнения всех операций 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 Python используется для обеспечения "чистоты" кода. Это контекстный менеджер, который автоматически закрывает файл после выполнения всех операций с ним, даже если в процессе возникли исключения. Пример кода:

```
with open('text.txt', 'r') as f:  
    data = f.read()  
    print(data)
```

Здесь файл автоматически закрывается после выхода из блока `with`. Конструкция `with ... as` может быть использована в других контекстах, где требуется автоматическое освобождение ресурсов после использования, например при работе 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?

Помимо методов `read()` и `write()`, существуют и другие методы для работы с файлами в Python. Например, метод `readline()` позволяет читать файл построчно, а метод `writelines()` позволяет записывать список строк в файл. Еще один полезный метод - `flush()`, который принудительно "сбрасывает" данные из буфера на диск.

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

Модуль `os` в Python предоставляет множество функций для работы с операционной системой, включая работу с файловой системой. Например, функции `os.rename()`, `os.remove()`, `os.mkdir()`, `os.rmdir()`, `os.listdir()`, `os.getcwd()`, `os.chdir()` Открытие файла только для чтения в Python:

Чтобы открыть файл только для чтения в Python, используйте функцию `open()` с параметром `'r'`. Например:

```
f = open('filename.txt', 'r')
```

Открытие файла только для записи в Python:

Для открытия файла только для записи используйте функцию `open()` с параметром `'w'`. Если файл с таким именем уже существует, его содержимое будет удалено:

```
f = open('filename.txt', 'w')
```

Чтение данных из файла в Python:

Для чтения данных из файла можно использовать методы `read()`, `readline()` или `readlines()`:

```
data = f.read() # Чтение всего файла
```

```
line = f.readline() # Чтение одной строки
```

```
lines = f.readlines() # Чтение всех строк в список
```

Запись данных в файл в Python:

Чтобы записать данные в файл, используйте метод `write()` или `writelines()` для записи строк или списков строк соответственно:

```
f.write('Some text\n') # Запись строки текста
```

```
f.writelines(['First line\n', 'Second line\n']) # Запись списка строк
```

Закрытие файла в Python:

После окончания работы с файлом его необходимо закрыть, используя метод `close()`:

```
f.close()
```

Конструкция `with ... as` в Python:

Конструкция `with ... as` используется для обеспечения корректного управления ресурсами, такими как файлы. Она автоматически закрывает файл после выхода из блока кода, что делает код более чистым и безопасным:

```
with open('filename.txt', 'r') as f:
```

```
    data = f.read()
```

Эта конструкция может использоваться не только для работы с файлами, но и для управления другими ресурсами, такими как соединения с базами данных, блокировки и т.д.

Дополнительные методы чтения/записи информации из файла:

Помимо `read()`, `readline()` и `readlines()` для чтения и `write()`, `writelines()` для записи, существуют другие методы, такие как `readinto()`, `flush()` для очистки буфера записи и `seek()` для изменения позиции в файле.

Функции модуля `os` для работы с файловой системой:

Модуль `os` в Python предоставляет множество функций для работы с файловой системой, помимо работы с файлами. Например, `os.remove()` для удаления файла, `os.rename()` для переименования, `os.walk()` для обхода директорий, `os.mkdir()` для создания новых папок и многие другие функции для работы с путями и директориями.