

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:

Пустяков Андрей Сергеевич

2 курс, группа ИВТ-б-о-22-1,

09.03.01 «Информатика и  
вычислительная техника (профиль)  
«Программное обеспечение средств  
вычислительной  
техники и автоматизированных  
систем», очная форма обучения

---

(подпись)

Руководитель практики:

Воронкин Р. А. кандидат технических  
наук, доцент, доцент кафедры  
инфокоммуникаций

---

(подпись)

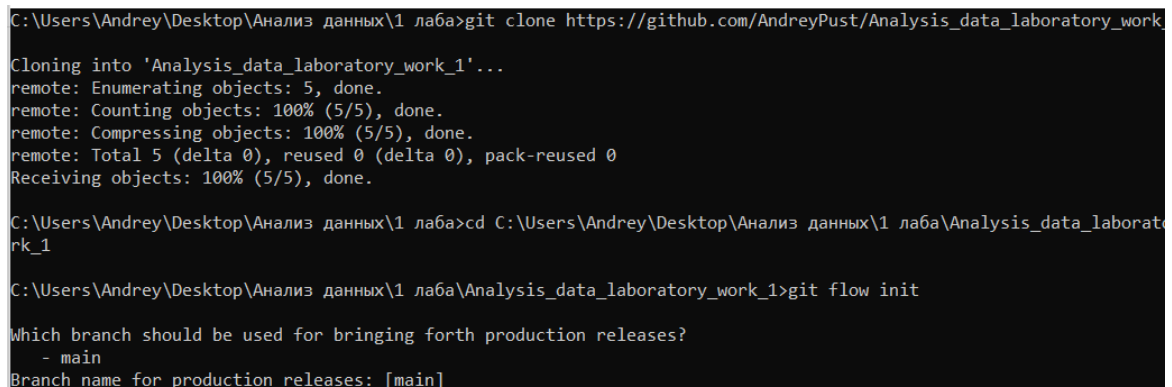
Ставрополь, 2023 г.

Тема: Работа с файлами в языке Python.

Цель: приобрести навыки по работе с текстовыми файлами при написании программ с помощью языка программирования Python версии 3.x, изучить основные методы модуля os для работы с файловой системой, получением аргументов командной строки.

Ход работы:

Создание общедоступного репозитория на «GitHub», клонирование репозитория, редактирование файла «.gitignore», организация репозитория согласно модели ветвления «git-flow» (рис. 1).



```
C:\Users\Andrey\Desktop\Анализ данных\1 лаба>git clone https://github.com/AndreyPust/Analysis_data_laboratory_work_1
Cloning into 'Analysis_data_laboratory_work_1'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.

C:\Users\Andrey\Desktop\Анализ данных\1 лаба>cd C:\Users\Andrey\Desktop\Анализ данных\1 лаба\Analysis_data_laboratory_work_1

C:\Users\Andrey\Desktop\Анализ данных\1 лаба\Analysis_data_laboratory_work_1>git flow init

Which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
```

Рисунок 1 – Организация модели ветвления «git-flow».

Проработка примеров лабораторной работы:

Пример 1.

Необходимо открыть файл с именем «file2.txt» в режиме записи в той же директории, что и модуль, а также записать в него некоторую информацию, после чего закрыть файл.

Код программы решения примера 1 с использованием функции «open» и оператора «with» (рис. 2).

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

if __name__ == "__main__":
    # open the file2.txt in append mode. Create a new file if no such file exists.
    fileptr = open("file2.txt", "w")
    # appending the content to the file
    fileptr.write(
        "Python is the modern day language. It makes things so simple.\n"
        "It is the fastest-growing programing language"
    )
    # closing the opened the file
    fileptr.close()

    # open the file2.txt in append mode. Create a new file if no such file exists.
    with open("file2.txt", "w") as fileptr:
        # appending the content to the file
        fileptr.write(
            "Python is the modern day language. It makes things so simple.\n"
            "It is the fastest-growing programing language"
        )
    )
```

Рисунок 2 – Код программы примера 1.

Результаты работы программы (содержимое файла «file2.txt») (рис. 3).

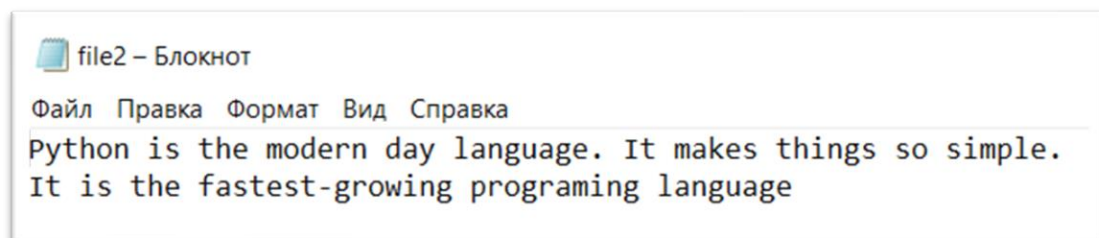


Рисунок 3 – Содержимое файла «file2.txt».

Пример 2.

Необходимо открыть файл «file2.txt» в режиме дозаписи в файл и дозаписать в него некоторую информацию.

Код программы примера 2 с использованием функции «open» и оператора «with» (рис. 4).

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

if __name__ == "__main__":
    # open the file.txt in write mode.
    fileptr = open("file2.txt", "a")

    # overwriting the content of the file
    fileptr.write(" Python has an easy syntax and user-friendly interaction.")

    # closing the opened file
    fileptr.close()

    # open the file2.txt in write mode.
    with open("file2.txt", "a") as fileptr:
        # overwriting the content of the file
        fileptr.write(" Python has an easy syntax and user-friendly interaction.")
```

Рисунок 4 – Код программы примера 2.

Результаты работы программы (содержимое файла «file2.txt» с учетом использования разных способов открытия файла) (рис. 5).

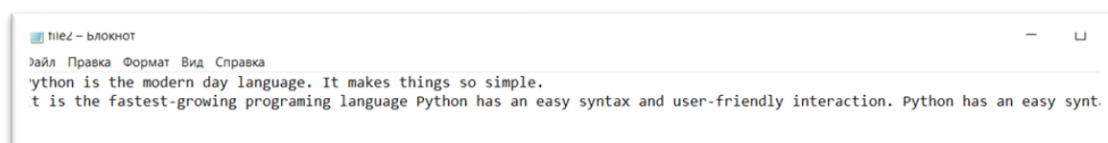


Рисунок 5 – Содержимое файла «file2.txt».

### Пример 3.

Необходимо построчно считать текстовую информацию из некоторого файла с использованием метода «readline()».

Код программы примера 3 с использованием функции «open» и оператора «with» (рис. 6).

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

if __name__ == "__main__":
    # open the file2.txt in read mode. causes error if no such file exists.
    fileptr = open("file2.txt", "r")

    # stores all the data of the file into the variable content
    content1 = fileptr.readline()
    content2 = fileptr.readline()

    # prints the content of the file
    print(content1)
    print(content2)

    # closes the opened file
    fileptr.close()

    # open the file2.txt in read mode. causes error if no such file exists.
    with open("file2.txt", "r") as fileptr:
        # stores all the data of the file into the variable content
        content1 = fileptr.readline()
        content2 = fileptr.readline()

        # prints the content of the file
        print(content1)
        print(content2)
```

Рисунок 6 – Код программы примера 2.

Результаты работы программы с учетом использования разных способов открыть файл (рис. 7).

```
"C:\Users\Andrey\Desktop\Анализ данных\1 лаба\Analysis_data_laboratory_work_1\venv\Scripts\python.exe"
Python is the modern day language. It makes things so simple.

It is the fastest-growing programing language Python has an easy syntax and user-friendly interaction.
Python is the modern day language. It makes things so simple.

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

Process finished with exit code 0
```

Рисунок 7 – Результаты работы программы примера 3.

#### Пример 4.

Необходимо получить список строк до конца некоторого файла с использованием метода «readlines()».

Код программы примера 4 с использованием функции «open» и оператора «with» (рис. 8).

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

if __name__ == "__main__":
    # open the file2.txt in read mode. causes error if no such file exists.
    fileptr = open("file2.txt", "r")

    # stores all the data of the file into the variable content
    content = fileptr.readlines()

    # prints the content of the file
    print(content)

    # closes the opened file
    fileptr.close()

    # open the file2.txt in read mode. causes error if no such file exists.
    with open("file2.txt", "r") as fileptr:
        # stores all the data of the file into the variable content
        content = fileptr.readlines()

        # prints the content of the file
        print(content)
```

Рисунок 8 – Код программы примера 4.

Результаты работы программы с учетом использования разных способов открыть файл (рис. 9).

```
"C:\Users\Andrey\Desktop\Анализ данных\1 лаба\Analysis_data_laboratory_work_1\venv\Scripts\python.exe" "C:\Users\Andrey\Desktop\Ан
['Python is the modern day language. It makes things so simple.\n', 'It is the fastest-growing programming language Python has an e
['Python is the modern day language. It makes things so simple.\n', 'It is the fastest-growing programming language Python has an e
```

Рисунок 9 – Результаты работы программы примера 4.

### Пример 5.

Необходимо создать новый файл с использованием вида доступа «x» функции «open()».

Код программы примера 5 с использованием функции «open» и оператора «with» (рис. 10).

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

if __name__ == "__main__":
    # open the newfile.txt in read mode. causes error if no such file exists.
    fileptr = open("newfile.txt", "x")
    print(fileptr)

    if fileptr:
        print("File created successfully")

    # closes the opened file
    fileptr.close()

    # open the newfile.txt in read mode. causes error if no such file exists.
    with open("newfile.txt", "x") as fileptr:
        print(fileptr)

    if fileptr:
        print("File created successfully")
```

Рисунок 10 – Код программы примера 5.

Результаты работы программы (оба файла созданы) (рис. 11).

```
"C:\Users\Andrey\Desktop\Анализ данных\1 лаба\Analysis_data_laboratory
<_io.TextIOWrapper name='newfile.txt' mode='x' encoding='cp1251'>
File created successfully
<_io.TextIOWrapper name='newfile2.txt' mode='x' encoding='cp1251'>
File created successfully
```

Рисунок 11 – Результаты работы программы примера 5.

### Пример 6.

Необходимо осуществить запись в файл с использованием кодировки «utf-8».

Код программы примера 5 (рис. 12).

```
#!/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", "w", 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 12 – Код программы примера 6.

Результаты работы программы (содержимое файла «text.txt» с указанной кодировкой) (рис. 13).

```
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 13 – Содержимое файла «text.txt».

### Пример 7.

Необходимо написать программу, которая считывает текст из файла и выводит на экран только предложения, содержащие запятые. Каждое предложение в файле записано на отдельной строке.

Код программы примера 7 для решения данной задачи (рис. 14).



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

if __name__ == "__main__":
    with open("text.txt", "r", encoding="utf-8") as fileptr:
        sentences = fileptr.readlines()
        # Вывод предложений с запятыми.
        for sentence in sentences:
            if "," in sentence:
                print(sentence)
```

Рисунок 14 – Код программы примера 7.

Результаты работы программы примера 7 (рис. 15).

```
"C:\Users\Andrey\Desktop\Анализ данных\1 лаба\Analysis_data_laboratory_work_1\
UTF-8 is capable of encoding all 1,112,064 valid character code points.

Process finished with exit code 0
```

Рисунок 15 – Результаты работы программы.

Пример 8.

Необходимо поменять указатель в файле на некоторые позиции и выяснить на каких позициях находится указатель в разных частях программы.

Код программы примера 8 для перемещения указателя и получения его позиции в файле (рис. 16).

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

if __name__ == "__main__":
    # open the file file2.txt in read mode
    with open("file2.txt", "r") as fileptr:
        # initially the filepointer is at 0
        print("The filepointer is at byte :", fileptr.tell())

        # changing the file pointer location to 10.
        fileptr.seek(10)

        # tell() returns the location of the fileptr.
        print("After reading, the filepointer is at:", fileptr.tell())
```

Рисунок 16 – Код программы примера 8.

Результаты работы программы примера 8 (рис. 17).

```
"C:\Users\Andrey\Desktop\Анализ данных\1 лаба\Analysis_data_laboratory_wor
The filepointer is at byte : 0
After reading, the filepointer is at: 10

Process finished with exit code 0
```

Рисунок 17 – Результаты работы программы.

Пример 9.

Необходимо, используя модуль «os», переименовать файл «file2.txt» в файл «file3.txt» в той попытке, что и модуль программы.

Код программы решения данной задачи (рис. 18).

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

import os

if __name__ == "__main__":
    # rename file2.txt to file3.txt
    os.rename(src="file2.txt", dst="file3.txt")
```

Рисунок 18 – Код программы примера 9.

Результаты работы программы (переименованный файл в той же директории, что и модуль программы) (рис. 19).



|   |                  |                   |
|---|------------------|-------------------|
|  example_9 | 13.02.2024 12:23 | Python File       |
|  file3     | 13.02.2024 9:21  | Текстовый докум.. |

Рисунок 19 – Содержимое каталога «examples».

Пример 10.

Необходимо удалить конкретный файл, находящийся в той же директории, что и модуль программы.

Код программы решения данной задачи (рис. 20).

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

import os

if __name__ == "__main__":
    # deleting the file named file3.txt
    os.remove("file3.txt")
```

Рисунок 20 – Код программы примера 10.

Результаты работы программы (содержимое директории «examples» и отсутствие в ней файла «file3.txt») (рис. 21).

| Имя        | Дата изменения   |
|------------|------------------|
| example_1  | 12.02.2024 22:22 |
| example_2  | 12.02.2024 22:23 |
| example_3  | 13.02.2024 9:25  |
| example_4  | 13.02.2024 9:33  |
| example_5  | 13.02.2024 9:47  |
| example_6  | 13.02.2024 10:04 |
| example_7  | 13.02.2024 11:55 |
| example_8  | 13.02.2024 12:17 |
| example_9  | 13.02.2024 12:23 |
| example_10 | 13.02.2024 12:34 |
| newfile    | 13.02.2024 9:47  |
| newfile2   | 13.02.2024 9:47  |
| text       | 13.02.2024 10:04 |

Рисунок 21 – Содержимое каталога «examples».

### Пример 11.

Необходимо в том же каталоге, что модуль программы создать новый каталог с именем «new».

Код программы для создания нового каталога (рис. 22).

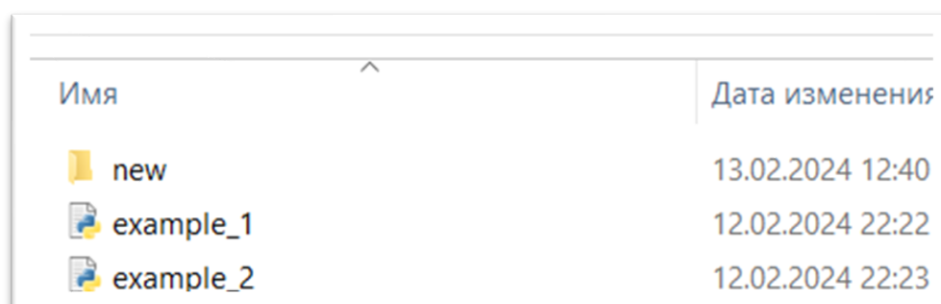
```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import os

if __name__ == "__main__":
    # creating a new directory with the name new
    os.mkdir("new")
```

Рисунок 22 – Код программы примера 11.

Результаты работы программы (содержащийся в данном каталоге новый каталог «new») (рис. 23).



| Имя       | Дата изменения   |
|-----------|------------------|
| new       | 13.02.2024 12:40 |
| example_1 | 12.02.2024 22:22 |
| example_2 | 12.02.2024 22:23 |

Рисунок 23 – Содержимое каталога «examples».

### Пример 12.

Необходимо получить полный путь текущего каталога (каталога, в котором содержится данный модуль).

Код программы для получение полного пути текущего каталога (рис. 24).

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

import os

if __name__ == "__main__":
    path = os.getcwd()
    print("Текущий каталог - ", path)
```

Рисунок 24 – Код программы примера 12.

Результаты работы программы примера №12 (полный путь текущего каталога) (рис. 25).

```
"C:\Users\Andrey\Desktop\Анализ данных\1 лаба\Analysis_data_laboratory_work_1\venv\Scripts\python.exe" "C:\Users\Andrey\Desktop\Анализ данных\1 лаба\Analysis_data_laboratory_work_1\examples
Текущий каталог - C:\Users\Andrey\Desktop\Анализ данных\1 лаба\Analysis_data_laboratory_work_1\examples

Process finished with exit code 0
```

Рисунок 24 – Результаты работы программы примера №12.

Пример 13.

Необходимо изменить текущий рабочий каталог на другой каталог.

Код программы, позволяющий сменить текущий рабочий каталог (рис. 26).

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

import os

if __name__ == "__main__":
    # Changing current directory with the new directory
    os.chdir("C:\\Windows")
    # It will display the current working directory
    print("Текущий рабочий каталог - ", os.getcwd())
```

Рисунок 26 – Код программы примера №13.

Результаты работы программы (вывод полного пути текущего каталога, после его смены) (рис. 27).

```
"C:\Users\Andrey\Desktop\Анализ данных\1 лаба\Analysis_data_laboratory_work_1\
Текущий рабочий каталог - C:\Windows
```

Рисунок 27 – Результаты работы программы примера №13.

#### Пример 14.

Необходимо удалить каталог в текущем каталоге с именем «new».

Код программы для удаления каталога в текущем каталоге (рис. 28).

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

import os

if __name__ == "__main__":
    # removing the new directory
    os.rmdir("new")
```

Рисунок 28 – Код программы примера №14.

Результаты работы программы примера №14 (содержимое текущего каталога) (рис. 29).


















| <a href="#">Главная</a> / <a href="#">Анализ данных</a> / <a href="#">Лабораторная работа №1</a> / <a href="#">Примеры</a> |                  |                 | <input type="text"/> |
|--|------------------|-----------------|----------------------|
| Имя  | Дата изменения   | Тип             |                      |
|  example_1                              | 12.02.2024 22:22 | Python File     |                      |
|  example_2                              | 12.02.2024 22:23 | Python File     |                      |
|  example_3                              | 13.02.2024 9:25  | Python File     |                      |
|  example_4                              | 13.02.2024 9:33  | Python File     |                      |
|  example_5                              | 13.02.2024 9:47  | Python File     |                      |
|  example_6                              | 13.02.2024 10:04 | Python File     |                      |
|  example_7                              | 13.02.2024 11:55 | Python File     |                      |
|  example_8                              | 13.02.2024 12:17 | Python File     |                      |
|  example_9                              | 13.02.2024 12:23 | Python File     |                      |
|  example_10                             | 13.02.2024 12:34 | Python File     |                      |
|  example_12                             | 18.02.2024 15:43 | Python File     |                      |
|  example_13                             | 18.02.2024 15:51 | Python File     |                      |
|  example_14                             | 18.02.2024 15:56 | Python File     |                      |
|  example_11                             | 13.02.2024 12:40 | Python File     |                      |
|  newfile                                | 13.02.2024 9:47  | Текстовый докум |                      |
|  newfile2                               | 13.02.2024 9:47  | Текстовый докум |                      |
|  text                                   | 13.02.2024 10:04 | Текстовый докум |                      |

Рисунок 29 – Содержимое текущего каталога.



### Пример 15.

Необходимо посчитать количество аргументов, которые были переданы командной строке.

Код программы примера №15 (рис. 30).

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

import sys

if __name__ == "__main__":
    print("Number of arguments:", len(sys.argv), "arguments")
    print("Argument List:", str(sys.argv))
```

Рисунок 30 – Код программы примера №15.

Результаты работы программы примера №15 (рис. 31).

```
C:\Users\Andrey\Desktop\Анализ данных\1 лаба\Analysis_data_laboratory_work_1\examples>python example_15.py arg1 arg2 arg3
Number of arguments: 4 arguments
Argument List: ['example_15.py', 'arg1', 'arg2', 'arg3']
```

Рисунок 31 – Результаты работы программы примера №15 после передачи аргументов.

### Пример 16.

Необходимо осуществить вывод переданных аргументов в командной строке.

Код программы примера №16 (рис. 32).

```
#!/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))
```

Рисунок 32 – Код программы примера №16.

Результаты работы программы (вывод переданных аргументов) (рис. 33).

```
C:\Users\Andrey\Desktop\Анализ данных\1 лаба\Analysis_data_laboratory_work_1\examples>python example_16.py Knowledge Hut 21
Argument #0 is example_16.py
Argument #1 is Knowledge
Argument #2 is Hut
Argument #3 is 21
No. of arguments passed is 4
```

Рисунок 33 – Результаты работы программы примера №16.

### Пример 17.

Необходимо создать программу для генерации пароля заданной длины. Длина пароля должна передаваться как аргумент командной строки сценария.

Код программы примера №17 для генерации пароля заданной длины (рис. 34).

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

import os
import secrets
import string
import sys

if __name__ == "__main__":
    if len(sys.argv) != 2:
        print("The password length is not given!", file=sys.stderr)
        sys.exit(1)

    chars = string.ascii_letters + string.punctuation + string.digits
    length_pwd = int(sys.argv[1])

    result = []
    for _ in range(length_pwd):
        idx = secrets.SystemRandom().randrange(len(chars))
        result.append(chars[idx])

    print(f"Secret Password: {''.join(result)}")
```

Рисунок 34 – Код программы примера №17.

Результаты работы программы (сгенерированный пароль при передаче сценарию в качестве аргументов параметр «20») (рис. 35).

```
C:\Users\Andrey\Desktop\Анализ данных\1 лаба\Analysis_data_laboratory_work_1\examples>python example_17.py 20
Secret Password: W+>U$Yw[2ao%sH1I''Fu
```

Рисунок 35 – Результаты работы программы примера №17.

### Выполнение индивидуальных заданий:

#### Задание 1.

Необходимо написать программу, которая считывает текст из текстового файла и определяет сколько в нем слов, состоящих из не менее чем из семи букв (Вариант 26 (7)).

Код программы для подсчета слов индивидуального задания (рис. 36).

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

if __name__ == "__main__":
    with (open("text.txt", "r", encoding="utf-8") as file):
        all_text = file.read()
        words = all_text.split()
        count_words = 0 # счетчик для слов из 7-ми и более букв

        for word in words:
            count_letter = 0 # счетчик для букв в слове

            # Посчитаем количество букв в каждом слове без учета прочих символов.
            for letter in word:
                if (letter != "," and letter != "." and letter != ";" and letter != ":"
                    and letter != "-" and letter != "!" and letter != "?"
                    and letter != "(" and letter != ")"):
                    count_letter += 1

            if count_letter >= 7:
                count_words += 1

    print("Количество слов из не менее чем семи букв в тексте: ", count_words)
```

Рисунок 36 – Код программы индивидуального задания 1.

Содержимое файла, в котором будет вестись подсчет (количество слов в нем из не менее чем 7 букв = 20) (рис. 37).

text – Блокнот

Файл Правка Формат Вид Справка

В начале июля в Москве распространялись все более и более тревожные слухи о ходе войны: говорили о воззвании государя к народу, о приезде самого государя из армии в Москву. И так как до 11 го июля манифест и воззвание не были получены, то о них и о положении России ходили преувеличенные слухи. Говорили, что государь уезжает потому, что армия в опасности, говорили, что Смоленск сдан, что у Наполеона миллион войска и что только чудо может спасти Россию.

Рисунок 37 – Содержимое файла «text».

Результаты работы программы индивидуального задания 1 (рис. 38).

```
"C:\Users\Andrey\Desktop\Анализ данных\1 лаба\Analysis_data_laboratory_
Количество слов из не менее чем семи букв в тексте: 20

Process finished with exit code 0
```

Рисунок 38 – Результаты работы программы.

## Задание 2.

Необходимо составить программу с использованием текстовых файлов.

Необходимо написать программу, которая считывает список слов из файла и собирает статистику о том, в каком проценте слов используется каждая буква алфавита. Необходимо вывести результат для всех слов и отметить ту, которая встречалась в словах наиболее редко. Знаки препинания и регистр букв должны игнорироваться (Вариант 26 (11)).

Код программы индивидуального задания 2 (рис. 39).

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

if __name__ == "__main__":
    with (open("text.txt", "r", encoding="utf-8") as file):
        all_text = file.read()
        all_text = all_text.lower()
        words = all_text.split()

        # Создадим словарь со всеми буквами алфавита.
        dictionary_letters = {
            'a': 0, 'b': 0, 'c': 0, 'd': 0, 'e': 0, 'f': 0, 'g': 0, 'h': 0,
            'i': 0,
            'j': 0, 'k': 0, 'l': 0, 'm': 0, 'n': 0, 'o': 0, 'p': 0, 'q': 0,
            'r': 0,
            's': 0, 't': 0, 'u': 0, 'v': 0, 'w': 0, 'x': 0, 'y': 0, 'z': 0
        }

        # Посчитаем количество слов для каждой буквы.
        for word in words:
            for letter in word:
                if letter in dictionary_letters:
                    dictionary_letters[letter] += 1

        # Получим информацию о букве, которая встретилась в наименьшем
        количестве слов.
        min_letter = 'a'
        for i in dictionary_letters:
            if dictionary_letters[i] <= dictionary_letters[min_letter]:
                min_letter = i

        # Выведем полученную информацию.
        len_text = len(words)
        for i in dictionary_letters:
            print("Буква", i, "встретилась в тексте в",
                  f"({dictionary_letters[i] / len_text) * 100:.{3}f}", "%",
                  "слов.")

        print("Буква, которая встретилась в тексте меньше всего:",
              min_letter)
```

Рисунок 39 – Код программы индивидуального задания 2.

Содержимое файла, с которым работает данная программа (рис. 40).

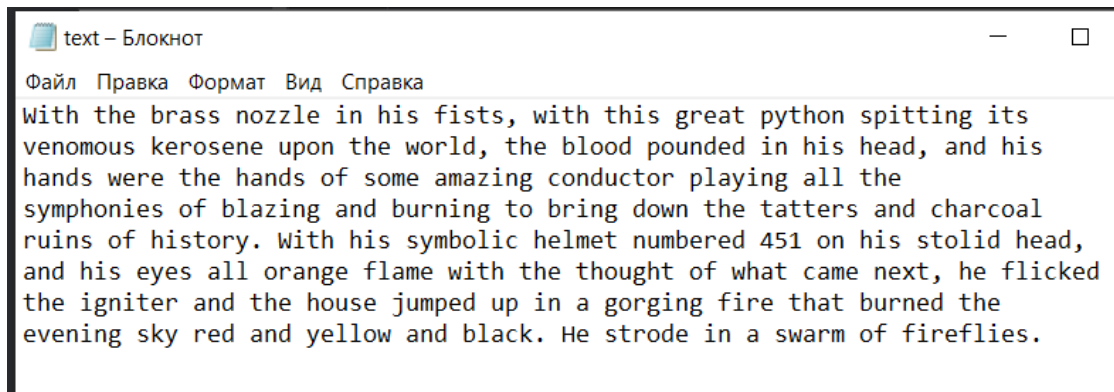


Рисунок 40 – Содержимое файла «text.txt».

Результаты работы программы индивидуального задания 2 (рис. 41).

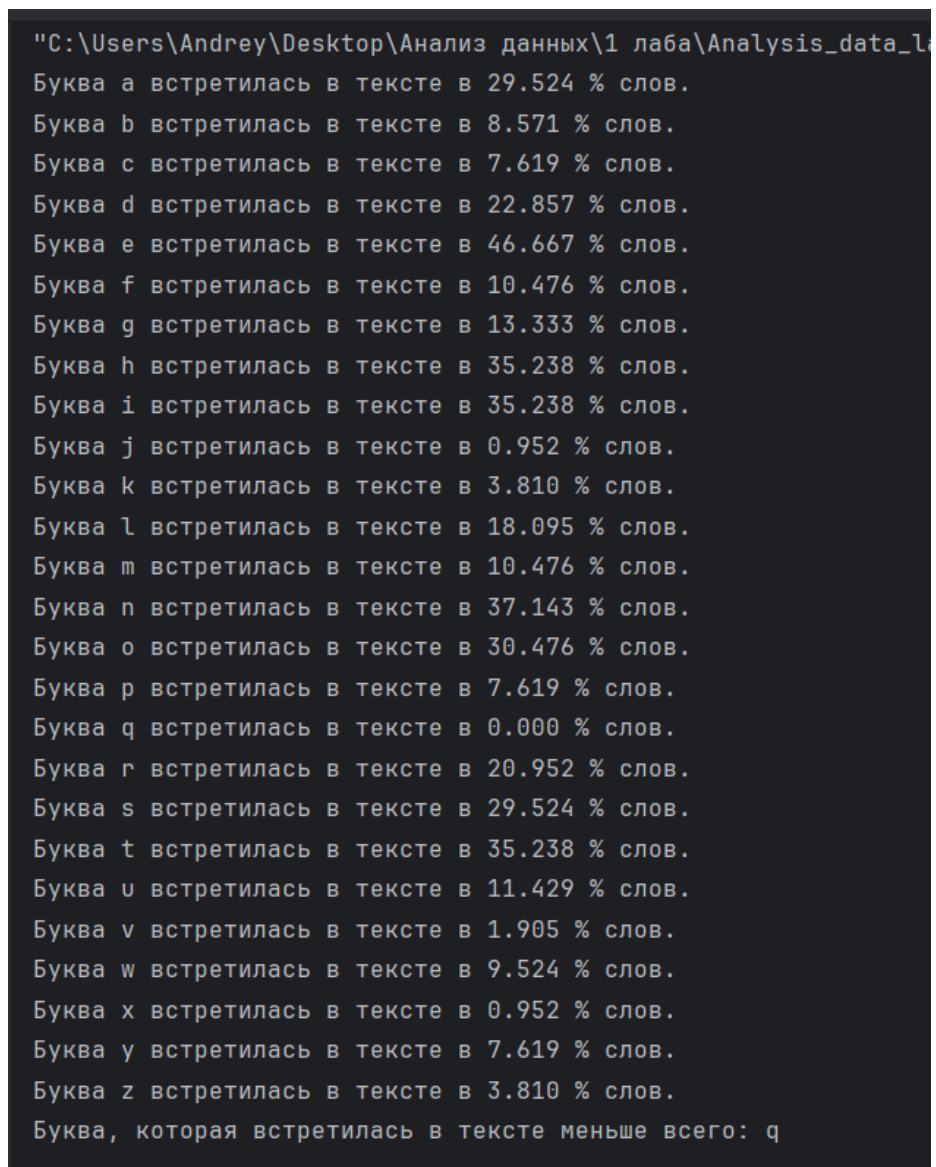


Рисунок 41 – Результаты работы программы.

### Задание 3.

Необходимо подобрать или придумать задание для работы с модулем «OS».

Необходимо пройтись по всем каталогам заданного каталога и вывести все содержимое всех каталогов и подкаталогов. Функция walk() модуля «os» позволяет проходиться по всем каталогам определенной директории.

Код программы индивидуального задания 3 для получения информации о каталогах и подкаталогах (рис. 42).

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

import os

if __name__ == "__main__":
    directory = input("Введите полный путь каталога, содержимое которого необходимо просмотреть: ")
    all_files = os.walk(directory) # проход по всем файлам и подкаталогам

    for catalog in all_files:
        print("Папка", catalog[0], "содержит:")
        print(f'Директории: {" ".join([folder for folder in catalog[1]])}')
        print(f'Файлы: {" ".join([file for file in catalog[2]])}')
        print('-' * 40)
```

Рисунок 42 – Код программы индивидуального задания 3.

Результаты работы программы для каталога «C:\Users\Andrey\Desktop\Анализ данных\1 лаба» (рис. 43).

```
"C:\Users\Andrey\Desktop\Анализ данных\1 лаба\Analysis_data_laboratory_work_1\venv\Scripts\python.exe" "C:\Users\Andrey\Desktop\Анализ данных\1 лаба\Analysis_data_laboratory_work_1\venv\Scripts\python.exe"
Введите полный путь каталога, содержимое которого необходимо просмотреть:C:\Users\Andrey\Desktop\Анализ данных\1 лаба
Папка C:\Users\Andrey\Desktop\Анализ данных\1 лаба содержит:
Директории: Analysis_data_laboratory_work_1
Файлы: ~$чет по ПР 2.15 Пустяков.docx, ~WRL0740.tmp, Лабораторная работа 2.15 (1).pdf, Отчет по ПР 2.15 Пустяков.docx
-----
Папка C:\Users\Andrey\Desktop\Анализ данных\1 лаба\Analysis_data_laboratory_work_1 содержит:
Директории: .git, .idea, examples, individual, venv
Файлы: .gitignore, LICENSE, README.md
-----
Папка C:\Users\Andrey\Desktop\Анализ данных\1 лаба\Analysis_data_laboratory_work_1\.git содержит:
```

Рисунок 43 – Результаты работы программы (вывод содержимого всех подкаталогов).

### Установка виртуальных окружений:

Создание нового виртуального окружения (рис. 44).

```
(base) C:\Users\Andrey>conda create --name lab_1
Collecting package metadata (current_repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
  current version: 23.7.4
  latest version: 24.3.0

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=24.3.0

## Package Plan ##

  environment location: C:\Users\Andrey\anaconda3\envs\lab_1

Proceed ([y]/n)? y

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate lab_1
#
# To deactivate an active environment, use
#
#     $ conda deactivate

(base) C:\Users\Andrey>_
```

Рисунок 44 – Создание отдельного виртуального окружения.

Активация нового виртуального окружения (команда – `conda activate lab_1`) (рис. 45).

```
(base) C:\Users\Andrey>conda activate lab_1

(lab_1) C:\Users\Andrey>_
```

Рисунок 45 – Активация виртуального окружения.



Добавление инструмента «flake8» в данном виртуальном окружении (команда - `conda install -c conda-forge flake8`) (рис. 46).

```
(lab_1) C:\Users\Andrey>conda install -c conda-forge flake8
Collecting package metadata (current_repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
  current version: 23.7.4
  latest version: 24.3.0

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=24.3.0

## Package Plan ##

) environment location: C:\Users\Andrey\anaconda3\envs\lab_1

added / updated specs:
  - flake8

The following packages will be downloaded:


```

| package                  | build      |        |             |
|--------------------------|------------|--------|-------------|
| bzip2-1.0.8              | hcfcfb64_5 | 122 KB | conda-forge |
| ca-certificates-2024.2.2 | h56e8100_0 | 152 KB | conda-forge |

Рисунок 46 – Добавление «flake8».

Добавление «black» в виртуальное окружение (рис. 47).

```

(lab_1) C:\Users\Andrey>conda install -c conda-forge black
Collecting package metadata (current_repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
  current version: 23.7.4
  latest version: 24.3.0

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=24.3.0

flake8-7.0.0          | 108 KB      | #####
## Package Plan ##

environment location: C:\Users\Andrey\anaconda3\envs\lab_1

added / updated specs:
- black

The following packages will be downloaded:

  package                                     | build
  -----
black-24.3.0                                | py312h2e8e312_0      388 KB  conda-forge
click-8.1.7                                 | win_pyh7428d3b_0      83 KB  conda-forge
colorama-0.4.6                              | pyhd8ed1ab_0         25 KB  conda-forge
mypy_extensions-1.0.0                       | pyha770c72_0         10 KB  conda-forge
packaging-24.0                              | pyhd8ed1ab_0         49 KB  conda-forge
pathspec-0.12.1                             | pyhd8ed1ab_0         40 KB  conda-forge

```

Рисунок 47 – Добавление «black».

Добавление «isort» к виртуальному окружению (рис. 48).

```

(lab_1) C:\Users\Andrey>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: 23.7.4
  latest version: 24.3.0

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=24.3.0

## Package Plan ##

  environment location: C:\Users\Andrey\anaconda3\envs\lab_1

  added / updated specs:
    - isort

The following packages will be downloaded:



| package      | build        |       |             |
|--------------|--------------|-------|-------------|
| isort-5.13.2 | pyhd8ed1ab_0 | 72 KB | conda-forge |
| Total:       |              | 72 KB |             |



The following NEW packages will be INSTALLED:

  isort                  conda-forge/noarch::isort-5.13.2-pyhd8ed1ab_0

```

Рисунок 47 – Добавление «isort».

Проверка наличия всех пакетов в виртуальном окружении (рис. 48).

```
(lab_1) C:\Users\Andrey>conda list
# packages in environment at C:\Users\Andrey\anaconda3\envs\lab_1:
#
# Name                                Version                                Build                                Channel
black                                 24.3.0                                py312h2e8e312_0                      conda-forge
bzip2                                 1.0.8                                  hcfcfb64_5                          conda-forge
ca-certificates                      2024.2.2                              h56e8100_0                          conda-forge
click                                 8.1.7                                  win_pyh7428d3b_0                    conda-forge
colorama                             0.4.6                                  pyhd8ed1ab_0                        conda-forge
flake8                                7.0.0                                  pyhd8ed1ab_0                        conda-forge
isort                                 5.13.2                                 pyhd8ed1ab_0                        conda-forge
libexpat                             2.6.2                                  h63175ca_0                          conda-forge
libffi                                3.4.2                                  h8ffe710_5                          conda-forge
libsqlite                             3.45.2                                hcfcfb64_0                          conda-forge
libzlib                               1.2.13                                 hcfcfb64_5                          conda-forge
mccabe                                0.7.0                                  pyhd8ed1ab_0                        conda-forge
mypy_extensions                      1.0.0                                  pyha770c72_0                        conda-forge
openssl                               3.2.1                                  hcfcfb64_1                          conda-forge
packaging                             24.0                                   pyhd8ed1ab_0                        conda-forge
pathspec                              0.12.1                                 pyhd8ed1ab_0                        conda-forge
pip                                   24.0                                   pyhd8ed1ab_0                        conda-forge
platformdirs                         4.2.0                                  pyhd8ed1ab_0                        conda-forge
pycodestyle                          2.11.1                                 pyhd8ed1ab_0                        conda-forge
pyflakes                             3.2.0                                  pyhd8ed1ab_0                        conda-forge
python                               3.12.2                                h2628c8c_0_cpython                  conda-forge
python_abi                           3.12                                   4_cp312                             conda-forge
setuptools                           69.2.0                                 pyhd8ed1ab_0                        conda-forge
tk                                    8.6.13                                h5226925_1                          conda-forge
tzdata                               2024a                                  h0c530f3_0                          conda-forge
ucrt                                  10.0.22621.0                          h57928b3_0                          conda-forge
vc                                    14.3                                   hcf57466_18                         conda-forge
vc14_runtime                         14.38.33130                           h82b7239_18                         conda-forge
vs2015_runtime                       14.38.33130                           hcb4865c_18                         conda-forge
wheel                                 0.43.0                                 pyhd8ed1ab_0                        conda-forge
xz                                    5.2.6                                  h8d14728_0                          conda-forge
```

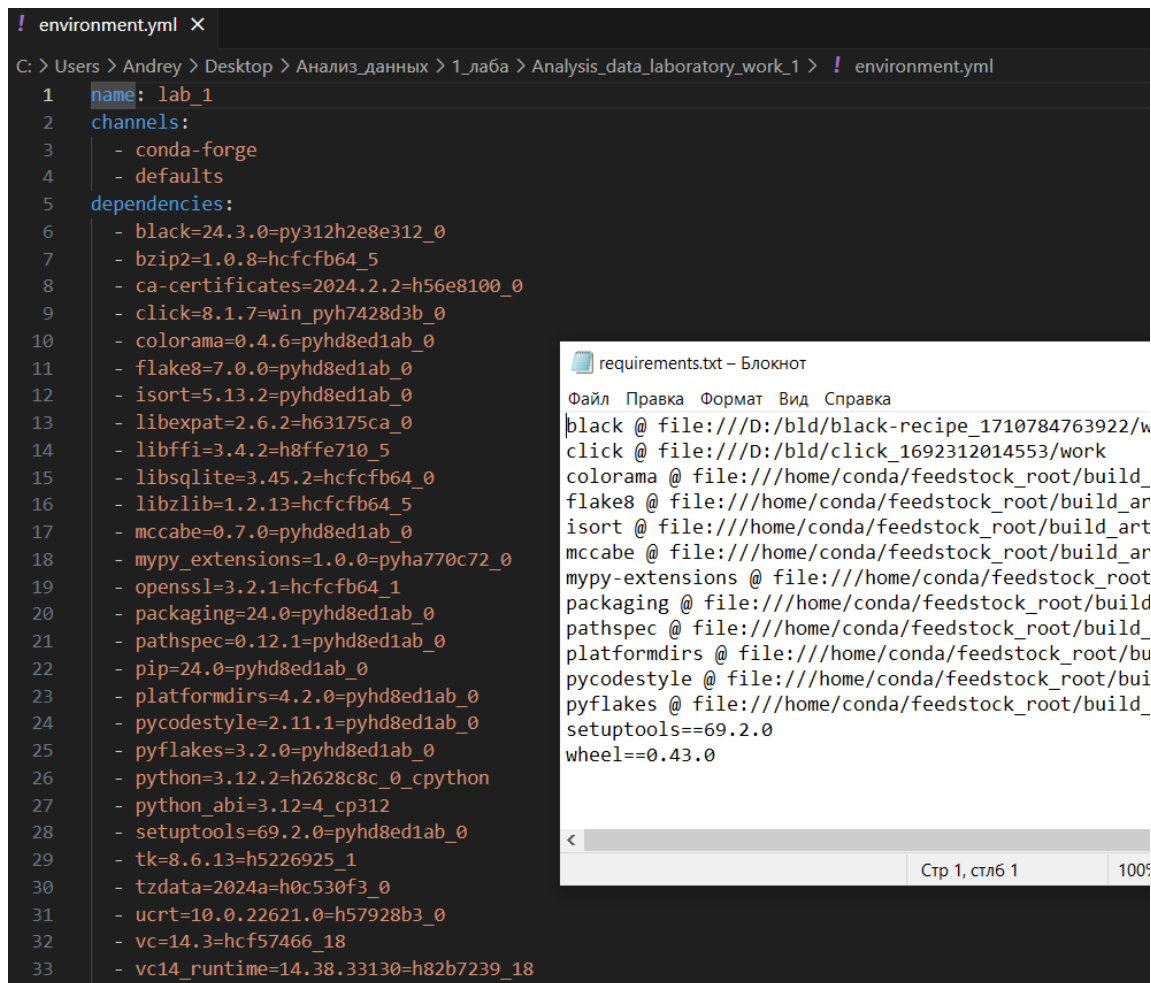
Рисунок 48 – Использование команды «conda list».

Формирование файлов «environment.yml» и «requirements.txt» (рис. 49).

```
(lab_1) C:\Users\Andrey\Desktop\Анализ_данных\1_лаба\Analysis_data_laboratory_work_1>conda env export > environment.yml
(lab_1) C:\Users\Andrey\Desktop\Анализ_данных\1_лаба\Analysis_data_laboratory_work_1>pip freeze > requirements.txt
```

Рисунок 49 – Файлы «environment.yml» и «requirements.txt».

Содержимое файлов «environment.yml» и «requirements.txt» (рис. 50).



The screenshot shows a code editor with two files open. The left pane displays `environment.yml` with the following content:

```
1 name: lab_1
2 channels:
3   - conda-forge
4   - defaults
5 dependencies:
6   - black=24.3.0=py312h2e8e312_0
7   - bzip2=1.0.8=hcfcfb64_5
8   - ca-certificates=2024.2.2=h56e8100_0
9   - click=8.1.7=win_pyh7428d3b_0
10  - colorama=0.4.6=pyhd8ed1ab_0
11  - flake8=7.0.0=pyhd8ed1ab_0
12  - isort=5.13.2=pyhd8ed1ab_0
13  - libexpat=2.6.2=h63175ca_0
14  - libffi=3.4.2=h8ffe710_5
15  - libsqlite=3.45.2=hcfcfb64_0
16  - libzlib=1.2.13=hcfcfb64_5
17  - mccabe=0.7.0=pyhd8ed1ab_0
18  - mypy_extensions=1.0.0=pyha770c72_0
19  - openssl=3.2.1=hcfcfb64_1
20  - packaging=24.0=pyhd8ed1ab_0
21  - pathspec=0.12.1=pyhd8ed1ab_0
22  - pip=24.0=pyhd8ed1ab_0
23  - platformdirs=4.2.0=pyhd8ed1ab_0
24  - pycodestyle=2.11.1=pyhd8ed1ab_0
25  - pyflakes=3.2.0=pyhd8ed1ab_0
26  - python=3.12.2=h2628c8c_0_cpython
27  - python_abi=3.12=4_cp312
28  - setuptools=69.2.0=pyhd8ed1ab_0
29  - tk=8.6.13=h5226925_1
30  - tzdata=2024a=h0c530f3_0
31  - ucrt=10.0.22621.0=h57928b3_0
32  - vc=14.3=hcfc57466_18
33  - vc14_runtime=14.38.33130=h82b7239_18
```

The right pane displays `requirements.txt` with the following content:

```
black @ file:///D:/bld/black-recipe_1710784763922/w
click @ file:///D:/bld/click_1692312014553/work
colorama @ file:///home/conda/feedstock_root/build_
flake8 @ file:///home/conda/feedstock_root/build_ar
isort @ file:///home/conda/feedstock_root/build_ar
mccabe @ file:///home/conda/feedstock_root/build_ar
mypy_extensions @ file:///home/conda/feedstock_root
packaging @ file:///home/conda/feedstock_root/build
pathspec @ file:///home/conda/feedstock_root/build_
platformdirs @ file:///home/conda/feedstock_root/bu
pycodestyle @ file:///home/conda/feedstock_root/bui
pyflakes @ file:///home/conda/feedstock_root/build_
setuptools==69.2.0
wheel==0.43.0
```

Рисунок 50 – Содержимое файлов «environment.yml» и «requirements.txt».

### Ответы на контрольные вопросы:

1. Открыть файл только для чтения позволяет режим доступа «r» в операторе «open()», а также можно не задавать режима доступа вовсе и по умолчанию файл будет открыт только для чтения. Аналогичным образом задается режим доступа и для оператора «with». Для открытия двоичного файла только для чтения используется режим доступа «rb».

2. Для того, чтобы открыть файл только для записи необходимо использовать режим доступа «w», причем указатель имеется в начале файла так как он перезаписывается, для записи двоичных файлов используется режим доступа «wb». Для дозаписи в файл необходимо использовать режим доступа «a» (чтобы не удалять его содержимое). Если файла с таким именем не существует в рабочем каталоге, то создается новый с таким именем.

3. Для чтения данных из файла используется метод «read()» с указанием количества байт, которые необходимо считать из файла. Также можно построчно считывать данные из файла используя цикл «for» и обращаясь при этом к файлу. Также можно осуществить построчное чтение данных из файла используя метод «readline()». Метод «readlines()» возвращает список строк до конца файла сразу.

4. Для записи данных в файл необходимо использовать соответствующий режим доступа («w» и «a» и их вариации) и оператор «write()».

5. Для того, чтобы закрыть файл используется метод «close()» или просто использовать оператор «with», тогда файл будет гарантированно закрыт после работы с ним.

6. Конструкция «with ... as» используется в Python также используется для работы с другими типами объектов, которые поддерживают протокол контекстного менеджера. Например, она часто применяется при работе с соединениями с базой данных, с сокетами для сетевого взаимодействия, с блокировками для синхронизации потоков и другими ресурсами, которые требуют явного освобождения после использования. Преимущества использования конструкции «with ... as» в таких случаях включают автоматическое управление ресурсами, гарантию вызова методов «enter» и «exit» объекта контекстного менеджера, а также обработку исключений внутри блока контекста.

7. Для чтения данных из CSV файлов используется оператор «csv.read()» и соответствующий модуль «csv». Модуль «pandas» используется для чтения различных форматов файлов. Для чтения файлов «json» используется соответствующий модуль и оператор «json.load». Метод «writeline» используется для записи списка строк в файл. Метод «print()» также позволяет записать информацию в файл. Метод «dump()» используется для записи информации в файл в формате JSON. Метод «writcsv()» используется для записи информации в файл в формате CSV.

8. В модуле «os» также присутствуют функции: «os.makedirs()» – создает несколько вложенных папок рекурсивно, «os.replace()» – перемещает файлы из рабочего каталога в другой каталог (также может перемещать и каталоги), «os.listdir()» – возвращает список файлов в указанном каталоге, «os.walk()» – возвращает список файлов в указанном каталоге, подкаталоги и их содержимое, «os.path.join()» – объединяет текущий путь с именем папки или файла, «os.removedirs()» – удаляет каталоги рекурсивно (только пустые каталоги). Для получения информации о файлах используется функция «os.stat()», причем содержит следующие метрики: «st\_size» – размер файла в байтах, «st\_atime» – время последнего доступа в секундах (временная метка), «st\_mtime» – время последнего изменения, «st\_ctime» – в Windows это время создания файла, а в Linux – последнего изменения метаданных.

Вывод: в ходе выполнения лабораторной работы была изучена работа с файлами, были изучены способы открытия и закрытия файлов, способы получения данных из файла и редактирования файлов (запись в файл, перезапись). Были изучены различные режимы доступа к файлам и их назначение. Был более подробно изучен модуль «os» для работы с файловой системой на языке Python. Были изучены способы получения аргументов командной строки.