

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ  
ФЕДЕРАЦИИ**

**Федеральное государственное автономное образовательное  
учреждение высшего образования  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Кафедра инфокоммуникаций  
«Работа с файлами в языке Python»**

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

Выполнил студент группы ПИЖ-б-о-21-1  
Коновалова В.Н. « » 2023г.

Подпись студента\_\_\_\_\_

Работа защищена « »\_\_\_\_\_2023г.

Проверил Воронкин Р.А. \_\_\_\_\_

(подпись)

Ставрополь 2023

**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.

**Выполнение работы:**

1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия IT и язык программирования Python.
3. Выполните клонирование созданного репозитория.
4. Дополните файл .gitignore необходимыми правилами для работы с IDE PyCharm.
5. Организуйте свой репозиторий в соответствии с моделью ветвления git-flow.
6. Создайте проект PyCharm в папке репозитория.
7. Проработать примеры лабораторной работы.

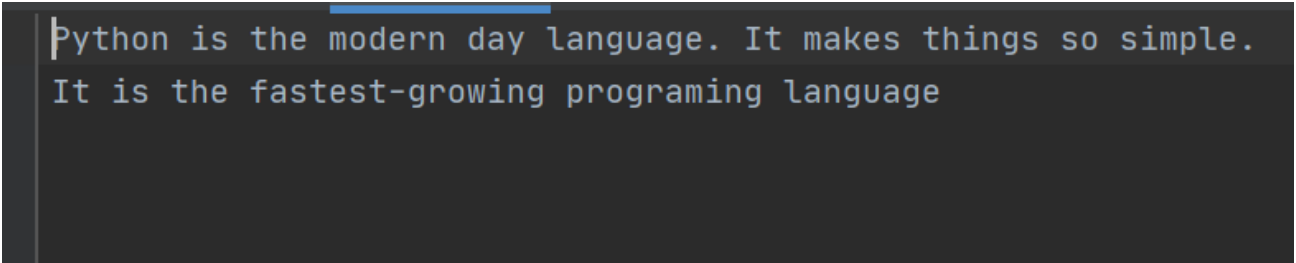
**Пример 1**

**Запись файла**

```
# 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()
```

Код с with:

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



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

Рисунок 1 – Результат работы программы

Пример 2. Запись файла.

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

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

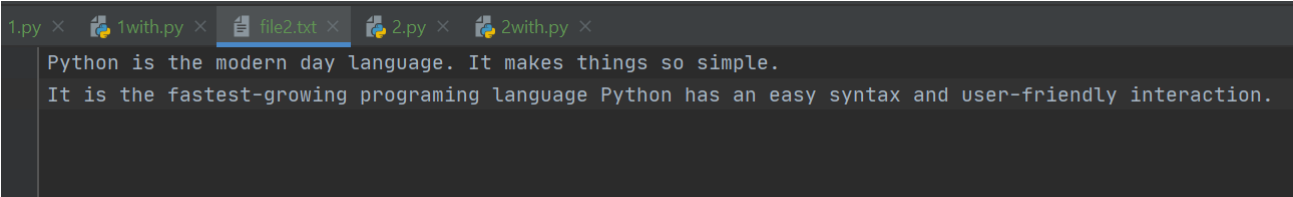
    # overwriting the content of the file
    fileptr.write(" Python has an easy syntax and user-friendly interaction.")

    # closing the opened file
    fileptr.close()
```

Код с with:

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

if __name__ == '__main__':
    # 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.")
```



1.py × 1with.py × file2.txt × 2.py × 2with.py ×  
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.

Рисунок 2 – Результат работы программы

### Пример 3. Чтение строк с помощью метода readline().

```
#!/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)

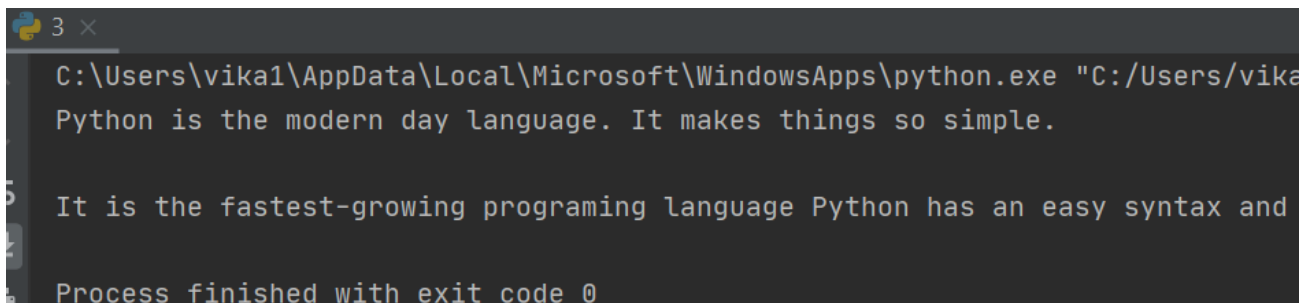
    # closes the opened file
    fileptr.close()
```

#### Код с with:

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

if __name__ == '__main__':
    # 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)
```



```
3 x
C:\Users\vika1\AppData\Local\Microsoft\WindowsApps\python.exe "C:/Users/vika1/Python is the modern day language. It makes things so simple.
Python is the modern day language. It makes things so simple.

It is the fastest-growing programming language Python has an easy syntax and

Process finished with exit code 0
```

Рисунок 3 – Результат работы программы

### Пример 4. Чтение строк с помощью функции readlines().

```
#!/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()
```

Код с with:

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

if __name__ == '__main__':
    # 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)
```

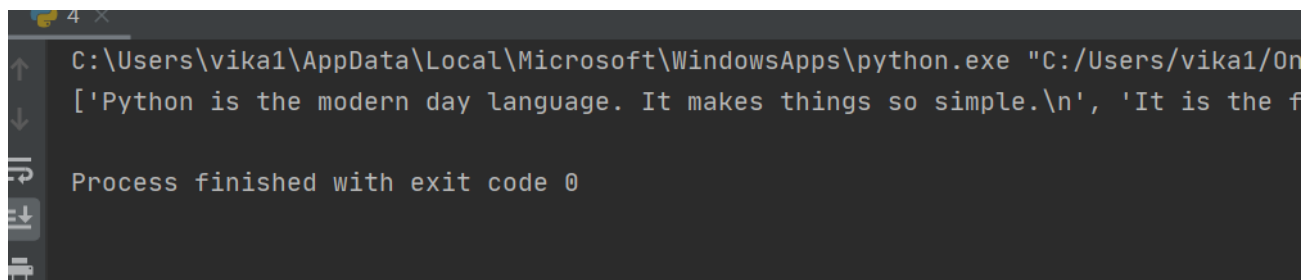


Рисунок 4 – Результат работы программы

Пример 5. Создание нового файла.

```
#!/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")

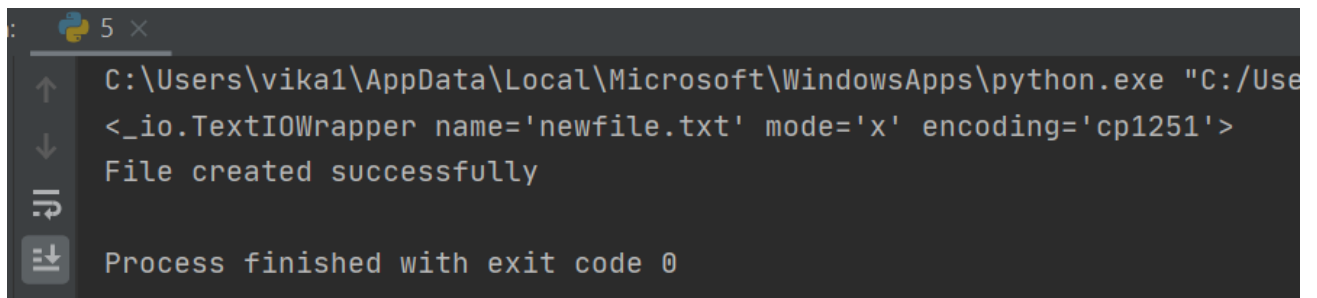
    # closes the opened file
    fileptr.close()
```

Код с with:

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

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

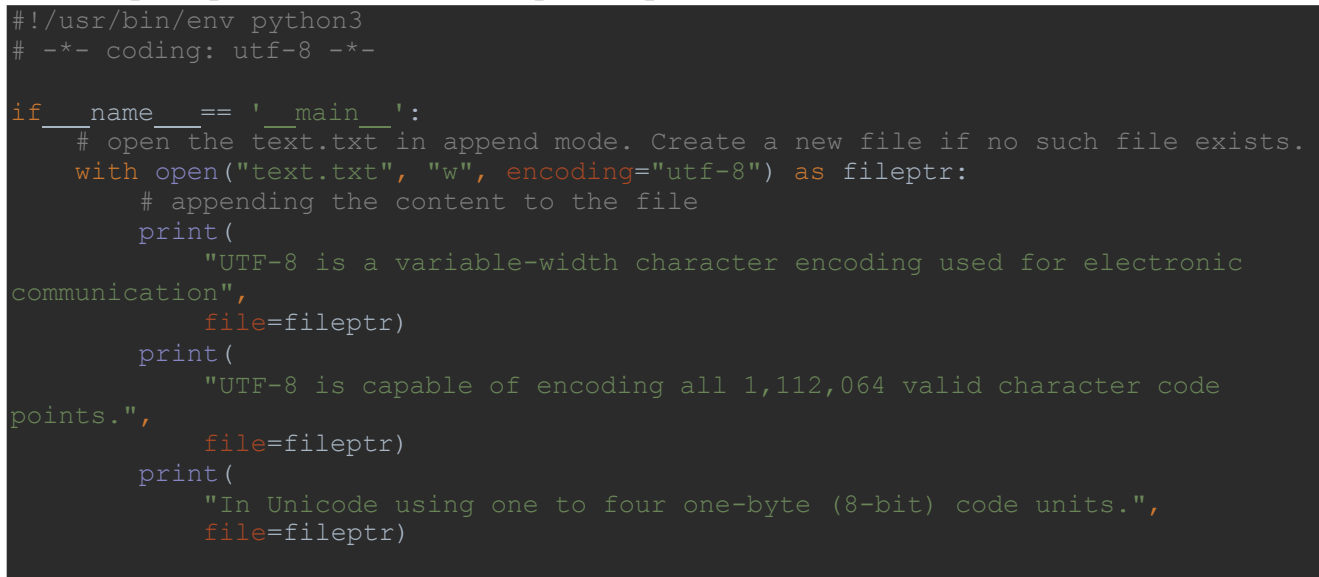
        if fileptr:
            print("File created successfully")
```



```
C:\Users\vika1\AppData\Local\Microsoft\WindowsApps\python.exe "C:/Use
<_io.TextIOWrapper name='newfile.txt' mode='x' encoding='cp1251'>
File created successfully
Process finished with exit code 0
```

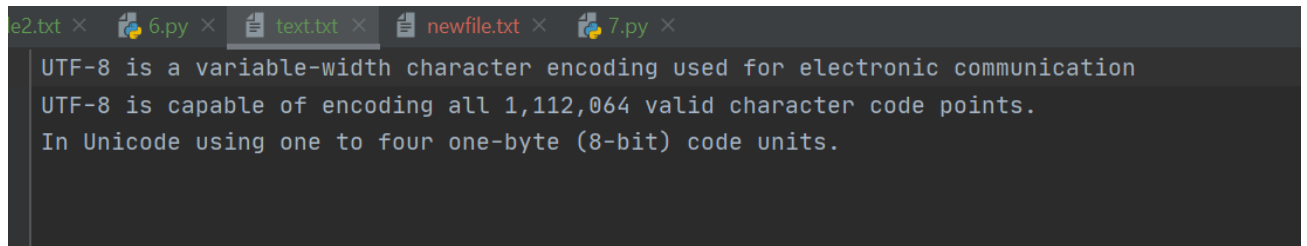
Рисунок 5 – Результат работы программы

### Пример 6. Изменение кодировки файла.



```
#!/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)
```



```
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 6– Результат работы программы

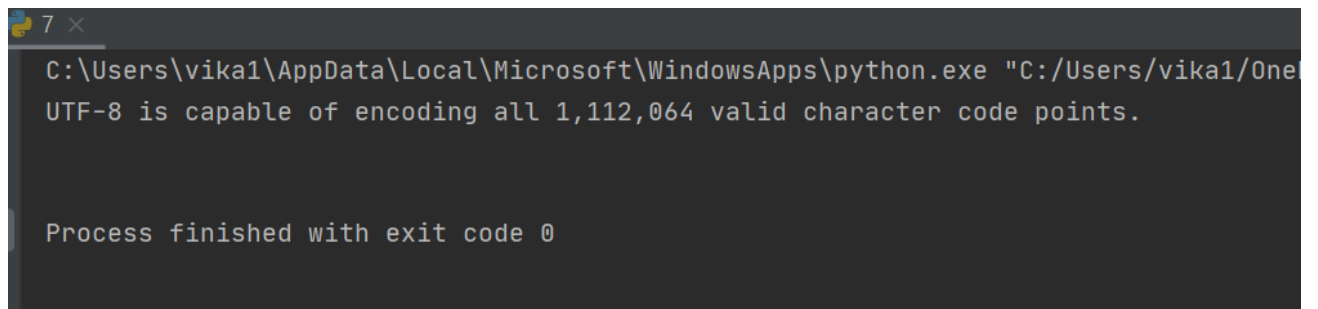
Пример 7. Написать программу, которая считывает текст из файла и выводит на экран только предложения, содержащие запятые. Каждое предложение в файле записано на отдельной строке.

Код:

```
#!/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)
```



```
7 x
C:\Users\vika1\AppData\Local\Microsoft\WindowsApps\python.exe "C:/Users/vika1/One...
UTF-8 is capable of encoding all 1,112,064 valid character code points.

Process finished with exit code 0
```

Рисунок 7 – Результат работы программы

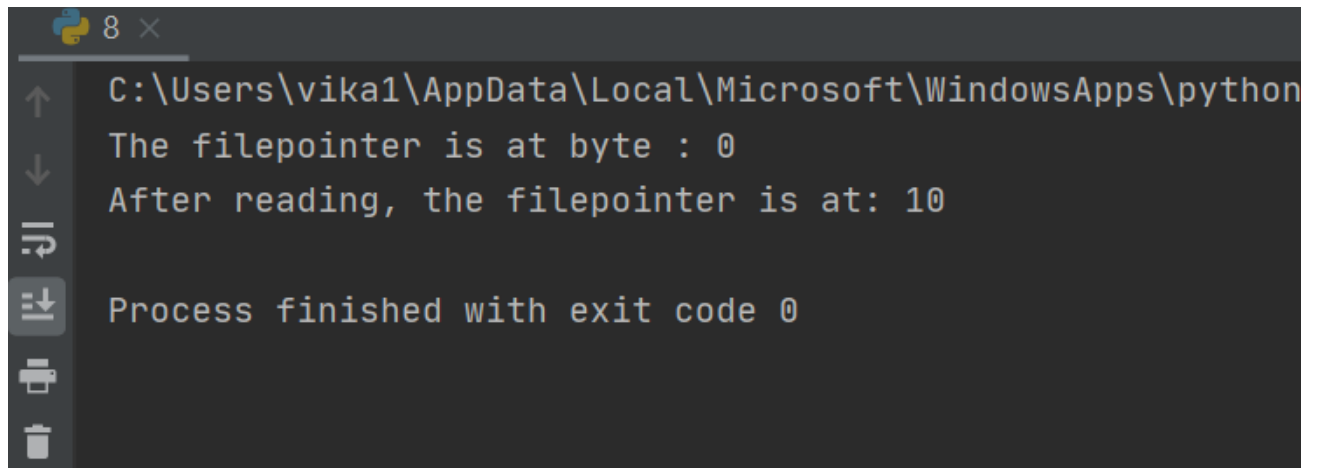
Пример 8. Позиция указателя файла.

```
#!/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())
```



A screenshot of a Python terminal window. The title bar shows a Python logo, the number '8', and a close button. The terminal output is as follows:

```
C:\Users\vika1\AppData\Local\Microsoft\WindowsApps\python
The filepointer is at byte : 0
After reading, the filepointer is at: 10

Process finished with exit code 0
```

On the left side of the terminal, there is a vertical toolbar with icons for: up arrow, down arrow, undo, redo, copy, paste, and delete.

Рисунок 8 – Результат работы программы



### Пример 9. Переименование файла.

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

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

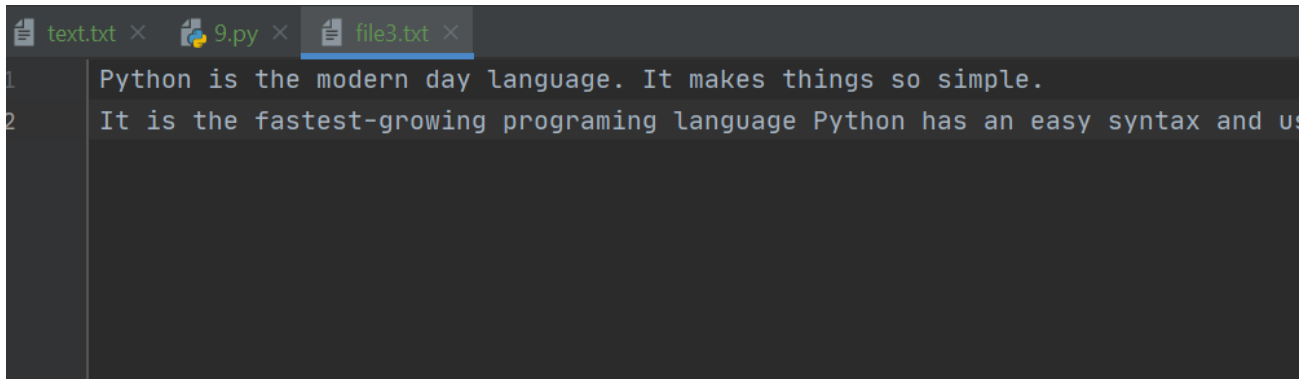


Рисунок 9 – Результат работы программы

### Пример 10. Удаление файла Код:

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

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

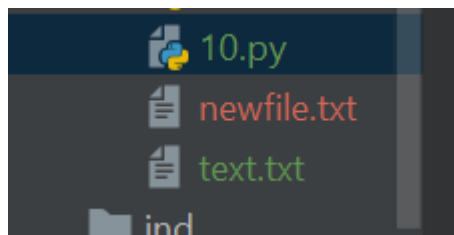


Рисунок 10 – Результат работы программы

### Пример 11. Создание нового каталога.

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

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

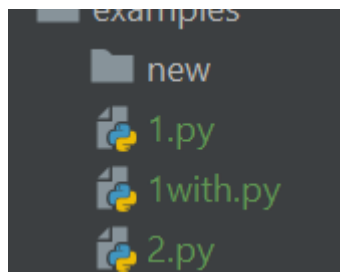


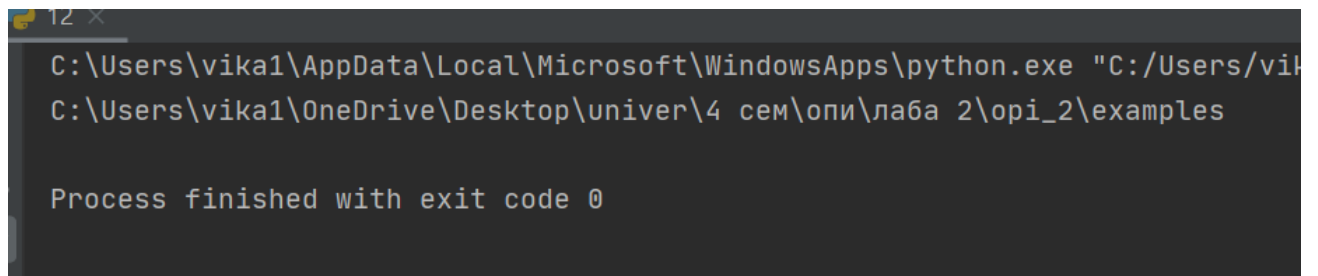
Рисунок 11 – Результат работы программы

### Пример 12. Получение текущего рабочего каталога.

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

import os

if __name__ == "__main__":
    path = os.getcwd()
    print(path)
```



```
12 x
C:\Users\vika1\AppData\Local\Microsoft\WindowsApps\python.exe "C:/Users/vik
C:\Users\vika1\OneDrive\Desktop\univer\4 сем\опи\лаба 2\опи_2\examples

Process finished with exit code 0
```

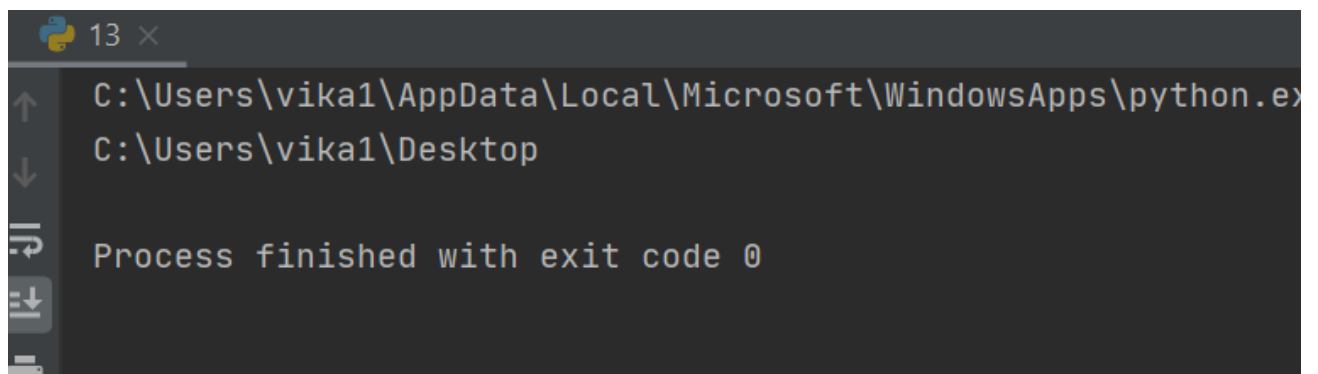
Рисунок 12 – Результат работы программы

### Пример 13. Изменение текущего рабочего каталога.

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

import os

if __name__ == "__main__":
    # Changing current directory with the new directory
    os.chdir("/Users/vika1/Desktop")
    # It will display the current working directory
    print(os.getcwd())
```



```
13 x
C:\Users\vika1\AppData\Local\Microsoft\WindowsApps\python.exe
C:\Users\vika1\Desktop

Process finished with exit code 0
```

Рисунок 13 – Результат работы программы

#### Пример 14. Удаление каталога.

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

import os

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

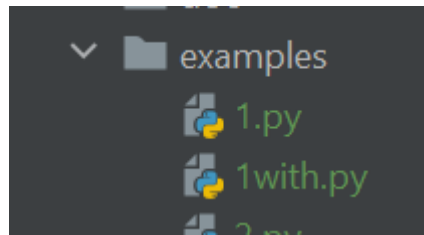


Рисунок 14 – Результат работы программы

Пример 15. Доступ к элементам командной строки в языке программирования Python.

```
#!/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))
```

```
C:\Users\vika1\OneDrive\Desktop\univer\4 сем\опи\лаба 2\опи_2\examples>python 15.py
Number of arguments: 4 arguments
Argument List: ['15.py', 'arg1', 'arg2', 'arg3']
```

Рисунок 15 – Результат работы программы

Пример 16. Изменение кодировки файла.

```
#!/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))
```

```
C:\Users\vika1\OneDrive\Desktop\univer\4 сем\опи\лаба 2\опи_2\examples>python 16.py Knowledge Hut 21
Argument #0 is 16.py
Argument #1 is Knowledge
Argument #2 is Hut
Argument #3 is 21
No. of arguments passed is 4
```

Рисунок 16 – Результат работы программы

### Пример 17. Изменение кодировки файла.

```
#!/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)}")
```

```
C:\Users\vika1\OneDrive\Desktop\univer\4 сем\опи\лаба 2\опи_2\examples>python 17.py 12
Secret Password: NvWgCoQ;/b'
```

Рисунок 17 – Результат работы программы

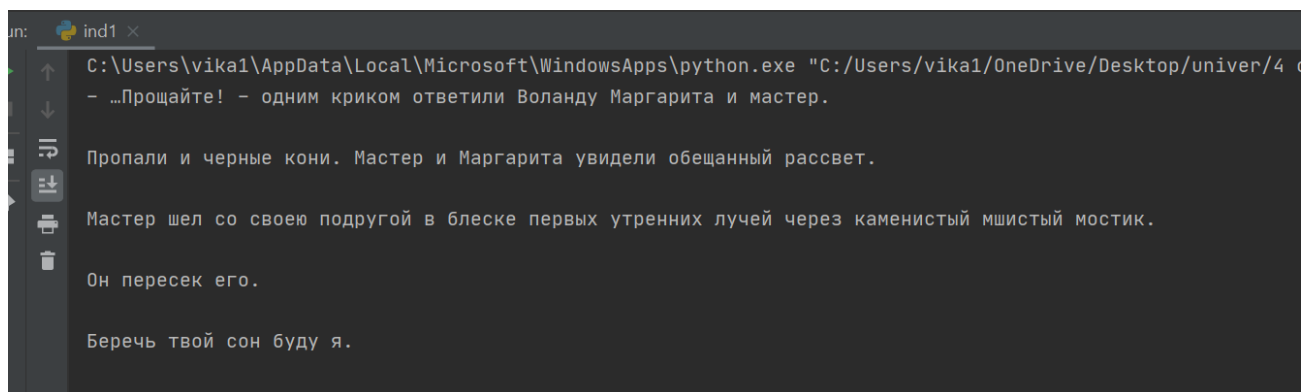
### Выполнить индивидуальные задания

Задание 1. Составить программу с использованием списков и словарей для решения задачи. Номер варианта определяется по согласованию с преподавателем. Исходный файл, из которого выполняется чтение, необходимо также добавить в репозиторий, каждое предложение в файле должно находиться на отдельной строке.

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

"""
Вариант 6
Написать программу, которая считывает текст из файла и выводит на экран только
предложения, не содержащие запятых.
"""

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 "," not in sentence:
                print(sentence)
```



```
un: ind1 x
C:\Users\vika1\AppData\Local\Microsoft\WindowsApps\python.exe "C:/Users/vika1/OneDrive/Desktop/univer/4 0
- ...Прощайте! - одним криком ответили Воланду Маргарита и мастер.

Пропали и черные кони. Мастер и Маргарита увидели обещанный рассвет.

Мастер шел со своею подругой в блеске первых утренних лучей через каменистый мшистый мостик.

Он пересек его.

Беречь твой сон буду я.
```

Рисунок 18 – Результат работы программы

Задание 2. Составить программу с использованием текстовых файлов.

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

"""
Вариант 6
Напишите программу, которая будет
способствовать дешифрации текста путем вывода на экран частоты появления разных букв.
При этом пробелы, знаки препинания и цифры должны быть проигнорированы. Также не
должен учитываться регистр, то есть символы а и А должны восприниматься как одна
буква.
Имя файла для анализа пользователь должен передавать программе посредством
аргумента командной строки. Если программе не удастся открыть файл для анализа или
аргументов командной строки будет слишком много, на экране должно быть отображено
соответствующее сообщение об ошибке.
"""
# импортируем системный модуль
import sys

# Отслеживаем, чтобы программа обязательно запускалась с одним переданным параметром
if len(sys.argv) != 2:
    print("Ошибка! Имя файла необходимо передать в качестве аргумента.")
    sys.exit(1)

# Открываем на чтение файл, имя которого было передано в командной строке
inf = open(sys.argv[1], "r", encoding="utf-8")
frequency = dict() # создаю пустой словарь
line = inf.readline()
while line != "":
    line = line.replace(".", " ")
    line = line.replace(",", " ")
    line = line.replace(":", " ")
    line = line.replace("!", " ")
    line = line.replace("-", " ")
    line = line.replace("?", " ")
    line = line.replace(" ", "")
    line = line.lower()
    line = line.rstrip()
    for letter in line:
        if letter in frequency:
            frequency[letter] += 1
        else:
            frequency[letter] = 1
    line = inf.readline()
inf.close()
for letter in frequency:
    print(letter, ":", frequency[letter])
```

```
C:\Users\vika1\OneDrive\Desktop\univer\4 сем\опи\лаба 2\опи_2\ind>python ind2.py text.txt
п : 40
р : 74
о : 136
щ : 5
а : 118
й : 38
т : 103
е : 122
л : 44
```

Рисунок 19 – Результат работы программы



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

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Программа, которая создает директорию, затем создает в этой директории файл с
некоторым текстом.
Затем переименовывает этот файл и удаляет его. После этого программа удаляет
созданную ранее директорию.
В задаче используются функций модуля os, а именно mkdir(), rename(), remove() и
rmdir().
"""
import os

if __name__ == "__main__":
    filename = "ind3.txt"
    directory = "new"
    filepath = directory + "/" + filename

    print(filepath)

    # Создаем директорию
    os.mkdir(directory)

    # Создаем файл
    with open(filepath, "w") as f:
        f.write("Hello!!!")

    # Переименовываем файл
    os.rename(filepath, directory + "/ind3_new.txt")

    # Удаляем файл
    os.remove(directory + "/ind3_new.txt")

    # Удаляем директорию
    os.rmdir(directory)
```

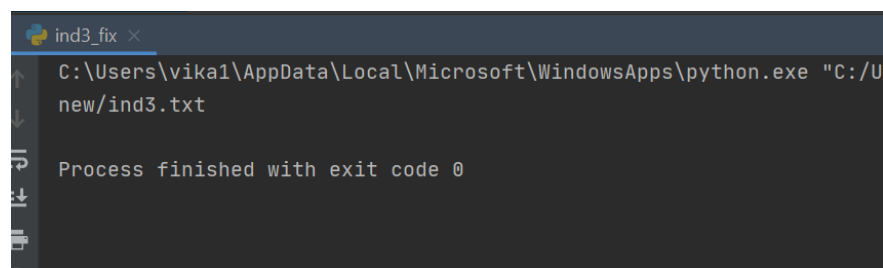


Рисунок 20 – Результат работы программы

10. Зафиксируйте сделанные изменения в репозитории.

### **Вопросы для защиты работы:**

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

С помощью команды: «fileobj = open("file.txt", "r")» или с помощью «with open("file2.txt", "w") as fileptr:»

#### **2. Как открыть файл в языке Python только для записи**

Для этого нужно указать код доступа «r» или «rb» (открывает файл в двоичном формате)

#### **3. Как прочитать данные из файла в языке Python?**

После открытия можно воспользоваться командами f.read() или f.readlines() (возвращает массив строк), f.readline() (возвращает 1 строку)

#### **4. Как записать данные в файл в языке Python?**

Во-первых, файл должен быть открыт с соответствующим режимом доступа, например, «w» (переписывает файл) или «a» (добавляет в конец).

Во-вторых, воспользоваться специальной командой, такой как f.write("///").

#### **5. Как закрыть файл в языке Python?**

Необходимо воспользоваться методом close(), однако, если файл был открыт с «with», то закрытие выполнится автоматически.

**6. Изучите самостоятельно работу конструкции with ... as. Каково ее назначение в языке Python? Где она может быть использована еще, помимо работы с файлами?**

Данная конструкция является менеджером контекста. Помимо файлов может использоваться в работе с базами данных:

```
def get_all_songs():
    with sqlite3.connect('db/songs.db') as connection:
        cursor = connection.cursor()
        cursor.execute("SELECT * FROM songs ORDER BY id desc")
        all_songs = cursor.fetchall()
    return all_songs
```

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

Есть возможность записать в файл большой объем данных, если он может быть представлен в виде списка строк.

```
with open("examp.le", "w") as f:
    f.writelines(list_of_strings)
```

Существует еще один, менее известный, способ, но, возможно, самый удобный из представленных. И как бы не было странно, он заключается в использовании функции print(). Сначала это утверждение может показаться странным, потому что общеизвестно, что с помощью нее происходит вывод в консоль. И это правда. Но если передать в необязательный аргумент file объект типа io.TextIOWrapper, каким и является объект файла, с которым мы работаем, то поток вывода функции print() перенаправляется из консоли в файл.

```
with open("examp.le", "w") as f:
    print(some_data, file=f)
```

С помощью `file.seek()` можно перемещать указатель в файле на определённое количество байтов.

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

Вернуться в предыдущую директорию: `os.chdir("..")`

Сделать несколько вложенных папок:

```
os.makedirs("nested1/nested2/nested3")
```

Заменить (переместить) этот файл в другой каталог: `os.replace("renamed-text.txt", "folder/renamed-text.txt")`

Распечатать все файлы и папки в текущем каталоге:

```
print("Все папки и файлы:", os.listdir())
```

Генератор дерева каталогов: `os.walk()`

Распечатать все файлы и папки рекурсивно: `for`

```
dirpath, dirnames, filenames in os.walk("."):
    ...
```

Получение информации о файле: `os.stat()`

Это вернет кортеж с отдельными метриками. В их числе есть следующие: `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.