

**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

Халимендик Я. Д. « » 2023г.

Подпись студента _____

Работа защищена « » _____ 20__ г.

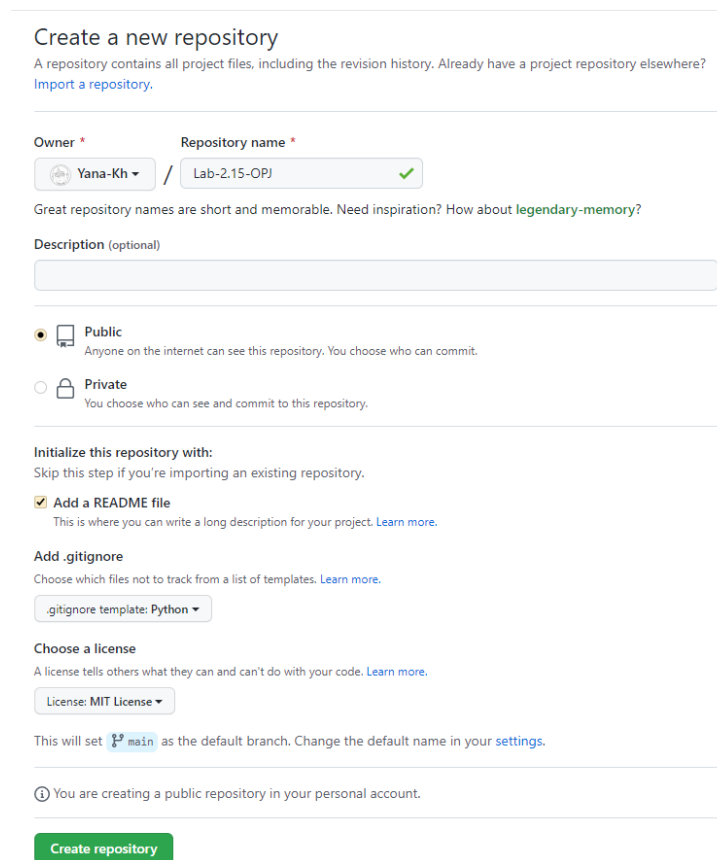
Проверил Воронкин Р.А. _____
(подпись)

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.

Ход работы:

1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия IT и язык программирования Python.



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner * / Repository name *

Great repository names are short and memorable. Need inspiration? How about [legendary-memory](#)?

Description (optional)

☒ Public
Anyone on the internet can see this repository. You choose who can commit.

☐ Private
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

☒ Add a README file
This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore
Choose which files not to track from a list of templates. [Learn more.](#)

Choose a license
A license tells others what they can and can't do with your code. [Learn more.](#)

This will set `main` as the default branch. Change the default name in your [settings](#).

You are creating a public repository in your personal account.

[Create repository](#)

Рисунок 1 – Создание репозитория

3. Выполните клонирование созданного репозитория.

```
C:\Users\anton>cd C:\Users\anton\Desktop\Git
C:\Users\anton\Desktop\Git>git clone https://github.com/Yana-Kh/Lab-2.15-OPJ.git
Cloning into 'Lab-2.15-OPJ'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.
C:\Users\anton\Desktop\Git>
```

Рисунок 2 – Клонирование репозитория

4. Дополните файл .gitignore необходимыми правилами для работы с IDE PyCharm.

```
C:\Users\антон\Desktop\Git\Lab-2.15-OPJ>git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   .gitignore

C:\Users\антон\Desktop\Git\Lab-2.15-OPJ>
```

Рисунок 3 – Дополнение файла .gitignore

5. Организуйте свой репозиторий в соответствии с моделью ветвления git-flow.

```
C:\Users\антон\Desktop\Git\Lab-2.15-OPJ>git flow init

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

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [C:/Users/антон/Desktop/Git/Lab-2.15-OPJ/.git/hooks]

C:\Users\антон\Desktop\Git\Lab-2.15-OPJ>
```

Рисунок 4 – Организация ветвления git-flow

Рисунок 4 – Организация репозитория в соответствии с моделью git-flow

6. Создайте проект PyCharm в папке репозитория.

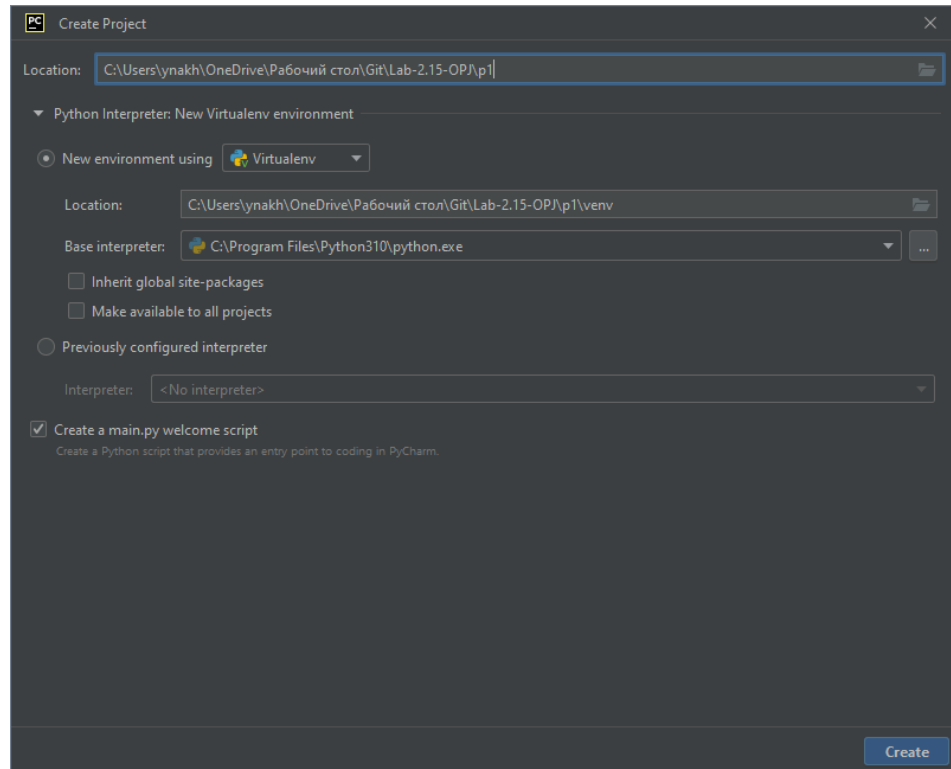


Рисунок 5 – Создание проекта PyCharm

7. Проработать примеры лабораторной работы.

Пример 1. Запись файла

Код:

```
#!/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 programming language"
    )
    # closing the opened the file
    fileptr.close()
```

Код с with:

```
#!/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.
```

```

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"
    )

```

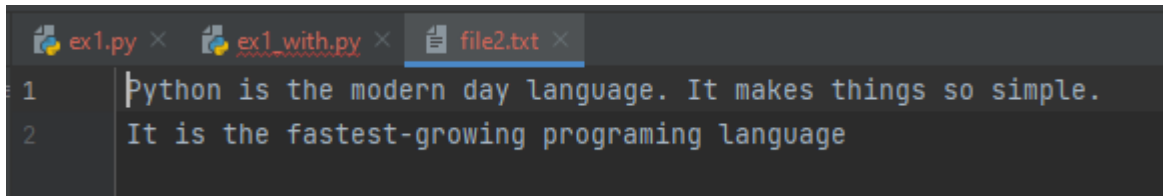


Рисунок 6 – Результат работы программы

Пример 2. Запись файла

Код:

```

#!/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()

```

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.")

```

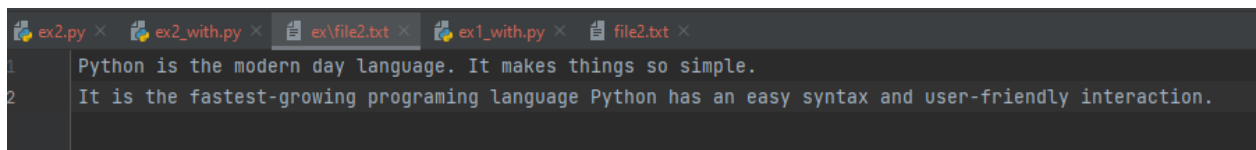


Рисунок 7 – Результат работы программы

Пример 3. Чтение строк с помощью метода readline()

Код:

```
#!/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)
```

```
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 8 – Результат работы программы

Пример 4. Чтение строк с помощью функции readlines()

Код:

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

if __name__ == '__main__':
    # open the fil2.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)

```

```

"C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-2.15-0PJ\py\venv\Scripts\python.exe" "C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-2.15-0PJ\py\ex\ex4.py"
['Python is the modern day language. It makes things so simple.\n', 'It is the fastest-growing programing language Python has an easy syntax and user-friendly interaction.']

```

Рисунок 9 – Результат работы программы

Пример 5. Создание нового файла

Код:

```

#!/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")

```

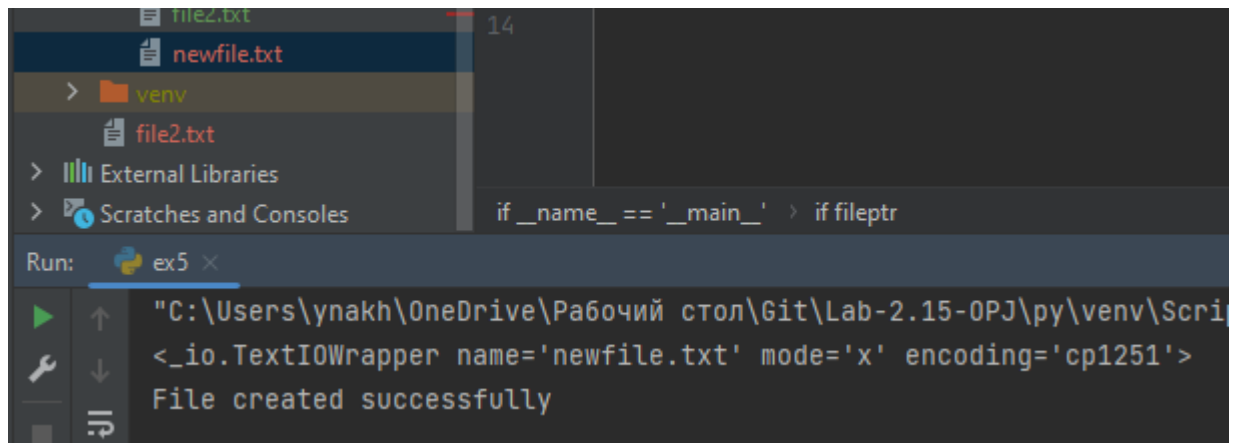


Рисунок 10 – Результат работы программы

Пример 6. Изменение кодировки файла

Код:

```
#!/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
        )
```

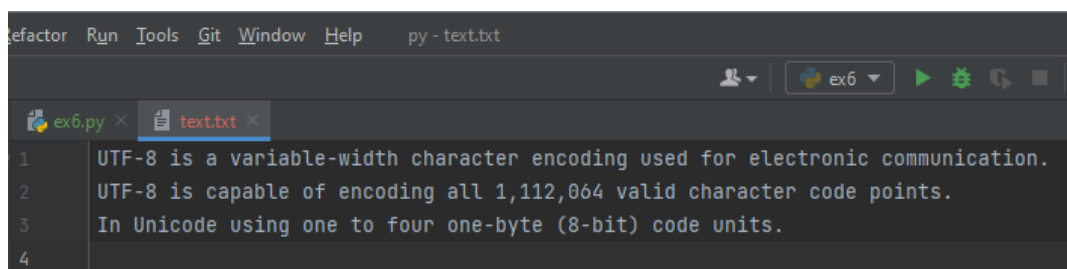


Рисунок 11 – Результат работы программы

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 f:
        sentences = f.readlines()

    # Вывод предложений с запятыми.
    for sentence in sentences:
        if "," in sentence:
            print(sentence)
```

```
UTF-8 is capable of encoding all 1,112,064 valid character code points.
```

Рисунок 12 – Результат работы программы

Пример 8. Позиция указателя файла

Код:

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

```
The filepointer is at byte : 0
After reading, the filepointer is at: 10
```

Рисунок 13 – Результат работы программы

Пример 9. Переименование файла

Код:

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

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

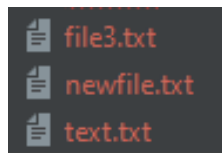


Рисунок 14 – Результат работы программы

Пример 10. Удаление файла

Код:

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

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

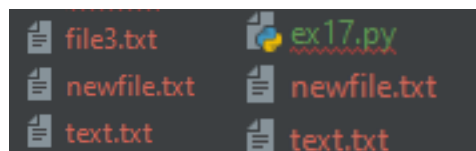


Рисунок 15 – Результат работы программы

Пример 11. Создание нового каталога

Код:

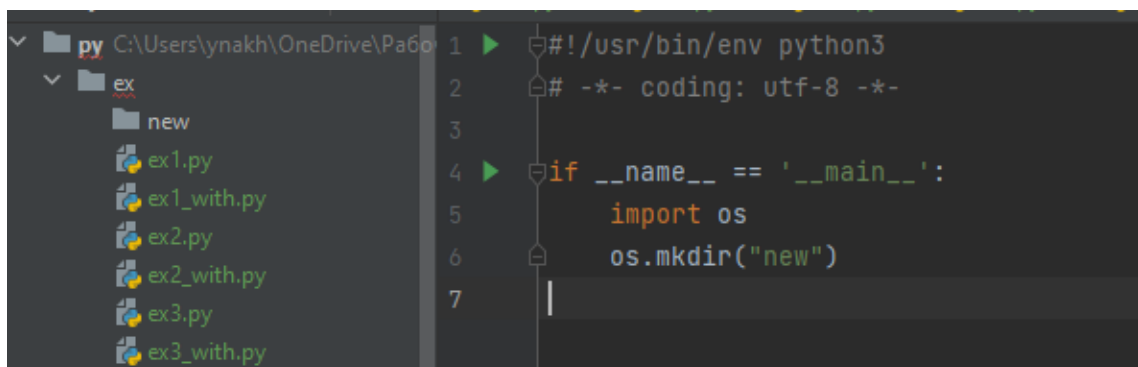


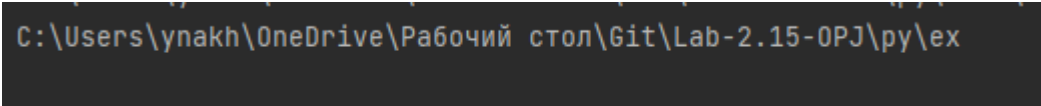
Рисунок 16 – Результат работы программы

Пример 12. Получение текущего рабочего каталога.

Код:

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

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



```
C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-2.15-OPJ\py\ex
```

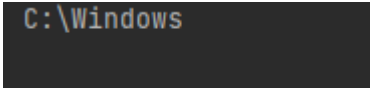
Рисунок 17 – Результат работы программы

Пример 13. Изменение текущего рабочего каталога

Код:

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

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



```
C:\Windows
```

Рисунок 18 – Результат работы программы

Пример 14. Удаление каталога

Код:

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

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

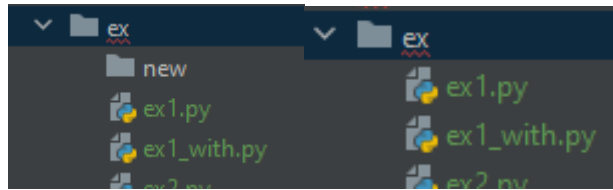


Рисунок 19 – Результат работы программы

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

Код:

```
#!/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\ynakh\OneDrive\Рабочий стол\Git\Lab-2.15-OPJ\py\ex>python ex15.py arg1 arg2 arg3
Number of arguments: 4 arguments
Argument List: ['ex15.py', 'arg1', 'arg2', 'arg3']
```

Рисунок 20 – Результат работы программы

Пример 16. Изменение кодировки файла

Код:

```
#!/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\ynakh\OneDrive\Рабочий стол\Git\Lab-2.15-OPJ\py\ex>python ex16.py Knowledge Hut 21
Argument #0 is ex16.py
Argument #1 is Knowledge
Argument #2 is Hut
Argument #3 is 21
No. of arguments passed is 4
C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-2.15-OPJ\py\ex>
```

Рисунок 21 – Результат работы программы

Пример 17. Изменение кодировки файла

Код:

```
#!/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\ynakh\OneDrive\Рабочий стол\Git\Lab-2.15-OPJ\py\ex> ex17 12
Secret Password: C_g[/ntXErB;
```

Рисунок 22 – Результат работы программы

8. 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.

Вариант 29(10). Написать программу, которая считывает текст из файла и выводит на экран только строки, не содержащие двузначных чисел.

Код:

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

if __name__ == '__main__':
```

```

with open("text1.txt", "r", encoding="utf-8") as f:
    sentences = f.readlines()

print("Предложения с 2-значными числами:")
# Вывод предложений с двухзначными числами.
se
    for ind, i in enumerate(sentence):
        if (i.isdigit() and ind != len(sentence) and
            sentence[ind + 1].isdigit()):
            print(sentence)

```

```

Предложения с 2-значными числами:
There are 32 people in the class in total.

There were more than 20 books in his collection.

```

Рисунок 23 – Результат работы программы

Задание 2

Составить программу с использованием текстовых файлов. Номер варианта необходимо получить у преподавателя.

Вариант 29(14) Перед публикацией текста или документа обычно принято удалять или изменять в них служебную информацию. В данном упражнении вам необходимо написать программу, которая будет заменять все служебные слова в тексте на символы звездочек (по количеству символов в словах). Вы должны осуществлять регистрозависимый поиск служебных слов в тексте, даже если эти слова входят в состав других слов. Список служебных слов должен храниться в отдельном файле. Сохраните отредактированную версию исходного файла в новом файле. Имена исходного файла, файла со служебными словами и нового файла должны быть введены пользователем.

Код:

```

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

if __name__ == '__main__':
    file_name = input("Введите имя редактируемого файла: ")
    words_name = input("Введите имя файла со словами: ")
    words = []

    # Чтение файла со служебными словами
    with open(words_name, "r", encoding="utf-8") as f:

```

```

        words = f.readlines()

# Открытие файла для записи
with open("new2.txt", "w", encoding="utf-8") as new_f:

    # Чтение заданного файла
    with open(file_name, "r+", encoding="utf-8") as f:
        sentences = f.readlines()
        for sentence in sentences:
            for i in words:
                if i[:-1] in sentence:
                    sentence = sentence.replace(i[:-1], "*" * (len(i) -
1))

    # Запись в файл новой версии
    new_f.write(sentence)

print("Операция прошла успешно")

```

```

Введите имя редактируемого файла: text2.txt
Введите имя файла со словами: words.txt
Операция прошла успешно

```

Рисунок 24 – Результат работы программы

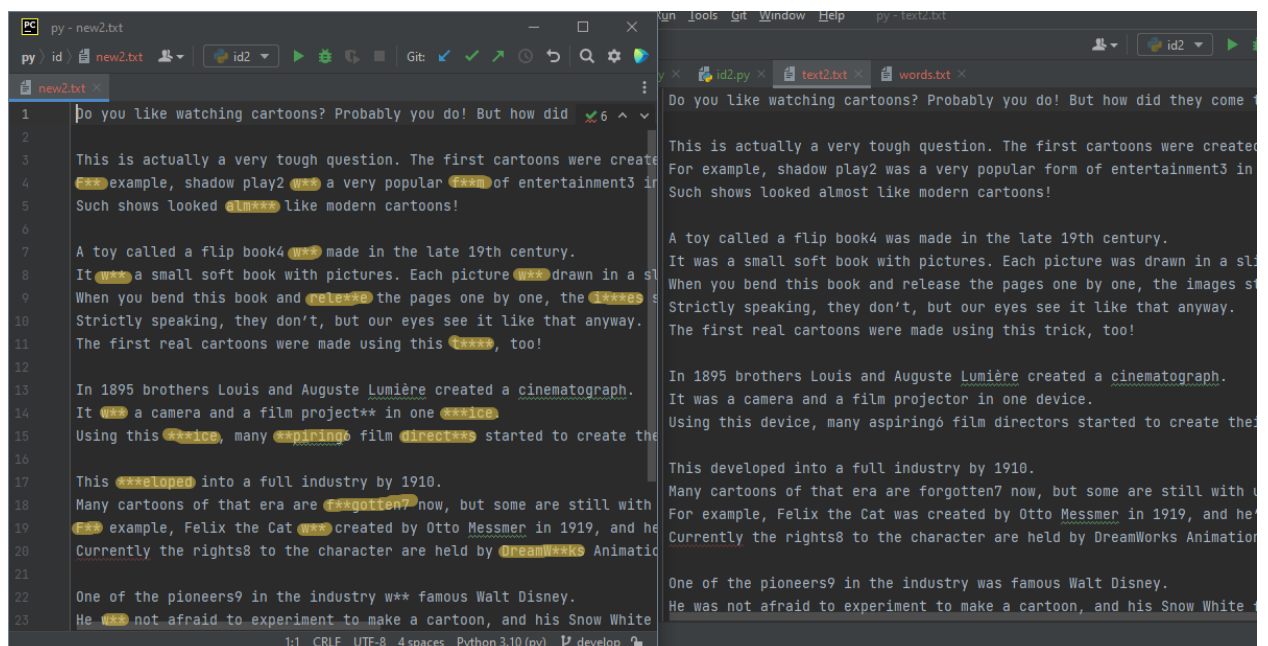


Рисунок 25 – Новый файл, созданный после обработки.

9. Зафиксируйте изменения в репозитории.



Рисунок 26 – Фиксирование изменений в репозитории

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

Задание: Создать каталог, поместить в него текстовый документ с именем этого каталога и переименование файла в этом каталоге.

Код:

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

import os

if __name__ == '__main__':
    # Создание директории и запоминание пути
    dir_name = input("Введите имя директории: ")
    os.mkdir(dir_name)
    path = os.getcwd()

    #Перемещение в новую директорию
    os.chdir(path + '\\' + dir_name)
    with open("file.txt", "w", encoding="utf-8") as f:
        # appending the content to the file
        f.write(os.getcwd())

    #Переименование файла
    new_name = input("Введите новое имя для файла: ")
    os.rename("file.txt", new_name)
```

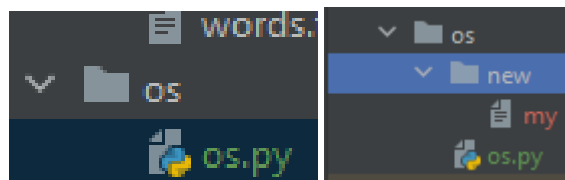


Рисунок 27 – Результат работы программы

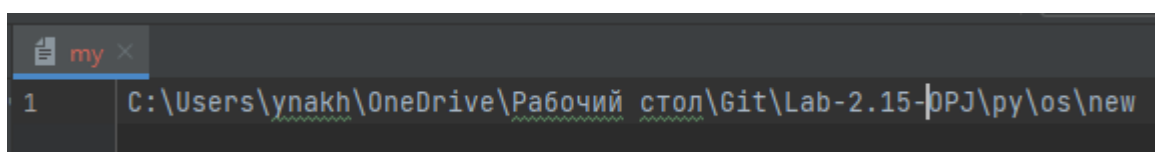


Рисунок 28 – Результат работы программы

10. Зафиксируйте сделанные изменения в репозитории.

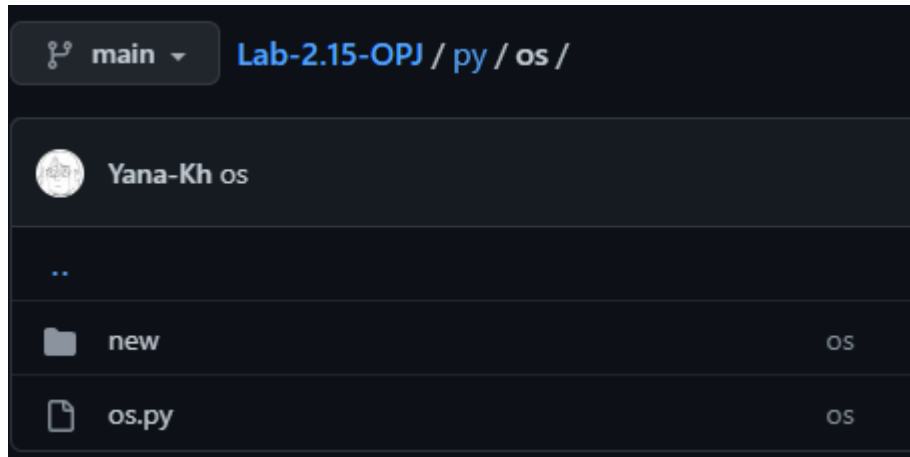


Рисунок 12 – Фиксирование изменений в репозитории

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.