

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 №2.15
дисциплины «Основы программной инженерии»

Выполнил:
Соколов Михаил Романович
2 курс, группа ПИЖ-б-о-22-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка
и сопровождение программного
обеспечения», очная форма обучения

(подпись)

Руководитель практики:
Богданов С.С., ассистент кафедры
инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2024 г.

Тема: Работа с файлами в языке Python.

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 командой строки.

Ход выполнения работы:

1. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и `.gitignore` файл для языка 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.](#)

Required fields are marked with an asterisk ().*

Repository template

No template ▾

Start your repository with a template repository's contents.

Owner * PoTtaTto ▾ / **Repository name *** fundamentals_of_software

✔ fundamentals_of_software_engineering_lab2_15 is available.

Great repository names are short and memorable. Need inspiration? How about **didactic-octo-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:

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

Add .gitignore

.gitignore template: Python ▾

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

Choose a license

License: MIT License ▾

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

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 с заданными настройками

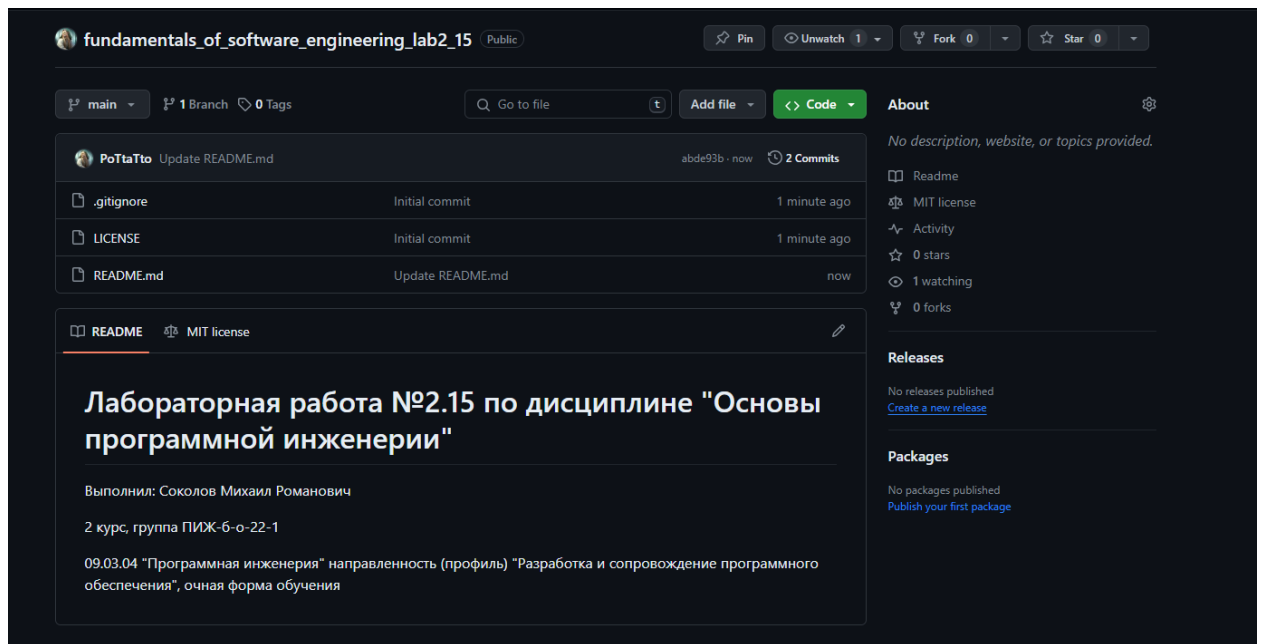


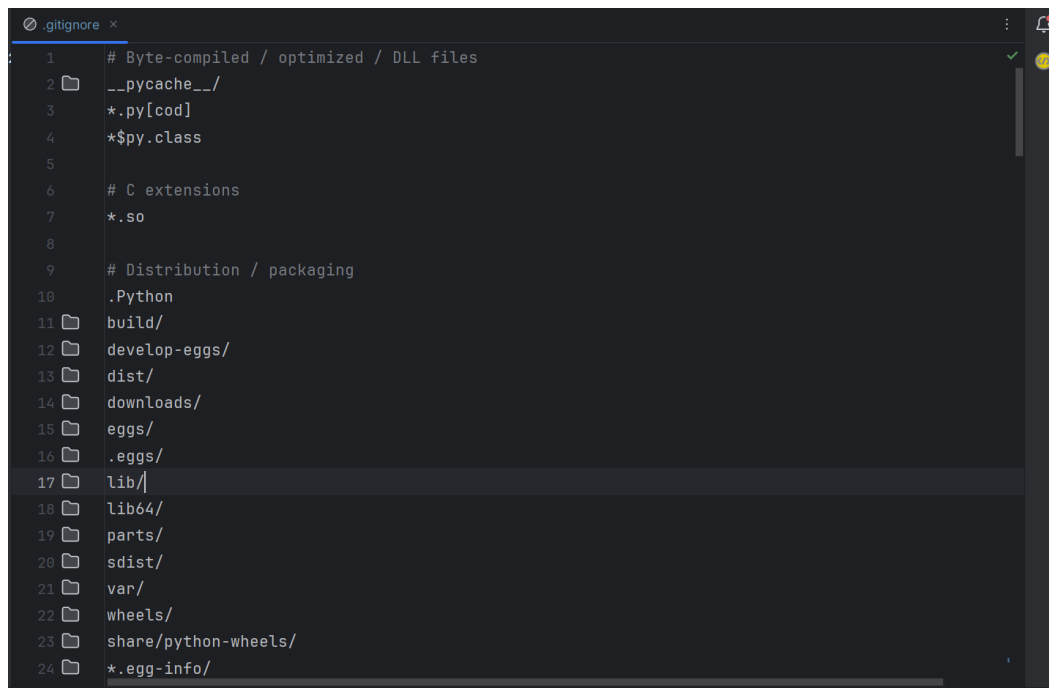
Рисунок 2 – Созданный репозиторий

```
PS C:\Study\Основы программной инженерии\2.15> git clone https://github.com/PoTtaTto/fundamentals_of_software_engineering_lab2_15
Cloning into 'fundamentals_of_software_engineering_lab2_15'...
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 8 (delta 1), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (8/8), 4.14 KiB | 4.14 MiB/s, done.
Resolving deltas: 100% (1/1), done.
PS C:\Study\Основы программной инженерии\2.15> |
```

Рисунок 3 – Клонирование репозитория

```
PS C:\Study\Основы программной инженерии\2.15\fundamentals_of_software_engineering_lab2_15> git checkout -b develop
Switched to a new branch 'develop'
PS C:\Study\Основы программной инженерии\2.15\fundamentals_of_software_engineering_lab2_15> |
```

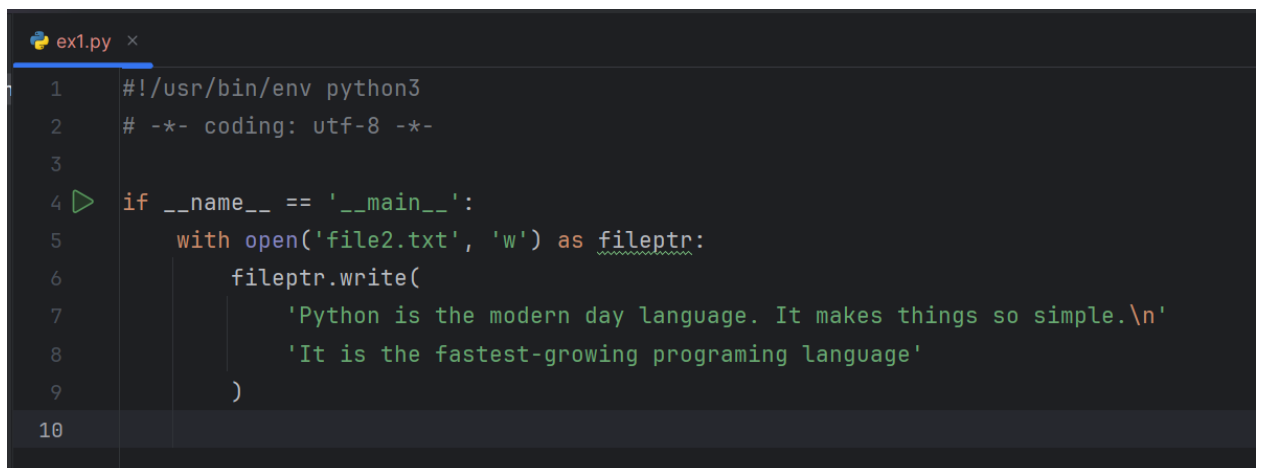
Рисунок 4 – Создание ветки develop, где будут происходить изменения проекта до его полного релиза



```
1 # Byte-compiled / optimized / DLL files
2 __pycache__/
3 *.py[cod]
4 *$py.class
5
6 # C extensions
7 *.so
8
9 # Distribution / packaging
10 .Python
11 build/
12 develop-eggs/
13 dist/
14 downloads/
15 eggs/
16 .eggs/
17 lib/
18 lib64/
19 parts/
20 sdist/
21 var/
22 wheels/
23 share/python-wheels/
24 *.egg-info/
```

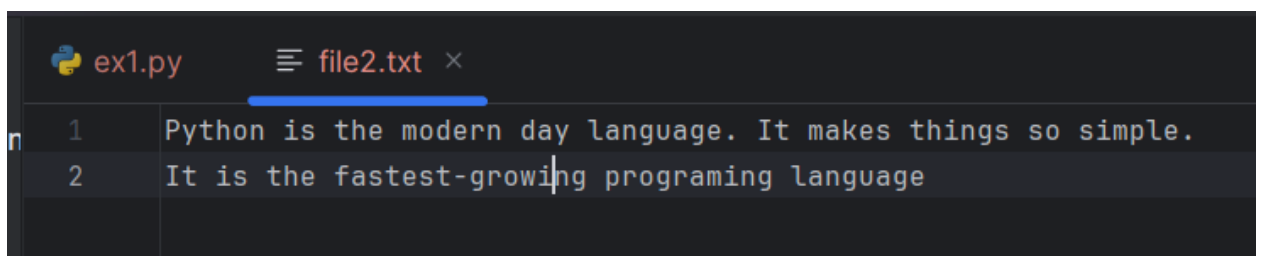
Рисунок 5 – Часть .gitignore, созданного GitHub

2. Проработаем примеры лабораторной работы:



```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 if __name__ == '__main__':
5     with open('file2.txt', 'w') as fileptr:
6         fileptr.write(
7             'Python is the modern day language. It makes things so simple.\n'
8             'It is the fastest-growing programing language'
9         )
10
```

Рисунок 6 – Запись текста в файл file2.txt



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

Рисунок 7 – Содержимое файла file2.txt после запуска программы

```
ex2.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  if __name__ == '__main__':
5      with open('file2.txt', 'a') as fileptr:
6          fileptr.write(' Python has an easy syntax and uesr-friendly interaction.')
7
```

Рисунок 8 – Добавление содержимого в файл file2.txt

```
ex2.py  file2.txt x
1  Python is the modern day language. It makes things so simple.
2  It is the fastest-growing programing language Python has an easy syntax and uesr-friendly interaction.
```

Рисунок 9 – Измененное содержимое файла file2.txt


```
ex3.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  if __name__ == '__main__':
5      with open('file2.txt', 'r') as fileptr:
6          content1 = fileptr.readline()
7          content2 = fileptr.readline()
8
9          print(content1, end='')
10         print(content2)
11
```

Рисунок 10 – Чтение первых двух строк файла file2.txt

```
C:\Users\MrPot\AppData\Local\Programs\Python\Python312\python.exe "C:\Study\Основы программной инженерии
Python is the modern day language. It makes things so simple.
It is the fastest-growing programing language Python has an easy syntax and uesr-friendly interaction.

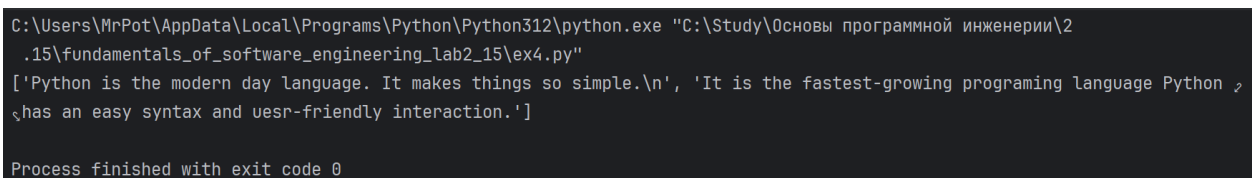
Process finished with exit code 0
```

Рисунок 11 – Вывод содержимого первых двух строк file2.txt



```
ex4.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  if __name__ == '__main__':
5      with open('file2.txt', 'r') as fileptr:
6          content = fileptr.readlines()
7          print(content)
```

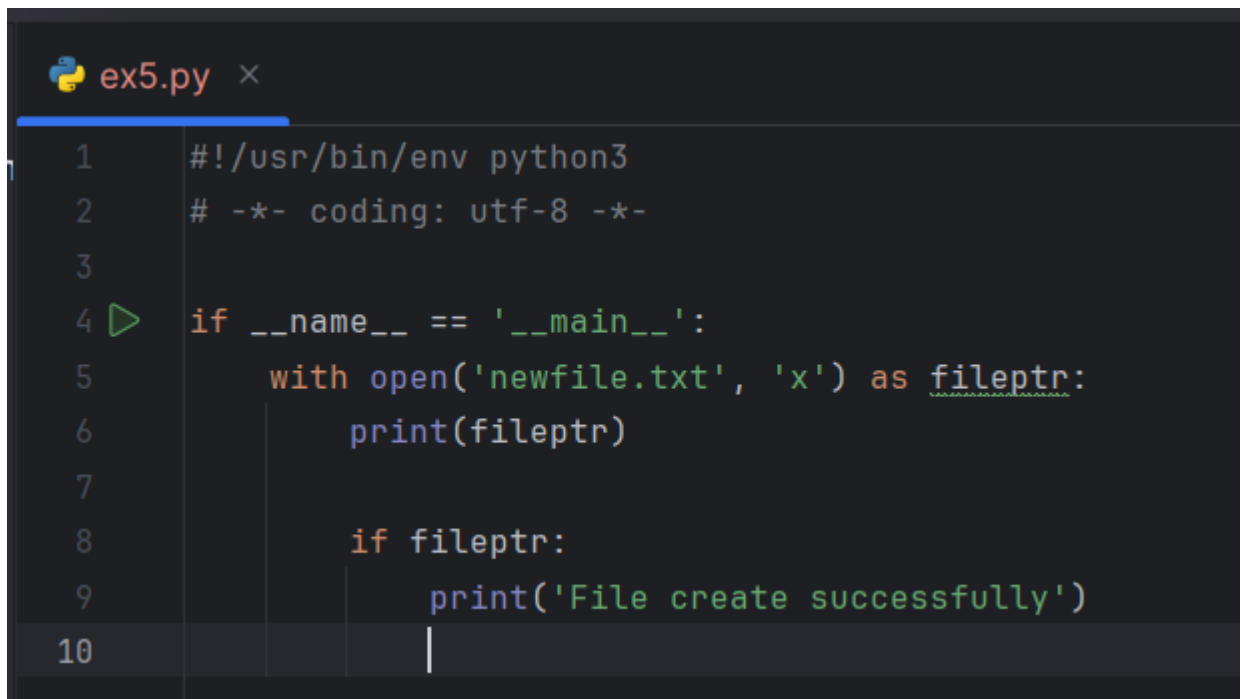
Рисунок 12 – Чтение всех строк file2.txt с помощью метода readlines()



```
C:\Users\MrPot\AppData\Local\Programs\Python\Python312\python.exe "C:\Study\Основы программной инженерии\2
.15\fundamentals_of_software_engineering_lab2_15\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 uesr-friendly interaction.']

Process finished with exit code 0
```

Рисунок 13 – Вывод списка строк file2.txt



```
ex5.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  if __name__ == '__main__':
5      with open('newfile.txt', 'x') as fileptr:
6          print(fileptr)
7
8      if fileptr:
9          print('File create successfully')
10
```

Рисунок 14 – Создание нового файла с помощью режима доступа «x»

```
C:\Users\MrPot\AppData\Local\Programs\Python\Python312\python.exe "  
<_io.TextIOWrapper name='newfile.txt' mode='x' encoding='cp1252'>  
File create successfully  
  
Process finished with exit code 0
```

Рисунок 15 – Вывод информации в консоль

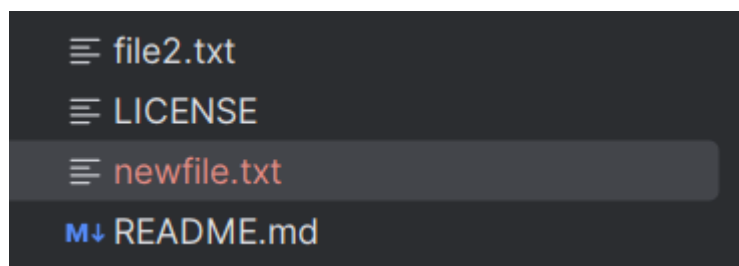


Рисунок 16 – Созданный файл в каталоге проекта

```
ex6.py ×  
1  #!/usr/bin/env python3  
2  # -*- coding: utf-8 -*-  
3  
4  if __name__ == '__main__':  
5      with open('text.txt', 'w', encoding='utf-8') as fileptr:  
6          print('UTF-8 is a variable-width character encoding used for electronic communication', file=fileptr)  
7          print('UTF-8 is capable of encoding all 1,112,064 valid character code points.', file=fileptr)  
8          print('In Unicode using one to four one-byte (8-bit) code units.', file=fileptr)  
9
```

Рисунок 17 – Создание и запись содержимого в файл text.txt с использованием кодировки UTF-8

```
ex6.py  text.txt ×  
1  UTF-8 is a variable-width character encoding used for electronic communication  
2  UTF-8 is capable of encoding all 1,112,064 valid character code points.  
3  In Unicode using one to four one-byte (8-bit) code units.  
4
```

Рисунок 18 – Содержимое файла text.txt

```
ex7.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  if __name__ == '__main__':
5      with open('text.txt', 'r', encoding='utf-8') as fileptr:
6          sentences = fileptr.readlines()
7
8          for sentence in sentences:
9              if ',' in sentence:
10                 print(sentence)
11
```

Рисунок 19 – Вывод тех предложений text.txt, в которых содержится запятая

```
C:\Users\MrPot\AppData\Local\Programs\Python\Python312\python.exe "C:\Stu
UTF-8 is capable of encoding all 1,112,064 valid character code points.

Process finished with exit code 0
```

Рисунок 20 – Предложение содержащие запятую

```
ex8.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  if __name__ == '__main__':
5      with open('file2.txt', 'r') as fileptr:
6          print('The filepointer is at byte: ', fileptr.tell())
7
8          fileptr.seek(10)
9
10         print('After reading, the filepointer is at: ', fileptr.tell())
11
```

Рисунок 21 – Искусственное смещение указателя файла на 10 байт

```
C:\Users\MrPot\AppData\Local\Programs\Python\
The filepointer is at byte: 0
After reading, the filepointer is at: 10

Process finished with exit code 0
```

Рисунок 22 – Вывод позиции указателя файла после запуска программы


```
ex9.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import os
5
6  if __name__ == '__main__':
7      os.rename(src: 'file2.txt', dst: 'file3.txt')
8
```

Рисунок 23 – Переименование файла file2.txt в file3.txt

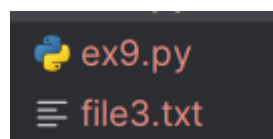


Рисунок 24 – Результат переименования file2.txt

```
ex10.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import os
5
6  if __name__ == '__main__':
7      os.remove('file3.txt')
8
```

Рисунок 25 – Удаление файла file3.txt

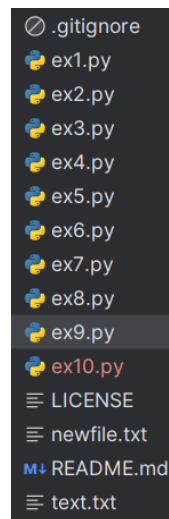


Рисунок 26 – Каталог проекта после удаление файла file3.txt

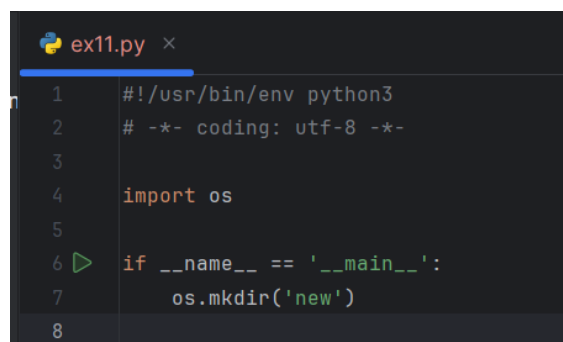


Рисунок 27 – Создание нового каталога «new»

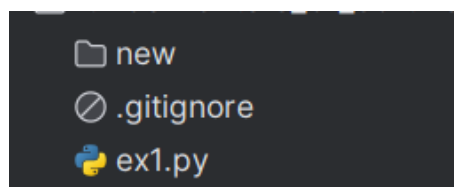


Рисунок 28 – Созданный каталог

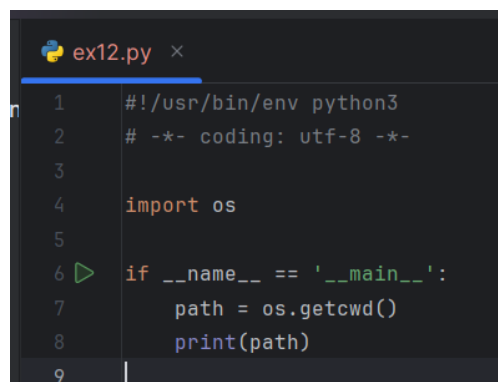
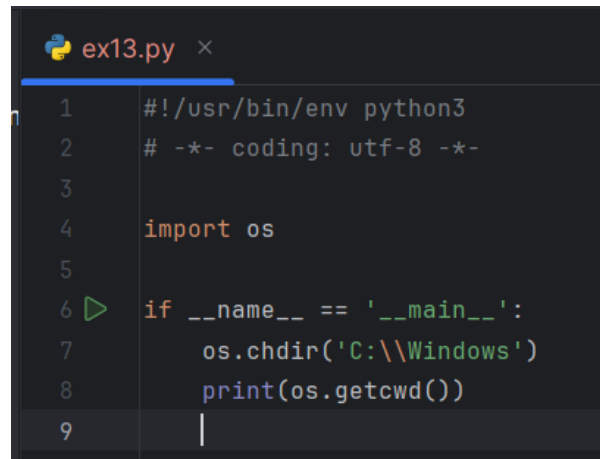


Рисунок 29 – Вывод текущего каталога

```
C:\Users\MrPot\AppData\Local\Programs\Python\Python312\python.exe "C:\Study\Основы програм
C:\Study\Основы программной инженерии\2.15\fundamentals_of_software_engineering_lab2_15

Process finished with exit code 0
```

Рисунок 30 – Вывод каталога проекта



```
ex13.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import os
5
6  if __name__ == '__main__':
7      os.chdir('C:\\Windows')
8      print(os.getcwd())
9  |
```

Рисунок 31 – Изменение текущего рабочего каталога

```
C:\Users\MrPot\AppData\Local\Programs\Python\Python312\python.exe "C:\Study\Основы програм
C:\Study\Основы программной инженерии\2.15\fundamentals_of_software_engineering_lab2_15

C:\Windows

Process finished with exit code 0
```

Рисунок 32 – Вывод нового рабочего каталога



```
ex14.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import os
5
6  if __name__ == '__main__':
7      os.rmdir('new')
8  |
```

Рисунок 33 – Удаление каталога «new»

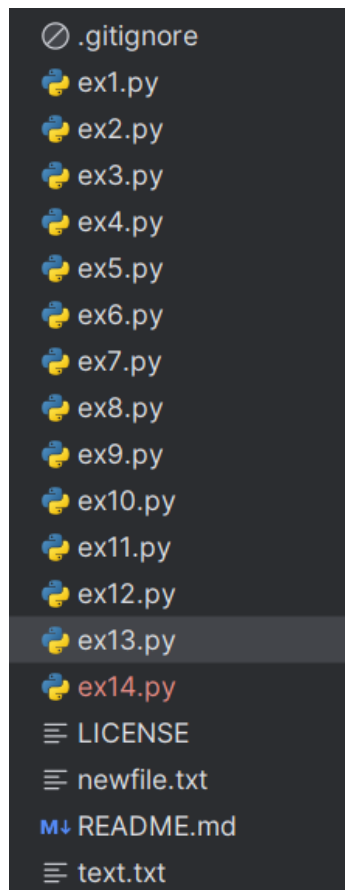


Рисунок 34 – Каталог проекта после удаления подкаталога «new»



Рисунок 35 – Вывод списка аргументов и его длины

```
PS C:\Study\Основы программной инженерии\2.15\fundamentals_of_software_engineering_lab2_15> py ex15.py arg1 arg2 arg3
Number of arguments: 4 arguments
Argument List: ['ex15.py', 'arg1', 'arg2', 'arg3']
PS C:\Study\Основы программной инженерии\2.15\fundamentals_of_software_engineering_lab2_15>
```

Рисунок 36 – Запуск программы с аргументами и их вывод

```
ex16.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6  if __name__ == '__main__':
7      for idx, arg in enumerate(sys.argv):
8          print(f'Argument #{idx} is {arg}')
9          print('No. of arguments passed is ', len(sys.argv))
10
```

Рисунок 37 – Вывод аргументов и их позиций в списке

```
PS C:\Study\Основы программной инженерии\2.15\fundamentals_of_software_engineering_lab2_15> py 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
PS C:\Study\Основы программной инженерии\2.15\fundamentals_of_software_engineering_lab2_15>
```

Рисунок 38 – Вывод аргументов с позицией и значением

```
ex17.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import secrets
5  import string
6  import sys
7
8
9  if __name__ == '__main__':
10     if len(sys.argv) != 2:
11         print('The password length is not given!', file=sys.stderr)
12         sys.exit(1)
13
14     chars = string.ascii_letters + string.punctuation + string.digits
15     length_pwd = int(sys.argv[1])
16
17     result = []
18     for _ in range(length_pwd):
19         idx = secrets.SystemRandom().randrange(len(chars))
20         result.append(chars[idx])
21
22     print(f'Secret password: {"".join(result)}')
23
```

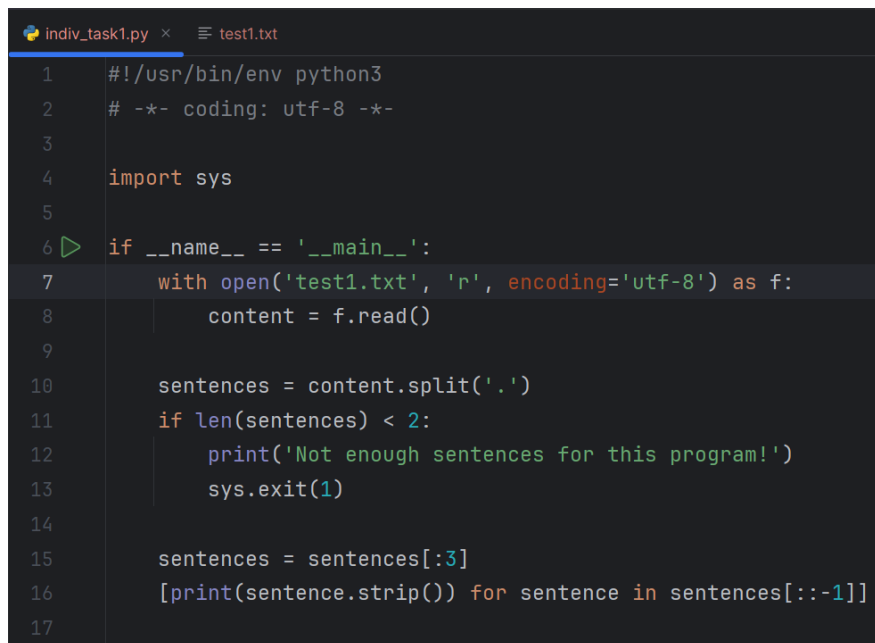
Рисунок 39 – Генерация пароля заданной длины

```
PS C:\Study\Основы программной инженерии\2.15\fundamentals_of_software_engineering_lab2_15> py ex17.py 25
Secret password: {*;Kmh[YW6:M{mJ2D6o9_R?2A
PS C:\Study\Основы программной инженерии\2.15\fundamentals_of_software_engineering_lab2_15>
```

Рисунок 40 – Сгенерированный пароль из 25 символов

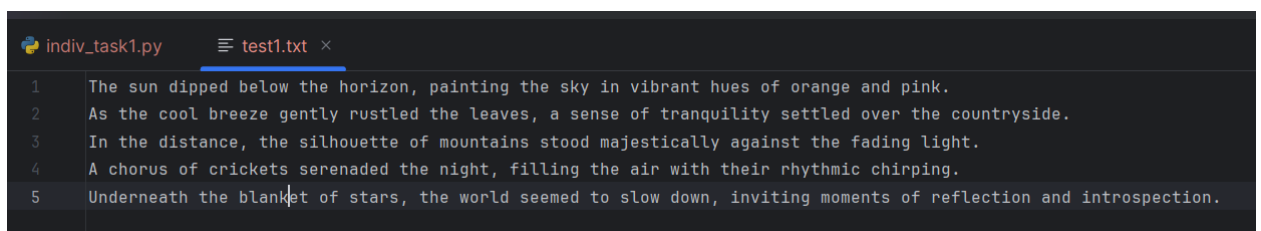
3. Выполним индивидуальные задания (вариант №20):

3.1. Написать программу, которая считывает из текстового файла три предложения и выводит их в обратном порядке:



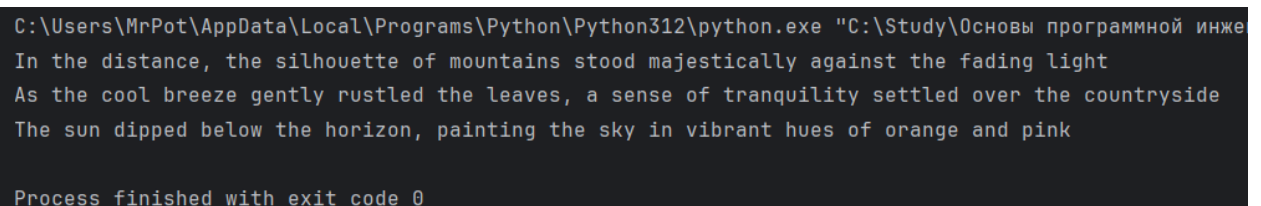
```
indiv_task1.py x test1.txt
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6  if __name__ == '__main__':
7      with open('test1.txt', 'r', encoding='utf-8') as f:
8          content = f.read()
9
10         sentences = content.split('.')
11         if len(sentences) < 2:
12             print('Not enough sentences for this program!')
13             sys.exit(1)
14
15         sentences = sentences[:3]
16         [print(sentence.strip()) for sentence in sentences[::-1]]
17
```

Рисунок 41 – Считывание предложений файла и вывод в обратном порядке



```
indiv_task1.py x test1.txt x
1  The sun dipped below the horizon, painting the sky in vibrant hues of orange and pink.
2  As the cool breeze gently rustled the leaves, a sense of tranquility settled over the countryside.
3  In the distance, the silhouette of mountains stood majestically against the fading light.
4  A chorus of crickets serenaded the night, filling the air with their rhythmic chirping.
5  Underneath the blanket of stars, the world seemed to slow down, inviting moments of reflection and introspection.
```

Рисунок 42 – Текст test1.txt



```
C:\Users\MrPot\AppData\Local\Programs\Python\Python312\python.exe "C:\Study\Основы программной инже
In the distance, the silhouette of mountains stood majestically against the fading light
As the cool breeze gently rustled the leaves, a sense of tranquility settled over the countryside
The sun dipped below the horizon, painting the sky in vibrant hues of orange and pink

Process finished with exit code 0
```

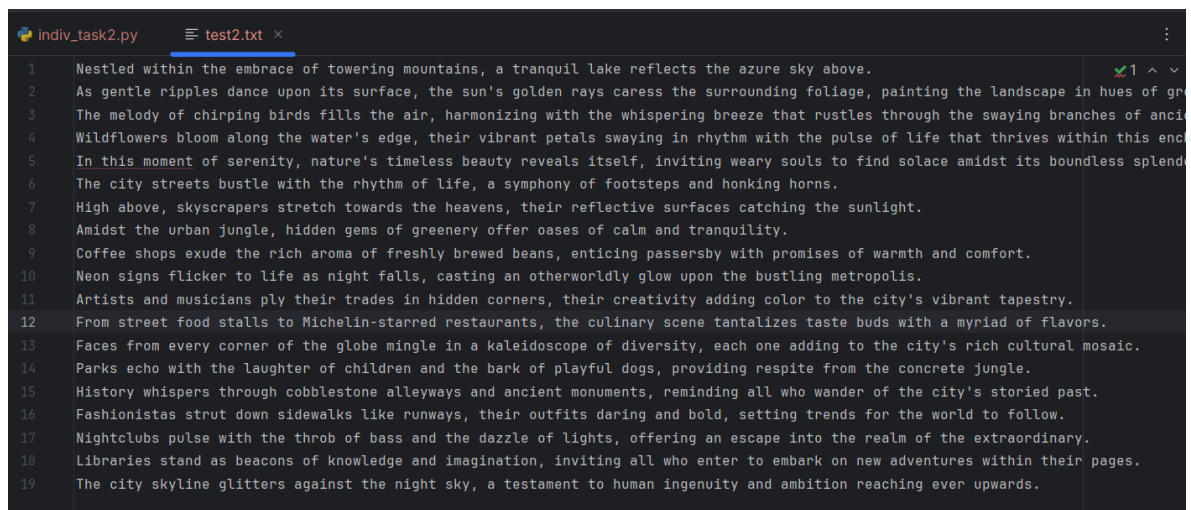
Рисунок 43 – Результат работы программы

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



```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6  if __name__ == '__main__':
7      if len(sys.argv) < 2:
8          print('Error: No file name!')
9      else:
10         filename = sys.argv[1]
11         try:
12             with open(filename, 'r', encoding='utf-8') as file:
13                 for _ in range(10):
14                     line = file.readline().strip()
15                     if not line:
16                         break
17                     print(line)
18         except FileNotFoundError:
19             print(f"Error: File '{filename}' not found.")
20
```

Рисунок 44 – Вывод первых 10 строк файла



```
1  Nestled within the embrace of towering mountains, a tranquil lake reflects the azure sky above.
2  As gentle ripples dance upon its surface, the sun's golden rays caress the surrounding foliage, painting the landscape in hues of gr
3  The melody of chirping birds fills the air, harmonizing with the whispering breeze that rustles through the swaying branches of anci
4  Wildflowers bloom along the water's edge, their vibrant petals swaying in rhythm with the pulse of life that thrives within this end
5  In this moment of serenity, nature's timeless beauty reveals itself, inviting weary souls to find solace amidst its boundless splend
6  The city streets bustle with the rhythm of life, a symphony of footsteps and honking horns.
7  High above, skyscrapers stretch towards the heavens, their reflective surfaces catching the sunlight.
8  Amidst the urban jungle, hidden gems of greenery offer oases of calm and tranquility.
9  Coffee shops exude the rich aroma of freshly brewed beans, enticing passersby with promises of warmth and comfort.
10 Neon signs flicker to life as night falls, casting an otherworldly glow upon the bustling metropolis.
11 Artists and musicians ply their trades in hidden corners, their creativity adding color to the city's vibrant tapestry.
12 From street food stalls to Michelin-starred restaurants, the culinary scene tantalizes taste buds with a myriad of flavors.
13 Faces from every corner of the globe mingle in a kaleidoscope of diversity, each one adding to the city's rich cultural mosaic.
14 Parks echo with the laughter of children and the bark of playful dogs, providing respite from the concrete jungle.
15 History whispers through cobblestone alleyways and ancient monuments, reminding all who wander of the city's storied past.
16 Fashionistas strut down sidewalks like runways, their outfits daring and bold, setting trends for the world to follow.
17 Nightclubs pulse with the throb of bass and the dazzle of lights, offering an escape into the realm of the extraordinary.
18 Libraries stand as beacons of knowledge and imagination, inviting all who enter to embark on new adventures within their pages.
19 The city skyline glitters against the night sky, a testament to human ingenuity and ambition reaching ever upwards.
```

Рисунок 45 – Содержимое файла test2.txt

```
PS C:\Study\Основы программной инженерии\2.15\fundamentals_of_software_engineering_lab2_15> py .\indiv_task2.py test2.txt
Nestled within the embrace of towering mountains, a tranquil lake reflects the azure sky above.
As gentle ripples dance upon its surface, the sun's golden rays caress the surrounding foliage, painting the landscape in hues of green and gold.
The melody of chirping birds fills the air, harmonizing with the whispering breeze that rustles through the swaying branches of ancient trees.
Wildflowers bloom along the water's edge, their vibrant petals swaying in rhythm with the pulse of life that thrives within this enchanting realm.
In this moment of serenity, nature's timeless beauty reveals itself, inviting weary souls to find solace amidst its boundless splendor.
The city streets bustle with the rhythm of life, a symphony of footsteps and honking horns.
High above, skyscrapers stretch towards the heavens, their reflective surfaces catching the sunlight.
Amidst the urban jungle, hidden gems of greenery offer oases of calm and tranquility.
Coffee shops exude the rich aroma of freshly brewed beans, enticing passersby with promises of warmth and comfort.
Neon signs flicker to life as night falls, casting an otherworldly glow upon the bustling metropolis.
PS C:\Study\Основы программной инженерии\2.15\fundamentals_of_software_engineering_lab2_15> █
```

Рисунок 46 – Вывод строк существующего файла

```
PS C:\Study\Основы программной инженерии\2.15\fundamentals_of_software_engineering_lab2_15> py .\indiv_task2.py test22323.txt
Error: File 'test22323.txt' not found.
PS C:\Study\Основы программной инженерии\2.15\fundamentals_of_software_engineering_lab2_15> █
```

Рисунок 47 – Вывод строк несуществующего файла

4. Сольем ветки develop и main/master и отправим изменения на удаленный репозиторий:

```
PS C:\Study\Основы программной инженерии\2.15\fundamentals_of_software_engineering_lab2_15> git log --oneline
caeabb7 (HEAD -> develop) individual task 2
38f4715 individual task 1
b413c95 17 example of lab
4742724 16 example of lab
6507227 15 example of lab
4c886b1 14 example of lab
6961588 13 example of lab
44744f9 12 example of lab
ce5f8c3 11 example of lab
e431527 10 example of lab
04a0892 9 example of lab
bf5258a 8 example of lab
4a26920 7 example of lab
a67f85c 6 example of lab
d3e2f1d 5 example of lab
4a0f6de 4 example of lab
518fa37 3 example of lab
524d561 2 example of lab
a2cde76 1 example of lab
abde93b (origin/main, origin/HEAD, main) Update README.md
8939330 Initial commit
PS C:\Study\Основы программной инженерии\2.15\fundamentals_of_software_engineering_lab2_15>
```

Рисунок 48 – История коммитов


```

PS C:\Study\Основы программной инженерии\2.15\fundamentals_of_software_engineering_lab2_15> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
PS C:\Study\Основы программной инженерии\2.15\fundamentals_of_software_engineering_lab2_15> git merge develop
Updating abde93b..caeabb7
Fast-forward
 .gitignore      | 2 +-
 ex1.py          | 9 ++++++++
 ex10.py         | 7 ++++++
 ex11.py         | 7 ++++++
 ex12.py         | 8 ++++++
 ex13.py         | 8 ++++++
 ex14.py         | 7 ++++++
 ex15.py         | 8 ++++++
 ex16.py         | 9 ++++++++
 ex17.py         | 22 ++++++++++++++++++++++
 ex2.py          | 6 ++++++
 ex3.py          | 10 ++++++++
 ex4.py          | 7 ++++++
 ex5.py          | 9 ++++++++
 ex6.py          | 8 ++++++
 ex7.py          | 10 ++++++++
 ex8.py          | 10 ++++++++
 ex9.py          | 7 ++++++
 indiv_task1.py  | 16 ++++++++++++++++++

```

Рисунок 49 – Слияние веток

```

PS C:\Study\Основы программной инженерии\2.15\fundamentals_of_software_engineering_lab2_15> git push origin main
Enumerating objects: 66, done.
Counting objects: 100% (66/66), done.
Delta compression using up to 12 threads
Compressing objects: 100% (63/63), done.
Writing objects: 100% (64/64), 9.37 KiB | 1.56 MiB/s, done.
Total 64 (delta 24), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (24/24), completed with 1 local object.
To https://github.com/PoTtaTto/fundamentals\_of\_software\_engineering\_lab2\_15
 abde93b..caeabb7  main -> main
PS C:\Study\Основы программной инженерии\2.15\fundamentals_of_software_engineering_lab2_15>

```

Рисунок 50 – Отправка изменений на удаленный репозиторий

Ответы на контрольные вопросы:

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

Чтобы открыть файл в языке Python только для чтения, используйте функцию `open()` с режимом доступа `'r'`, `'rb'`. Например: `open('file.txt', 'r')`.

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

Для открытия файла только для записи в Python используйте функцию `open()` с режимом доступа `'w'`, `'wb'`. Например: `open('file.txt', 'w')`.

3. Как прочитать данные из файла в языке Python?

Для чтения данных из файла в Python можно использовать методы `read()`, `readline()` или `readlines()` объекта файла, который возвращается функцией `open()`.

4. Как записать данные в файл в языке Python?

Запись данных в файл в Python можно осуществить с помощью метода `write()` объекта файла, который возвращается функцией `open()`.

5. Как закрыть файл в языке Python?

Чтобы закрыть файл в Python, используйте метод `close()` объекта файла.

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

Конструкция `with ... as` в Python используется для создания контекстного менеджера, который автоматически управляет ресурсами (например, файлами), освобождая их после использования. Она может быть также использована, например, для работы с сетевыми соединениями или базами данных.

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

<code>file.close()</code>	закрывает открытый файл
---------------------------	-------------------------

<code>file.fileno()</code>	возвращает целочисленный дескриптор файла
<code>file.flush()</code>	очищает внутренний буфер
<code>file.isatty()</code>	возвращает True, если файл привязан к терминалу
<code>file.next()</code>	возвращает следующую строку файла
<code>file.read(n)</code>	чтение первых n символов файла
<code>file.readline()</code>	читает одну строчку строки или файла
<code>file.readlines()</code>	читает и возвращает список всех строк в файле
<code>file.seek(offset[,whence])</code>	устанавливает текущую позицию в файле
<code>file.seekable()</code>	проверяет, поддерживает ли файл случайный доступ. Возвращает True, если да
<code>file.tell()</code>	возвращает текущую позицию в файле
<code>file.truncate(n)</code>	уменьшает размер файл. Если n указала, то файл обрезается до n байт, если нет — до текущей позиции
<code>file.write(str)</code>	добавляет строку str в файл
<code>file.writelines(sequence)</code>	добавляет последовательность строк в файл

Таблица 1 – Все доступные методы файла в Python

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

- `os.path`: Этот модуль предоставляет функции для работы с путями к файлам и директориям, такие как `os.path.exists()`, `os.path.abspath()`, `os.path.dirname()`, `os.path.basename()`, `os.path.join()` и другие.
- `os.listdir(path)`: Возвращает список файлов и директорий в указанной директории.
- `os.getcwd()`: Возвращает текущий рабочий каталог.
- `os.chdir(path)`: Изменяет текущий рабочий каталог.
- `os.path.isfile(path)`: Проверяет, является ли путь к файлу.
- `os.path.isdir(path)`: Проверяет, является ли путь к директории.