

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»
Вариант 9

Выполнил:
Дудкин Константин Александрович
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и
вычислительная техника»
направление «Программное
обеспечение средств вычислительной
техники и автоматизированных
систем»,
очная форма обучения

(подпись)

Руководитель практики:
Кандидат технических наук, доцент
кафедры инфокоммуникаций, доцент
Воронкин Роман Александрович

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2024 г.

Тема: Работа с файлами в языке Python

Цель: Приобретение навыков по работе с текстовыми файлами при написании программ с помощью языка программирования Python версии 3.x, изучение методов модуля os для работы с файловой системой, получение аргументов командной строки

Порядок выполнения работы

Создал репозиторий, оформил его по модели ветвления git-flow и дополнил его файлом .gitignore

Проработал примеры:

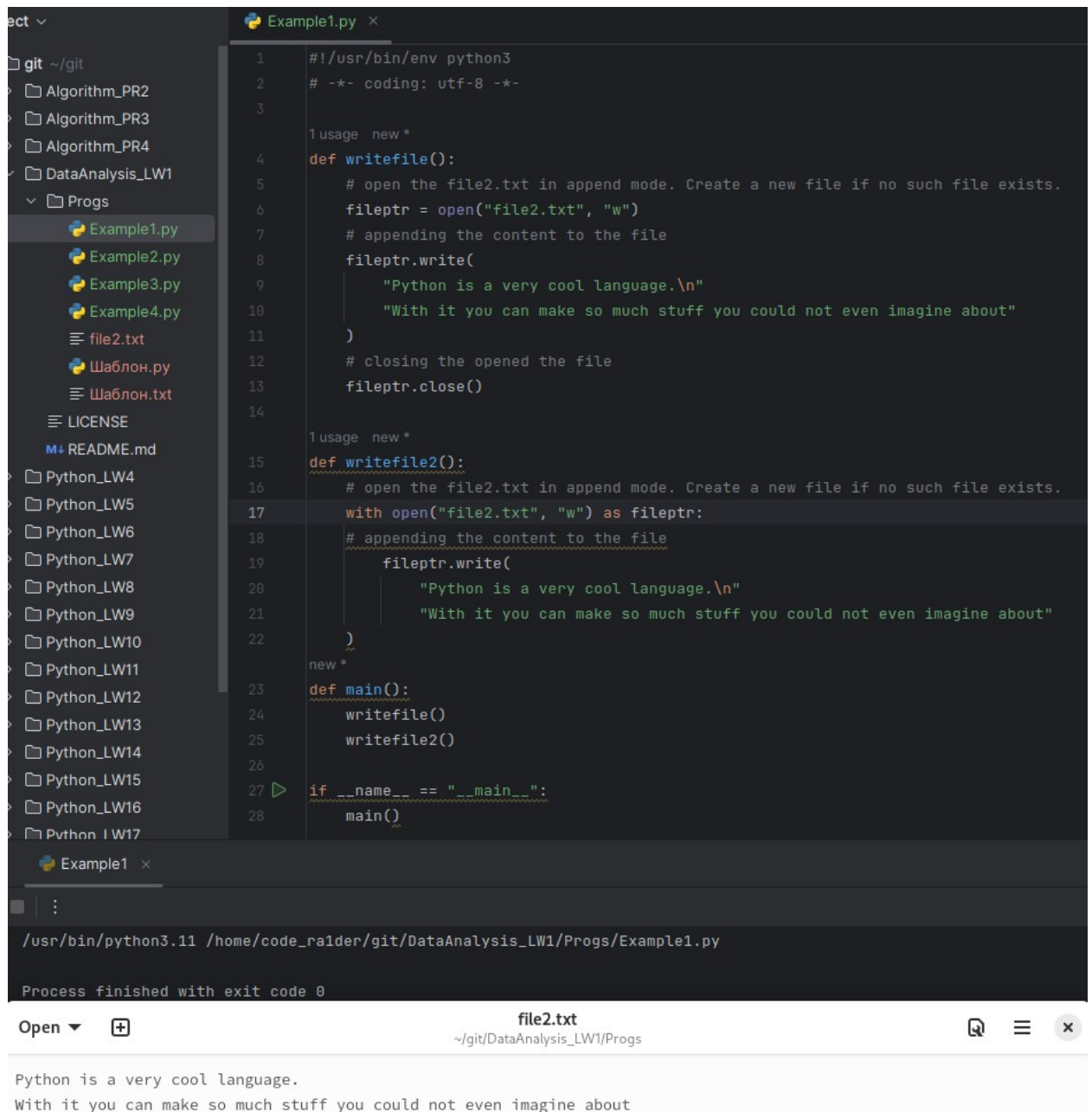


Рисунок 1. Пример 1

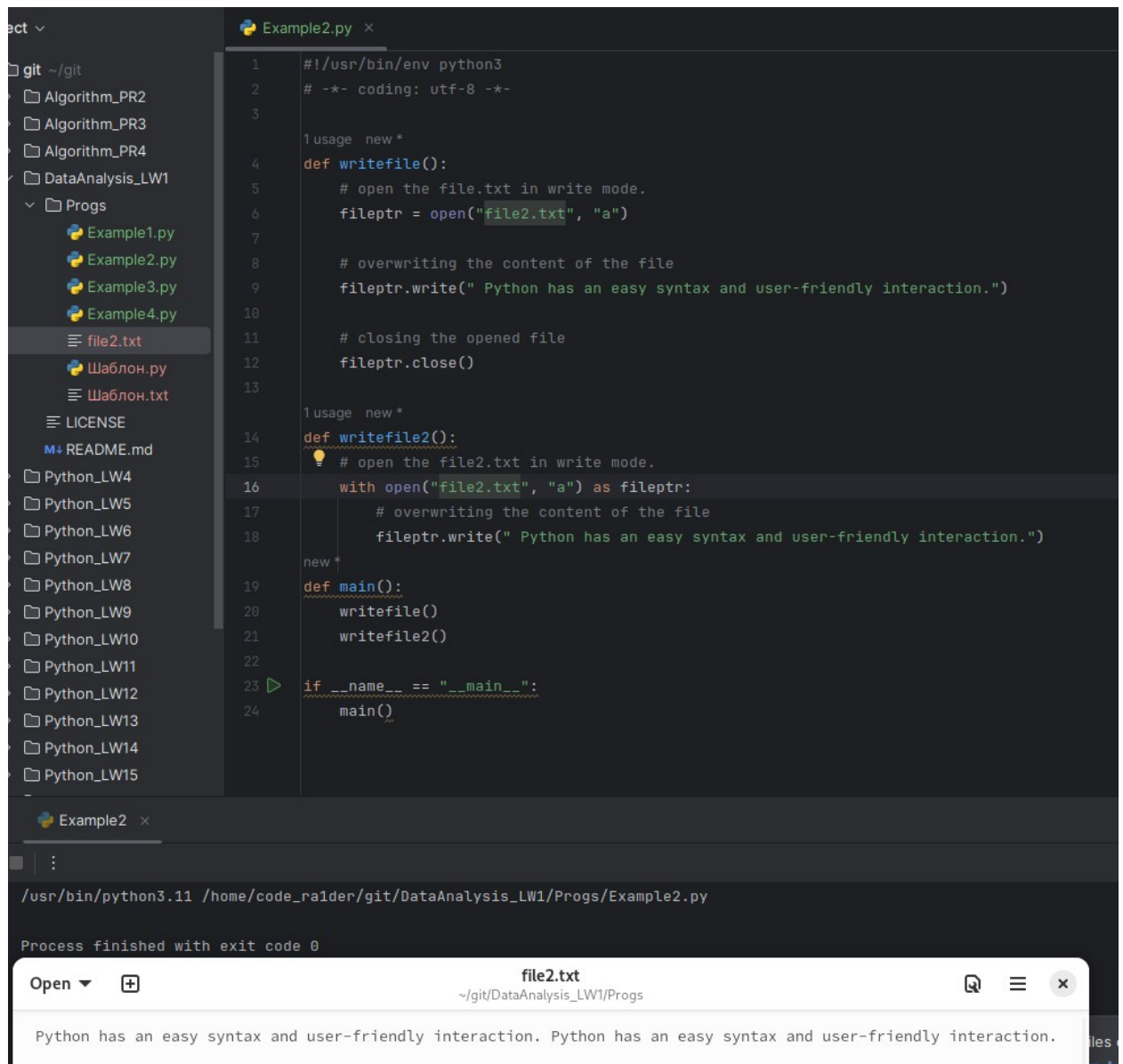


Рисунок 2. Пример 2

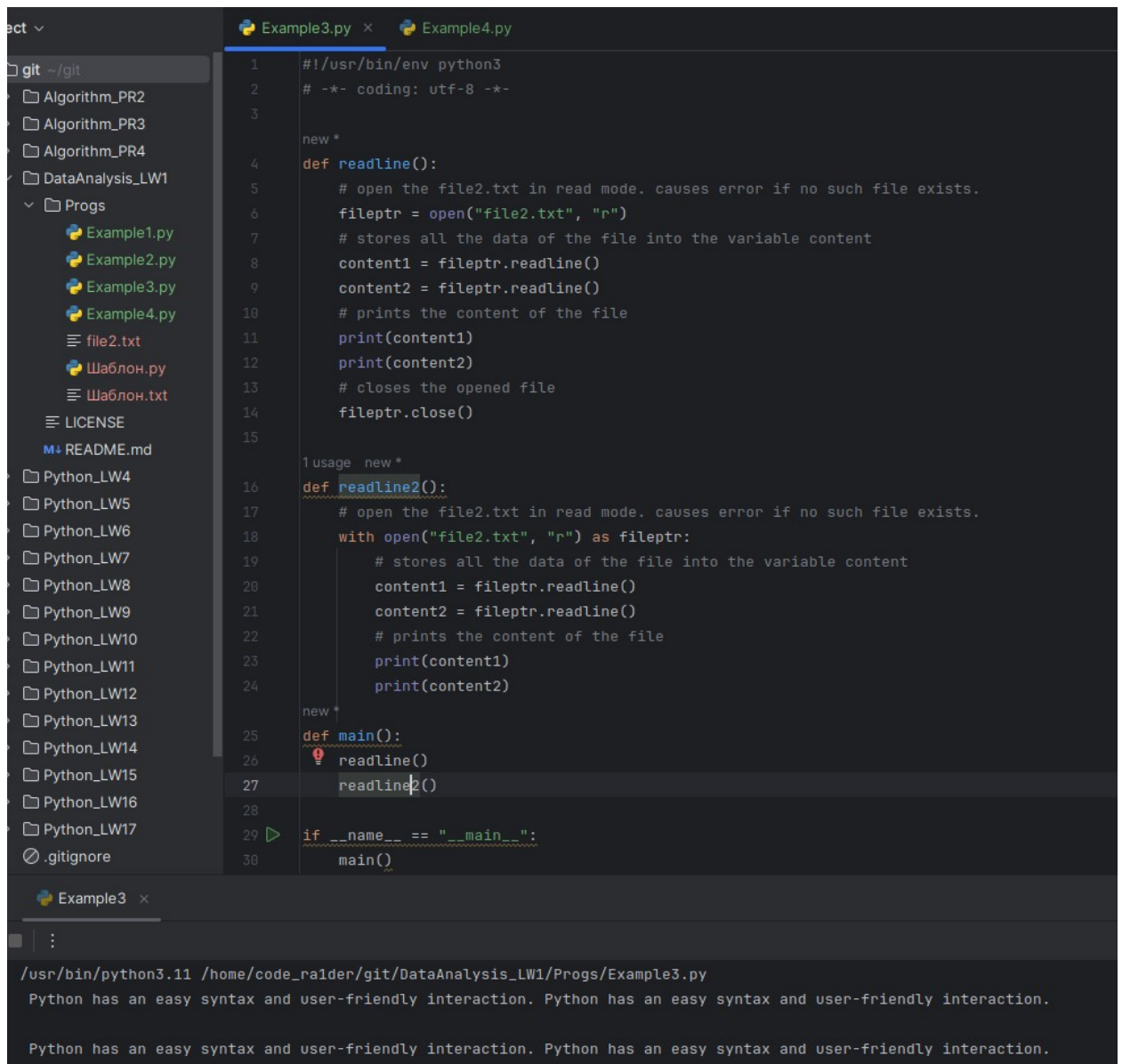


Рисунок 3. Пример 3

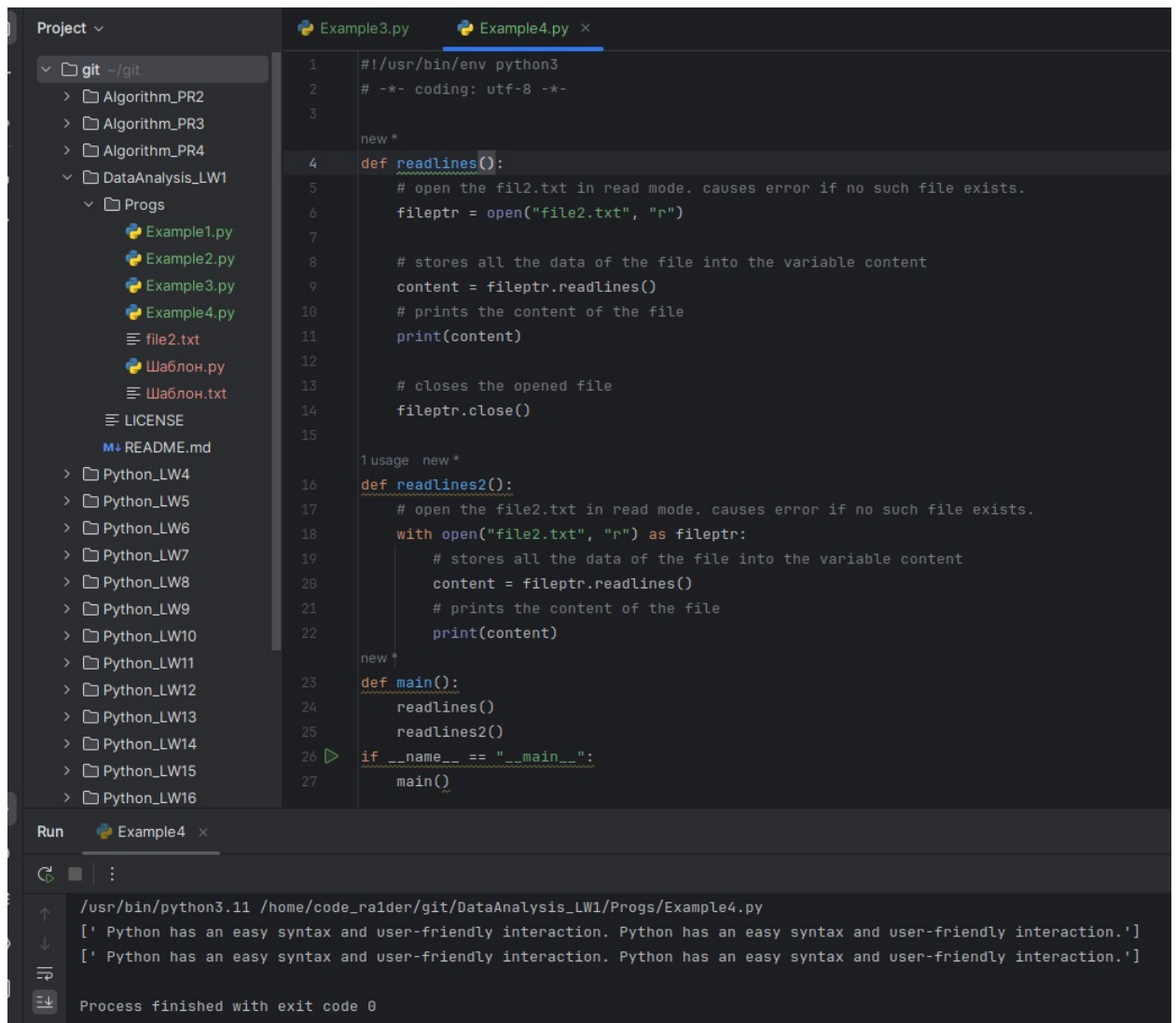
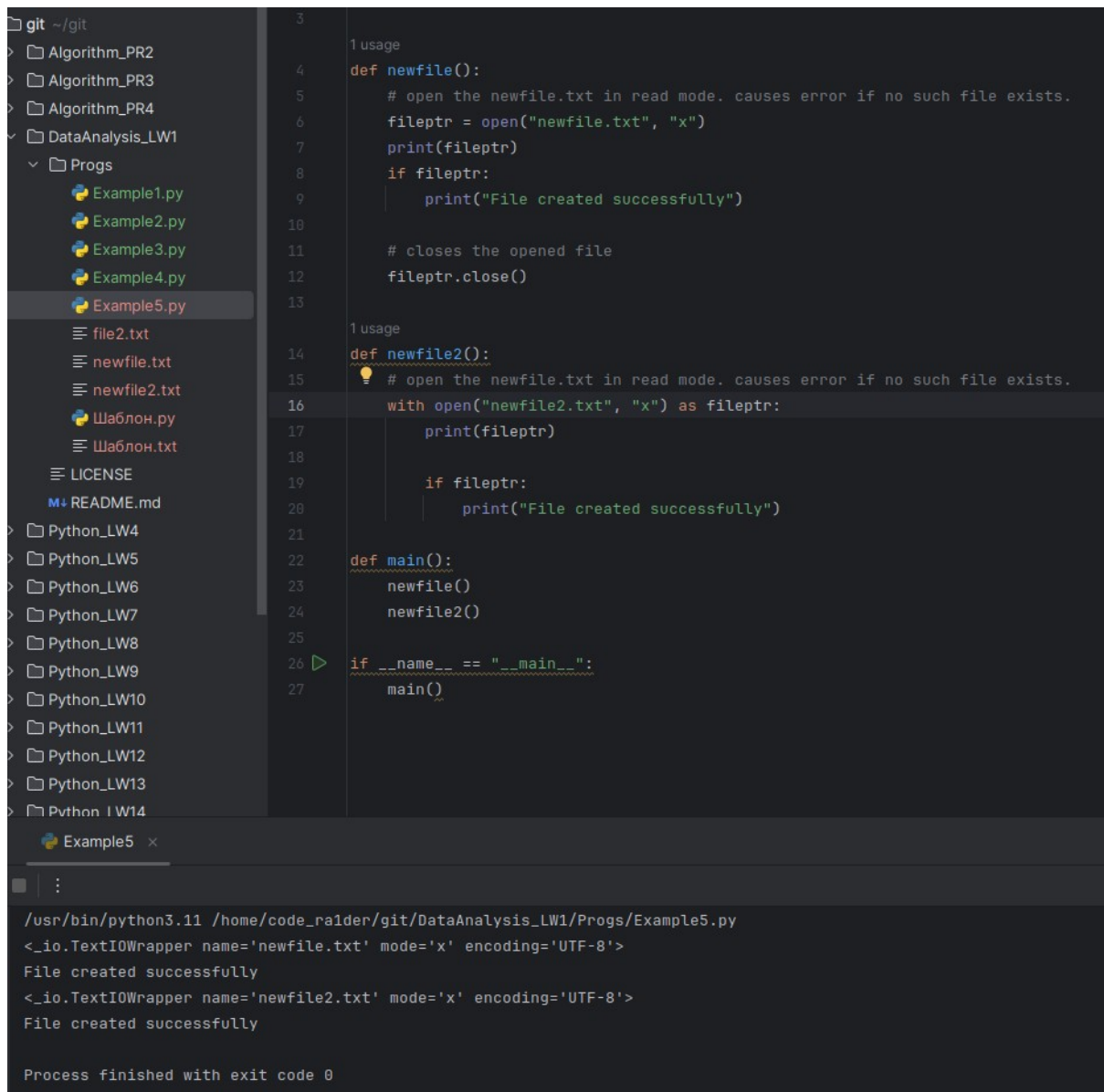


Рисунок 4. Пример 4



The image shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a directory structure with folders like 'Algorithm_PR2', 'Algorithm_PR3', 'Algorithm_PR4', 'DataAnalysis_LW1', 'Python_LW4', 'Python_LW5', 'Python_LW6', 'Python_LW7', 'Python_LW8', 'Python_LW9', 'Python_LW10', 'Python_LW11', 'Python_LW12', 'Python_LW13', and 'Python_LW14'. The 'DataAnalysis_LW1' folder is expanded, showing a subfolder 'Progs' with files 'Example1.py', 'Example2.py', 'Example3.py', 'Example4.py', and 'Example5.py'. The 'Example5.py' file is selected. The code editor shows the following Python code:

```
3
4
5 1 usage
6
7 def newfile():
8     # open the newfile.txt in read mode. causes error if no such file exists.
9     fileptr = open("newfile.txt", "x")
10    print(fileptr)
11    if fileptr:
12        print("File created successfully")
13
14    # closes the opened file
15    fileptr.close()
16
17 1 usage
18
19 def newfile2():
20     # open the newfile2.txt in read mode. causes error if no such file exists.
21     with open("newfile2.txt", "x") as fileptr:
22         print(fileptr)
23         if fileptr:
24             print("File created successfully")
25
26 def main():
27     newfile()
28     newfile2()
29
30 if __name__ == "__main__":
31     main()
```

The output of the script is shown in the bottom panel:

```
/usr/bin/python3.11 /home/code_raider/git/DataAnalysis_LW1/Progs/Example5.py
<_io.TextIOWrapper name='newfile.txt' mode='x' encoding='UTF-8'>
File created successfully
<_io.TextIOWrapper name='newfile2.txt' mode='x' encoding='UTF-8'>
File created successfully

Process finished with exit code 0
```

Рисунок 5. Пример 5

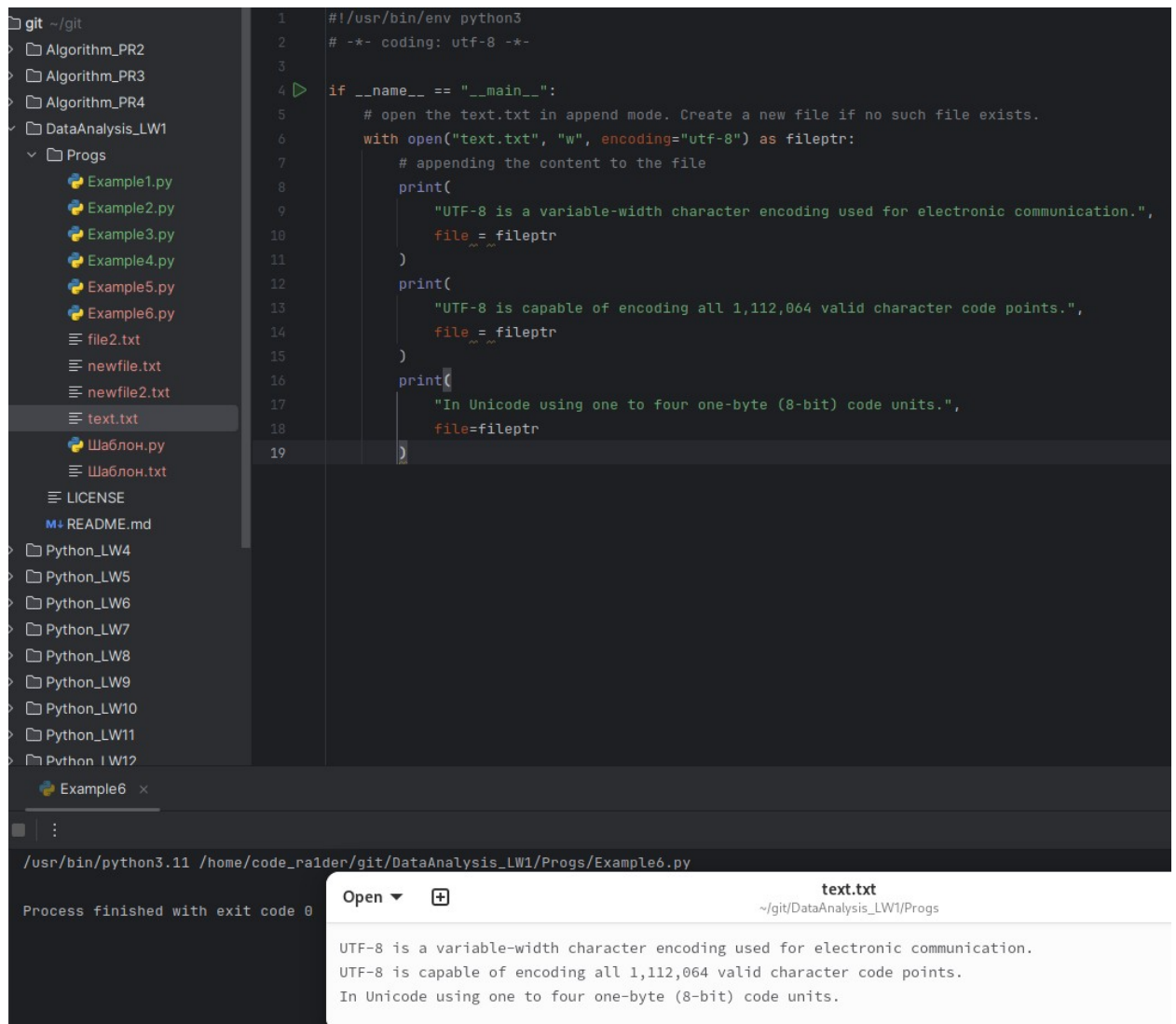


Рисунок 6. Пример 6

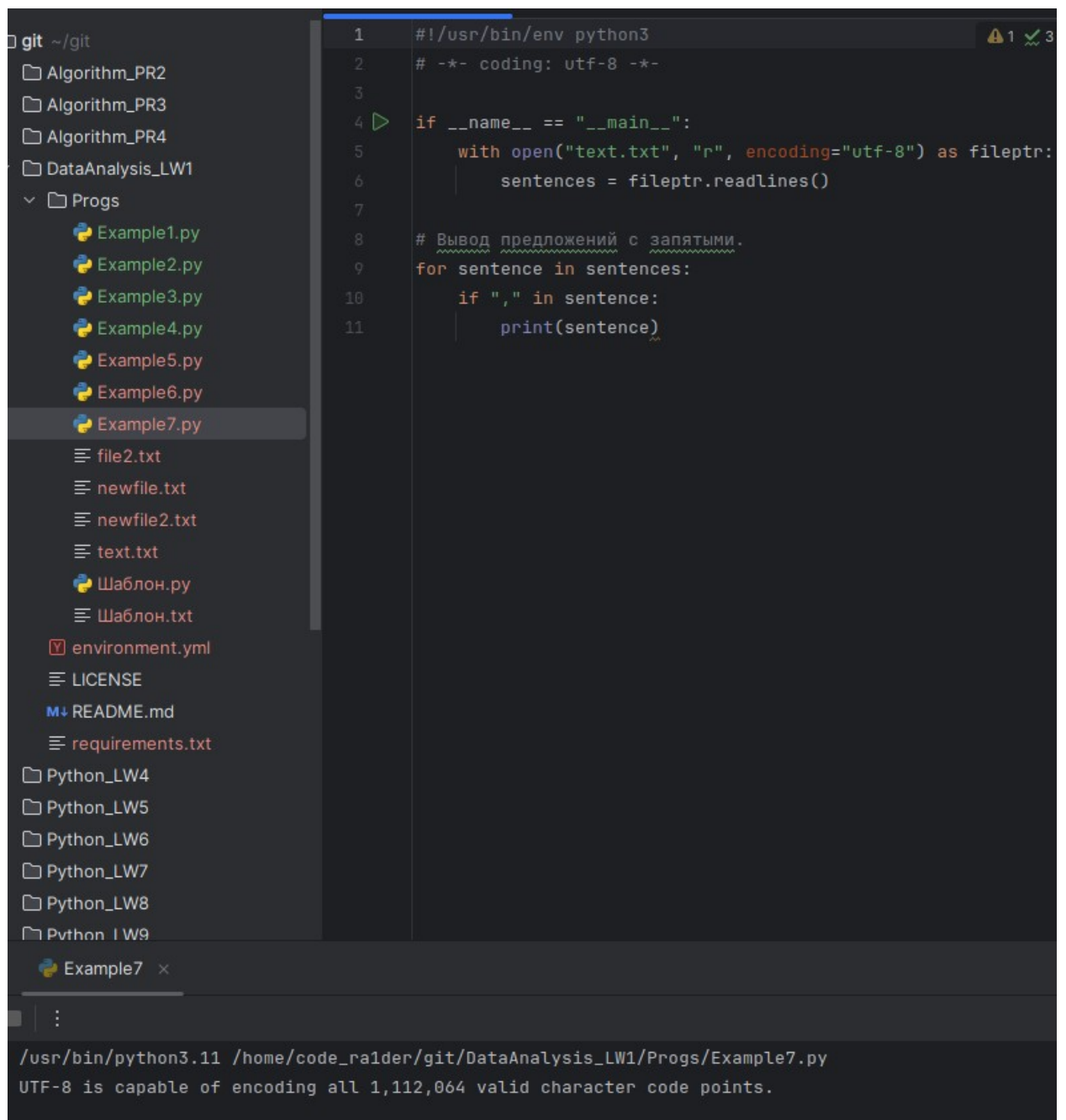


Рисунок 7. Пример 7

The image shows a code editor with a file explorer on the left and a terminal at the bottom. The file explorer shows a directory structure with folders like 'Algorithm_PR2', 'Algorithm_PR3', 'Algorithm_PR4', 'DataAnalysis_LW1', and 'Progs'. Under 'Progs', there are several Python files, with 'Example8.py' selected. The code editor displays the following Python code:

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  1 usage
5  def readmode():
6      # open the file file2.txt in read mode
7      with open("file2.txt", "r") as fileptr:
8          # initially the filepointer is at 0
9          print("The filepointer is at byte :", fileptr.tell())
10         # changing the file pointer location to 10
11         fileptr.seek(10);
12         # tell() returns the location of the fileptr.
13         print("After reading, the filepointer is at:", fileptr.tell())
14
15  def main():
16      readmode()
17
18  if __name__ == "__main__":
19      main()
```

The terminal at the bottom shows the command `/usr/bin/python3.11 /home/code_raider/git/DataAnalysis_LW1/Progs/Example8.py` and its output:

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

Process finished with exit code 0
```

Рисунок 8. Пример 8

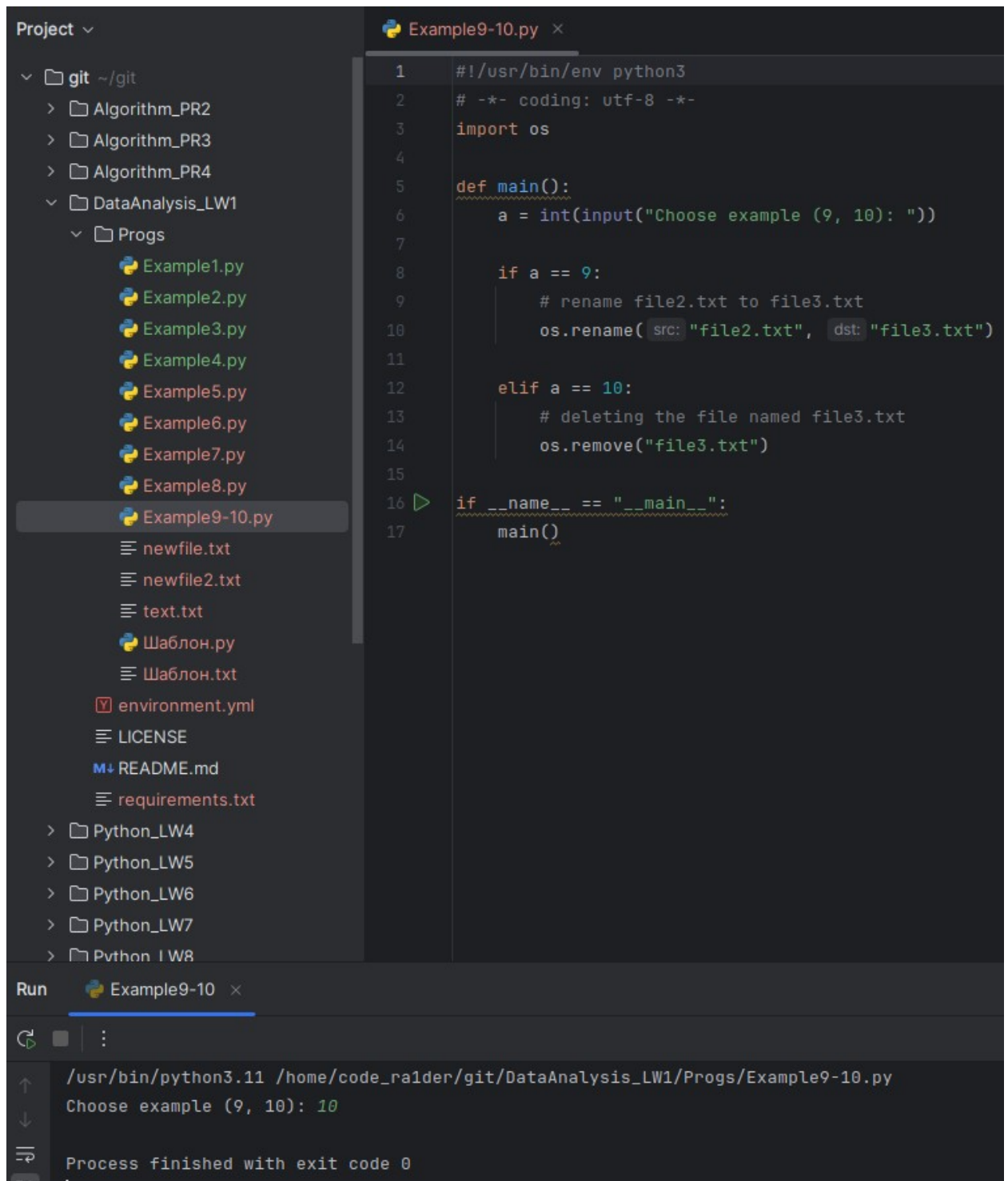


Рисунок 9. Примеры 9 и 10 (сначала файл изменил свое название, а потом был удален)

The image shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with folders 'Algorithm_PR4' and 'DataAnalysis_LW1'. Under 'DataAnalysis_LW1', there is a folder 'Progs' containing several Python files, including 'Example11-14.py' which is selected. Below the 'Progs' folder are files like 'newfile.txt', 'newfile2.txt', 'text.txt', 'Шаблон.py', 'Шаблон.txt', 'environment.yml', 'LICENSE', 'README.md', and 'requirements.txt'. Below these are folders 'Python_LW4' through 'Python_LW10'. The code editor on the right shows the content of 'Example11-14.py'. The code is a Python script that takes an input 'a' and performs different actions based on its value: if 'a' is 11, it creates a new directory; if 'a' is 12, it prints the current directory; if 'a' is 13, it changes the current directory to '/home/code_raider/' and prints it; if 'a' is 14, it removes the 'new' directory. The script uses the 'os' module. The execution output at the bottom shows the command to run the script, the input '14', and the message 'Process finished with exit code 0'.

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import os
5
6  def main():
7      a = int(input("Choose example (11, 12, 13, 14): "))
8
9      if a == 11:
10         # creating a new directory with the name new
11         os.mkdir("new")
12
13     if a == 12:
14         path = os.getcwd()
15         print(path)
16
17     if a == 13:
18         # Changing current directory with the new directory
19         os.chdir("/home/code_raider/")
20         # It will display the current working directory
21         print(os.getcwd())
22
23     if a == 14:
24         # removing the new directory
25         os.rmdir("new")
26
27 if __name__ == "__main__":
28     main()
```

Example11 x

:

/usr/bin/python3.11 /home/code_raider/git/DataAnalysis_LW1/Progs/Example11.py
Choose example (11, 12, 13, 14): 14

Process finished with exit code 0

Рисунок 10. Примеры 11-14

The image shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with folders 'Algorithm_PR4', 'DataAnalysis_LW1', and 'Progs'. The 'Progs' folder is expanded, showing files 'Example1.py' through 'Example16.py', 'newfile.txt', 'newfile2.txt', 'text.txt', 'Шаблон.py', 'Шаблон.txt', 'environment.yml', 'LICENSE', 'README.md', and 'requirements.txt'. The 'Example15.py' file is selected. The code editor shows the following Python code:

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6  if __name__ == "__main__":
7      print("Number of arguments:", len(sys.argv), "arguments")
8      print("Argument List:", str(sys.argv))
```

The output of the script is shown in the terminal at the bottom:

```
/usr/bin/python3.11 /home/code_raider/git/DataAnalysis_LW1/Progs/Example15.py
Number of arguments: 1 arguments
Argument List: ['/home/code_raider/git/DataAnalysis_LW1/Progs/Example15.py']

Process finished with exit code 0
```

Рисунок 11. Пример 15

The image shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with folders 'Algorithm_PR4', 'DataAnalysis_LW1', and 'Progs'. The 'Progs' folder is expanded, showing files 'Example1.py' through 'Example16.py', 'newfile.txt', 'newfile2.txt', 'text.txt', 'Шаблон.py', 'Шаблон.txt', 'environment.yml', 'LICENSE', 'README.md', and 'requirements.txt'. The 'Example16.py' file is selected. The code editor shows the following Python code:

```
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
10     print("No. of arguments passed is ", len(sys.argv))
```

The output of the script is shown in the terminal at the bottom:

```
/usr/bin/python3.11 /home/code_raider/git/DataAnalysis_LW1/Progs/Example16.py
Argument #0 is /home/code_raider/git/DataAnalysis_LW1/Progs/Example16.py
No. of arguments passed is 1

Process finished with exit code 0
```

Рисунок 12. Пример 16


```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import os
5 import secrets
6 import string
7 import sys
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
```

code_ra1der@archlinux:~/git/DataAnalysis_LW1/Progs

```
(base) [code_ra1der@archlinux Progs]$ python Example17.py 6
Secret Password: aT''M=
(base) [code_ra1der@archlinux Progs]$ python Example17.py 12
Secret Password: D@di-e`@[7UF
(base) [code_ra1der@archlinux Progs]$ python Example17.py 30
Secret Password: oBHA>G3.P5LR7{=QpFB,r4)*E1]Vm0
(base) [code_ra1der@archlinux Progs]$
```

Рисунок 13. Пример 17

Выполнил индивидуальное задание (вариант 9): Необходимо создать программу, которая считывает с файла английский текст и выводит на экран слова, начинающиеся и оканчивающиеся на гласные буквы


```

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

# Написать программу, которая считывает английский текст из файла и выводит на экран
# слова текста, начинающиеся и оканчивающиеся на гласные буквы.

@ Code_Raider *
def check(word):
    check = "AEIOUaeiou"
    return len(word) > 1 and word[0] in check and word[-1] in check

@ Code_Raider
def split(text):
    for char in '.,!?!;:':
        text = text.replace(char, ' ')
    words = text.split()
    return words

1 usage @ Code_Raider
def readeng():
    with open("task1.txt", "r") as fileptr:
        text = fileptr.read()
        words = split(text)

        for word in words:
            if check(word):
                print(word)

@ Code_Raider
def main():
    readeng()

▶ if __name__ == "__main__":
    main()

```

Рисунок 14. Индивидуальное задание 1, вариант 9

Выполнил индивидуальное задание 2 (вариант 9): Необходимо сделать программу, которая считывает слова, записанные в текстовом файле, и из них составляет пароль, размер которого $8 \leq x \leq 10$, а каждое слово имеет заглавную букву, чтобы определить, из каких слов состоит созданный пароль

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

# Написать программу, которая будет открывать файл со списком
# слов, случайным образом выбирать два из них и сцеплять вместе для получения итогового
# пароля. При создании пароля нужно следовать следующему требованию: он должен состоять
# минимум из восьми символов и максимум из десяти, а каждое из используемых слов
# должно быть длиной хотя бы в три буквы. Кроме того, нужно сделать заглавными первые буквы
# обоих слов, чтобы легко можно было понять, где заканчивается одно и начинается другое.
# По завершении процесса полученный пароль должен быть отображен на экране.

import random

class Code_Ralder:
    def generate(words):
        while True:
            word1, word2 = random.sample(words, k=2)
            if len(word1) >= 3 and len(word2) >= 3:
                password = word1.capitalize() + word2.capitalize()
                if len(password) >= 8 and len(password) <= 10:
                    return password

1 usage  class Code_Ralder *
def passmaking():
    try:
        with open("task2.txt", "r") as fileptr:
            words = [line.strip() for line in fileptr.readlines()]

            password = generate(words)
            print("Generated password: ", password)
    except FileNotFoundError:
        print("File task2.txt not found. Create a new one and fill it with words (should be written apart on different lines)")
```

Рисунок 15. Индивидуальное задание 2, вариант 9 (часть 1)

The screenshot shows a code editor with a file explorer on the left, a code editor in the center, and a terminal at the bottom. The file explorer shows a list of files including Example8.py through Example17.py, newfile.txt, and newfile2.txt. The code editor shows the implementation of the password generator, including the generate method and the passmaking function. The terminal shows the output of the program, which is "Generated password: HelloWorld".

```
Example8.py 35
Example9-10.py 36
Example11-14.py 37
Example15.py 38
Example16.py 39
Example17.py 40
newfile.txt 41
newfile2.txt 42

Task2 x
:
/usr/bin/python3.11 /home/code_ralder/git/DataAnalysis_LW1/Progs/
Generated password: HelloWorld

Process finished with exit code 0
```

task2.txt
~/git/D...Tasks

geneva
hello
world
help
starts
retry
apples
forbid
creative
ear
rig
raw
begin
into
for
rewamp
restart
ends
end

Рисунок 16. Индивидуальное задание 2, вариант 9 (часть 2)

Ответы на вопросы

1. Для того, чтобы открыть файл только для чтения, можно воспользоваться следующей функцией:

```
file = open(„file.txt“, „r“)
```

2. Для того, чтобы открыть файл только для записи, можно воспользоваться предыдущей функцией с параметром „w“:

```
file = open(„file.txt“, „w“)
```

3. Для чтения данных из файла используются следующие методы:

```
read()
```

```
readline()
```

```
readlines()
```

4. Для записи данных в файл можно использовать метод `write()`

5. Чтобы закрыть файл в Python нужно использовать метод `close()`

6. Конструкция `with ... as` используется в Python используется для упрощения и автоматизации управления контекстом выполнения. Она позволяет упростить работу как с файлами, так и с другими ресурсами и программами, тоже требующими управление контекстом (например, базы данных)

7. Кроме указанных ранее методов, для работы с файлами используются также и ряд других методов: `flush()`, `seek()`, `tell()` и др. Они позволяют редактировать и манипулировать файл как угодно пользователю

8. В модуле `os` существует множество функций, позволяющих редактировать файлы системы как угодно пользователю. Наиболее понятными и используемыми являются такие функции, как `os.rename()`, `os.remove()`, `os.mkdir()`, `os.rmdir()` и т. д.

Вывод: В течении всей лабораторной работы были приобретены навыки по манипуляции файлами в программах Python, изучен модуль `os` и методы чтения и записи информации в файл