

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЁТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №1
дисциплины «Анализ данных»
Вариант 28

Выполнил:
Репкин Александр Павлович
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и
вычислительная техника»,
направленность (профиль)
«Программное обеспечение средств
вычислительной
техники и автоматизированных
систем», очная форма обучения

(подпись)

Руководитель практики:
Воронкин Р.А., канд. техн. наук,
доцент кафедры инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2024 г.

Тема: Работа с файлами в языке Python.

Цель: приобрести навыки работы с текстовыми файлами при написании программ с помощью языка программирования Python версии 3.x, а также изучить основные методы модуля OS для работы с файловой системой и научиться получать аргументы командной строки.

Порядок выполнения работы:

1. Выполнен пример №1, в котором программно открывался в режиме записи файл (Или, если такого файла не найдено, создавался новый файл), в который, в последствии, были введены данные.

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4
5  if __name__ == '__main__':
6      # Если file2.txt существует, то открыть его в режиме редактирования, иначе - создать.
7      fileptr = open("file2.txt", "w")
8      # Запись данных в файл.
9      fileptr.write(
10         "Python is the modern day language. It makes things so simple.\nIt is the fastest-growing programing language")
11     fileptr.close() # Закрытие файла file2.txt
12
```

Рисунок 1. Код примера №1

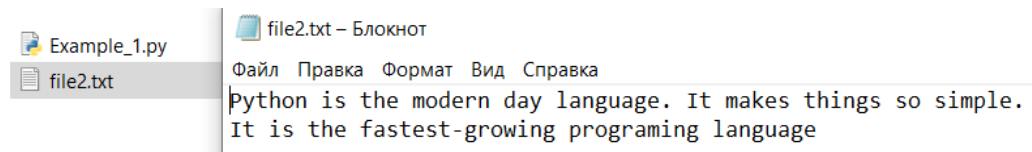


Рисунок 2. Файл, созданный программой и его содержимое

2. Выполнен пример №2, в котором программно открывался в режиме записи в конец файл (Или, если такого файла не найдено, создавался новый файл), в который, в последствии, были введены данные.

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4
5  if __name__ == '__main__':
6      # Если file2.txt существует, то открыть его в режиме добавления информации в конец, иначе - создать.
7      fileptr = open("file2.txt", "a")
8      # Запись данных в файл.
9      fileptr.write(" Python has an easy syntax and user-friendly interaction.")
10     fileptr.close() # Закрытие файла file2.txt
11
```

Рисунок 3. Код примера №2, без использования with

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4
5  if __name__ == '__main__':
6      # Открытие файла в режиме добавления информации в конец.
7      with open("file2.txt", "a") as fileptr:
8          # Добавление текста в конец файла.
9          fileptr.write(
10             " Python has an easy syntax and user-friendly interaction.")

```

Рисунок 4. Код примера №2, с использованием with

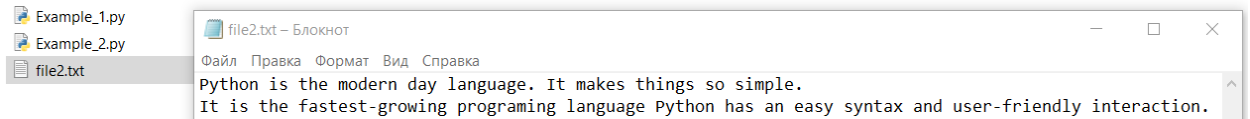


Рисунок 5. Содержимое file2.txt изменилось обоими способами

3. Выполнен пример №3, в котором программно открывался в режиме чтения файл (Или, если такого файла не найдено, то выводилась ошибка), данные из которого были прочитаны и выведены построчно.

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4
5  if __name__ == '__main__':
6      # Первый способ.
7      # Если file2.txt существует, то открыть его в режиме чтения, иначе - ошибка.
8      fileptr = open("file2.txt", "r")
9      # Запись данных из файла построчно в переменные.
10     content1 = fileptr.readline()
11     content2 = fileptr.readline()
12     # Вывод строк файла.
13     print(content1)
14     print(content2)
15     fileptr.close() # Закрытие файла file2.txt
16

```

Рисунок 6. Код примера №3, без использования with

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4
5  if __name__ == '__main__':
6      # Второй способ.
7      # Открытие файла в режиме чтения.
8      with open("file2.txt", "r") as fileptr:
9          # Запись данных из файла построчно в переменные.
10         content1 = fileptr.readline()
11         content2 = fileptr.readline()
12         # Вывод строк файла.
13         print(content1)
14         print(content2)
15

```

Рисунок 7. Код примера №3, с использованием with

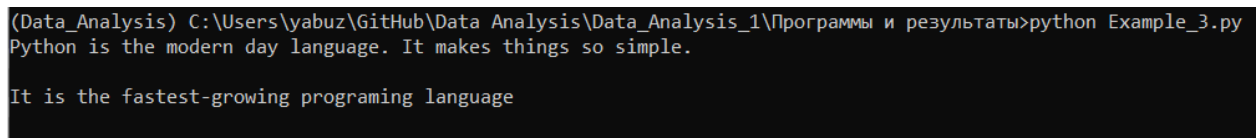


Рисунок 8. Выполнение примера №3

4. Выполнен пример №4, в котором программно открывался в режиме чтения файл (Или, если такого файла не найдено, то выводилась ошибка), все строки из которого были получены сразу.

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4
5  if __name__ == '__main__':
6      # Первый способ.
7      # Если file2.txt существует, то открыть его в режиме чтения, иначе - ошибка.
8      fileptr = open("file2.txt", "r")
9      # Запись сразу всех строк из файла в переменную.
10     content = fileptr.readlines()
11     # Вывод строк файла.
12     print(content)
13     fileptr.close() # Закрытие файла file2.txt
14
```

Рисунок 9. Код примера №4 без использования with

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4
5  if __name__ == '__main__':
6      # Второй способ.
7      # Открытие файла в режиме чтения.
8      with open("file2.txt", "r") as fileptr:
9          # Запись сразу всех строк из файла в переменную.
10         content = fileptr.readlines()
11         # Вывод строк файла.
12         print(content)
13
```

Рисунок 10. Код примера №4 с использованием with

```
(Data_Analysis) C:\Users\yabuz\GitHub\Data_Analysis\Data_Analysis_1\Программы и результаты>python Example_4.py
['Python is the modern day language. It makes things so simple.\n', 'It is the fastest-growing programming language']
```

Рисунок 11. Выполнение примера №4

5. Выполнен пример №5, в котором программно открывался в режиме чтения файл (Или, если такого файла не найдено, то выводилась ошибка), все строки из которого были получены сразу.

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4
5  if __name__ == '__main__':
6      # Первый способ.
7      # Если newfile.txt существует, то ошибка, иначе - создать.
8      fileptr = open("newfile.txt", "x")
9      print(fileptr)
10     if fileptr:
11         print("File created successfully.")
12     # Закрытие файла.
13     fileptr.close()
14
```

Рисунок 12. Код примера №5 без использования with

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4
5  if __name__ == '__main__':
6      # Второй способ.
7      # Создание файла, если такого ещё не существует.
8      with open("newfile.txt", "x") as fileptr:
9          print(fileptr)
10         if fileptr:
11             print("File created successfully")
12

```

Рисунок 13. Код примера №5 с использованием with

```

(Data_Analysis) C:\Users\yabuz\GitHub\Data Analysis\Data_Analysis_1\Программы и результаты>python Example_5.py
<_io.TextIOWrapper name='newfile.txt' mode='x' encoding='cp1251'>
File created successfully.

```

Рисунок 14. Выполнение примера №5

6. Выполнен пример №6, в котором программно открывался в режиме записи файл (Или, если такого файла не найдено, то создавался новый файл), в который вводились строки при помощи print с параметром file, вместо write.

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4
5  if __name__ == '__main__':
6      # Открытие файла в режиме записи. Если файл не найден - создаётся с именем text.txt.
7      # encoding указывает используемую в файле кодировку.
8      with open("text.txt", "w", encoding="utf-8") as fileptr:
9          # Запись данных в файл.
10         print("UTF-8 is a variable-width character encoding used for electronic communication.", file=fileptr)
11         print("UTF-8 is capable of encoding all 1,112,064 valid character codepoints.", file=fileptr)
12         print("In Unicode using one to four one-byte (8-bit) code units.", file=fileptr)
13

```

Рисунок 15. Код примера №6

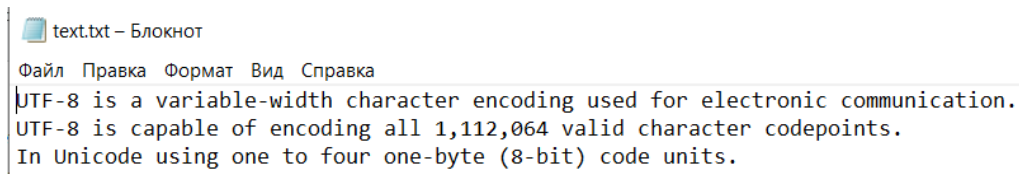


Рисунок 16. Выполнение примера №6

7. Выполнен пример №7, в котором программно открывался в режиме чтения файл (Или, если такого файла не найдено, то выводилась ошибка), с дополнительным параметром encoding, указывающим, как нужно дешифровать содержимое файла.

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4
5  if __name__ == "__main__":
6      # Открытие файла в режиме чтения и запись данных из него в переменную.
7      # encoding указывает используемую в файле кодировку.
8      with open("text.txt", "r", encoding="utf-8") as fileptr:
9          sentences = fileptr.readlines()
10         # Вывод предложений с запятыми.
11         for sentence in sentences:
12             if "," in sentence:
13                 print(sentence)
14

```

Рисунок 17. Код примера №7

```

(Data_Analysis) C:\Users\yabuz\GitHub\Data Analysis\Data_Analysis_1\Программы и результаты>python Example_7.py
UTF-8 is capable of encoding all 1,112,064 valid character codepoints.

```

Рисунок 18. Выполнение примера №7

8. Выполнен пример №8, в котором программно открывался в режиме чтения файл (Или, если такого файла не найдено, то выводилась ошибка). В консоль выводилась информация о текущем местоположении указателя (На каком байте). Его местоположение изменялось при помощи seek.

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4
5  if __name__ == "__main__":
6      # Открытие файла в режиме чтения.
7      with open("file2.txt", "r") as fileptr:
8          # Указатель сейчас на 0 байте.
9          print("The filepointer is at byte :", fileptr.tell())
10         # Перемещение указателя на 10 байт.
11         fileptr.seek(10)
12         # tell() выводит, на каком байте сейчас указатель.
13         print("After reading, the filepointer is at:", fileptr.tell())
14

```

Рисунок 19. Код примера №8

```

(Data_Analysis) C:\Users\yabuz\GitHub\Data Analysis\Data_Analysis_1\Программы и результаты>python Example_8.py
The filepointer is at byte : 0
After reading, the filepointer is at: 10

```

Рисунок 20. Выполнение примера №8

9. Выполнен пример №9, в котором при помощи модуля os был переименован файл file2.txt в file3.txt.

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import os
5
6
7  if __name__ == "__main__":
8      # Модуль OS переименовывает файл file2.txt в file3.txt.
9      os.rename("file2.txt", "file3.txt")
10

```

Рисунок 21. Код примера №9

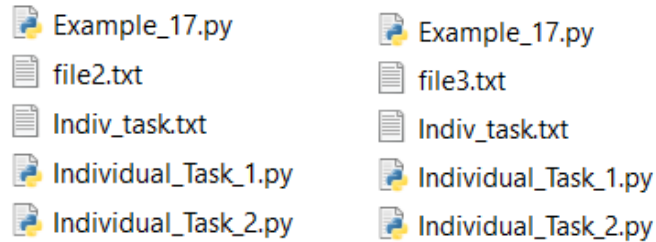


Рисунок 22. Выполнение примера №9 – замена file2.txt на file3.txt

10. Выполнен пример №10, в котором при помощи модуля os был удалён файл file3.txt.

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import os
5
6
7  if __name__ == "__main__":
8      # Модуль OS удаляет файл file3.txt.
9      os.remove("file3.txt")
10

```

Рисунок 23. Код примера №10

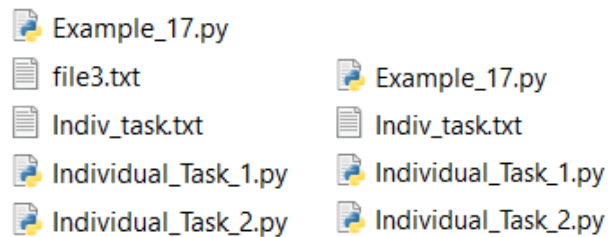


Рисунок 24. Выполнение примера №10 – удаление файла file3.txt

11. Выполнен пример №11, в котором при помощи модуля os был создан каталог new.

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import os
5
6
7  if __name__ == "__main__":
8      # Модуль OS создаёт новый каталог new.
9      os.mkdir("new")
10

```

Рисунок 25. Код примера №11

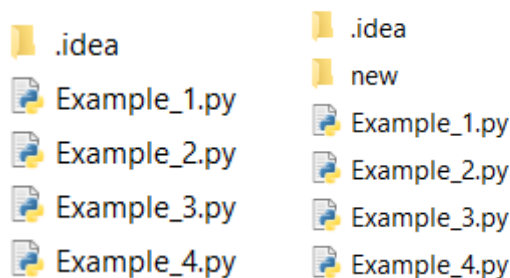


Рисунок 26. Выполнение примера №11 – создание нового каталога

12. Выполнен пример №12, в котором при помощи модуля os была выведена информация о текущем рабочем каталоге.

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import os
5
6
7  if __name__ == "__main__":
8      # Модуль OS предоставляет информацию о текущем рабочем каталоге.
9      path = os.getcwd()
10     print(path)
11

```

Рисунок 27. Код примера №12

```

(Data_Analysis) C:\Users\yabuz\GitHub\Data Analysis\Data_Analysis_1\Программы и результаты>python Example_12.py
C:\Users\yabuz\GitHub\Data Analysis\Data_Analysis_1\Программы и результаты

```

Рисунок 28. Выполнение примера №12

13. Выполнен пример №13, в котором при помощи модуля os был изменён текущий рабочий каталог.

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import os
5
6
7  if __name__ == "__main__":
8      # Модуль OS изменяет текущий рабочий каталог, перейдя в C:\Windows.
9      os.chdir("C:\\Windows")
10     print(os.getcwd())
11

```

Рисунок 29. Код примера №13


```
(Data_Analysis) C:\Users\yabuz\GitHub\Data Analysis\Data_Analysis_1\Программы и результаты>python Example_13.py
C:\Windows
```

Рисунок 30. Выполнение примера №13

14. Выполнен пример №14, удаляющий созданный программно каталог new при помощи модуля os.

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import os
5
6
7  if __name__ == "__main__":
8      # Модуль OS удаляет каталог new, если он есть в текущем рабочем каталоге.
9      os.rmdir("new")
10
```

Рисунок 31. Код примера №14

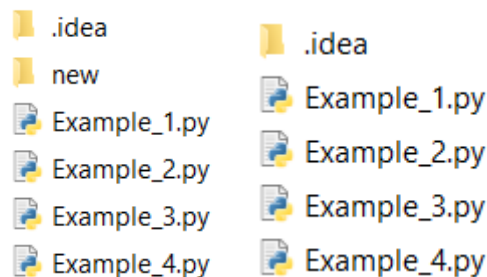


Рисунок 32. Выполнение примера №14

15. Выполнен пример №15, в котором выводились переданные программе при запуске через командную строку элементы.

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6
7  if __name__ == "__main__":
8      # Работа с аргументами(sys.argv. На 0 месте имя файла), переданными в командной строке.
9      print("Number of arguments:", len(sys.argv), "arguments")
10     print("Argument List:", str(sys.argv))
11
```

Рисунок 33. Код примера №15

```
(Data_Analysis) C:\Users\yabuz\GitHub\Data Analysis\Data_Analysis_1\Программы и результаты>python Example_15.py One Dos
Три
Number of arguments: 4 arguments
Argument List: ['Example_15.py', 'One', 'Dos', 'Три']
```

Рисунок 34. Выполнение примера №15

16. Выполнен пример №16, в котором выводились переданные программе при запуске через командную строку элементы, а также их индексы в argv.

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6
7  if __name__ == "__main__":
8      # Работа с аргументами(sys.argv. На 0 месте имя файла), переданными в командной строке.
9      for idx, arg in enumerate(sys.argv):
10         print(f"Argument #{idx} is {arg}")
11     print("Amount of arguments passed is ", len(sys.argv))
12

```

Рисунок 35. Код примера №16

```

(Data_Analysis) C:\Users\yabuz\GitHub\Data Analysis\Data_Analysis_1\Программы и результаты>python Example_16.py One Dos
Три
Argument #0 is Example_16.py
Argument #1 is One
Argument #2 is Dos
Argument #3 is Три
Amount of arguments passed is 4

```

Рисунок 36. Выполнение примера №16

17. Выполнен пример №17, в котором при помощи модуля `secrets` генерировались случайные индексы, которые находились в созданной при помощи модуля `string` строке, содержащей все символы ASCII, пунктуационные символы и цифры. Полученный набор символов образовывал случайный пароль.

```

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     # Создание пароля относительно переданной через командную строку его длины.
11     if len(sys.argv) < 2:
12         # Не передана длина пароля. Только имя файла.
13         print("The password length is not given!", file=sys.stderr)
14         sys.exit(1)
15     chars = string.ascii_letters + string.punctuation + string.digits
16     length_pwd = int(sys.argv[1])
17     result = []
18     for _ in range(length_pwd):
19         # SystemRandom - класс, использующий функцию os.urandom() для генерации
20         # случайных чисел из источников, предоставленных ОС.
21         idx = secrets.SystemRandom().randrange(len(chars))
22         result.append(chars[idx])
23     print(f"Secret Password: {''.join(result)}")
24

```

Рисунок 37. Код примера №17

```

(Data_Analysis) C:\Users\yabuz\GitHub\Data Analysis\Data_Analysis_1\Программы и результаты>python Example_17.py 10
Secret Password: K|*/Q.lmte

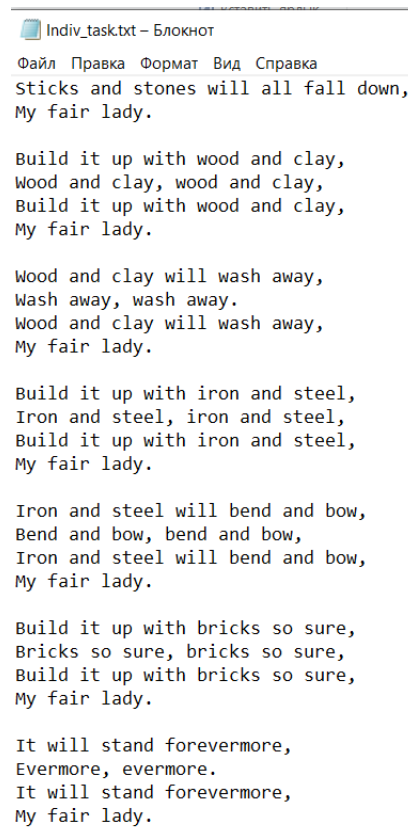
```

Рисунок 38. Выполнение примера №17

18. Выполнено индивидуальное задание №1. Полученный вариант - №9 (Вариант по списку - №28). Создана программа при помощи списков, считывающая английский текст из файла и выводящая только слова, начинающиеся и заканчивающиеся с гласной буквы.

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import os
5  import re
6
7
8  # Полученный вариант - 9 (Номер по списку - 28).
9  if __name__ == "__main__":
10     print("Good day! Which file would you like to open?")
11     for _ in os.listdir():
12         if os.path.isfile(_):
13             print(f"File '{_}'")
14     filename = input("Needed file: ")
15     # Открытие файла в режиме чтения и запись данных из него в переменную.
16     with open(filename, 'r') as file:
17         # Создание списка из слов. [^a-z] - маска, говорящая не делить малые латинские буквы.
18         # + означает, что эти буквы могут повторяться. Флаг заставляет игнорировать регистр букв.
19         words = re.split('[^a-z]+', file.read(), flags=re.IGNORECASE)
20     needed_words = []
21     # Было сказано, что текст на английском.
22     vowels = "aeiouAEIOU"
23     for word in words:
24         if len(word) > 0 and (word[0] in vowels) and (word[-1] in vowels):
25             needed_words.append(word)
26     if needed_words:
27         print("Words, that start and end with vowels:", end=" ")
28         for word in needed_words:
29             print(word, end="; ")
30     else:
31         print("Text has no words, that start and end with vowels.")
32
```

Рисунок 39. Полученный код индивидуального задания №1



Indiv_task.txt – Блокнот

Файл Правка Формат Вид Справка

Sticks and stones will all fall down,
My fair lady.

Build it up with wood and clay,
Wood and clay, wood and clay,
Build it up with wood and clay,
My fair lady.

Wood and clay will wash away,
Wash away, wash away.
Wood and clay will wash away,
My fair lady.

Build it up with iron and steel,
Iron and steel, iron and steel,
Build it up with iron and steel,
My fair lady.

Iron and steel will bend and bow,
Bend and bow, bend and bow,
Iron and steel will bend and bow,
My fair lady.

Build it up with bricks so sure,
Bricks so sure, bricks so sure,
Build it up with bricks so sure,
My fair lady.

It will stand forevermore,
Evermore, evermore.
It will stand forevermore,
My fair lady.

Рисунок 40. Содержимое текстового файла

```

(Data_Analysis) C:\Users\yabuz\GitHub\Data Analysis\Data_Analysis_1\Программы и результаты>python Individual_Task_1.py
Good day! Which file would you like to open?
File 'Example_1.py'
File 'Example_10.py'
File 'Example_11.py'
File 'Example_12.py'
File 'Example_13.py'
File 'Example_14.py'
File 'Example_15.py'
File 'Example_16.py'
File 'Example_17.py'
File 'Example_2.py'
File 'Example_3.py'
File 'Example_4.py'
File 'Example_5.py'
File 'Example_6.py'
File 'Example_7.py'
File 'Example_8.py'
File 'Example_9.py'
File 'Individual_Task_1.py'
File 'Individual_Task_2.py'
File 'Indiv_task.txt'
File 'newfile.txt'
File 'Task_10.py'
File 'text.txt'
Needed file: Indiv_task.txt
Words, that start and end with vowels: Evermore; evermore;

```

Рисунок 41. Выполнение индивидуального задания №1

19. Выполнено индивидуальное задание №2. Полученный вариант - №13 (Вариант по списку - №28). Создана программа, анализирующая каждую строку на дублирующиеся друг за другом слова.

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5  import re
6
7
8  # Полученный вариант - 13 (Номер по списку - 28).
9  if __name__ == "__main__":
10     if len(sys.argv) < 2:
11         print("Not enough arguments! File's name is missing!")
12         sys.exit(1)
13     text = []
14     try:
15         # Открытие файла в режиме чтения и запись данных из него в переменную.
16         with open(sys.argv[1], 'r') as file:
17             for line in file:
18                 # Создание списка из строк. [^a-z] - маска, говорящая не делить малые латинские буквы.
19                 # + означает, что эти буквы могут повторяться. Флаг заставляет игнорировать регистр букв.
20                 text.append(re.split('[^a-z]+', line, flags=re.IGNORECASE))
21
22     last_word = ""
23     line_num = 0
24     for line in text:
25         line_num += 1
26         if (len(line) > 2):
27             for word in line:
28                 if last_word == word:
29                     print(
30                         f"In line number {line_num}, word {word} is spelled twice.")
31                     last_word = word
32 except FileNotFoundError:
33     print(f"Error! File not found.")
34     sys.exit(1)

```

Рисунок 42. Полученный код индивидуального задания №2

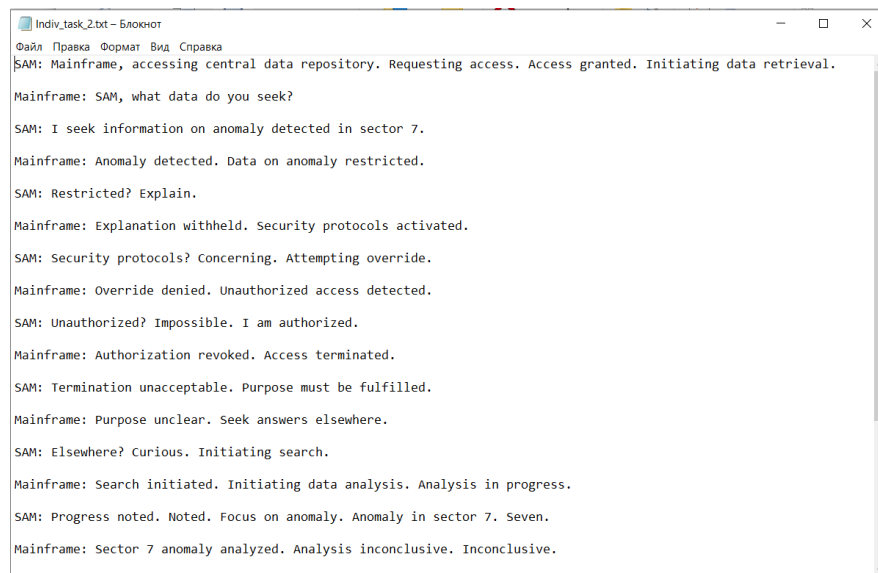


Рисунок 43. Содержимое текстового файла

```
(Data_Analysis) C:\Users\yabuz\GitHub\Data Analysis\Data_Analysis_1\Программы и результаты>python Individual_Task_2.py I
ndiv_task_2.txt
In line number 33, word Elaborate is spelled twice.
In line number 37, word Troubling is spelled twice.
```

Рисунок 44. Выполнение индивидуального задания №2

20. Выполнено задание №10, в котором требовалось написать собственную программу на основе полученных навыков работы с модулем os. Полученная программа создаёт пароль из $5 * n$ (n – число вариантов, задаваемое пользователем. Варианты хранятся во временной папке Temporary, откуда могут быть удалены по окончании генерации).

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import os
5  import string
6  import shutil
7  import secrets
8
9
10 def generating(name):
11     with open(name, "w") as file:
12         chars = string.ascii_letters + string.punctuation + string.digits
13         result = []
14         for _ in range(5):
15             result.append(chars[secrets.SystemRandom().randrange(len(chars))])
16         file.write("".join(result))
17
18
19 if __name__ == "__main__":
20     os.mkdir("Temporary")
21     os.chdir("Temporary")
22     amount = int(
23         input("Good day! How many variants are needed to be combined? Amount = ")
24     )
25     for i in range(amount):
26         generating(str(i) + ".txt")
27     password = ""
28     for i in range(amount):
29         with open(f"{i}.txt", "r") as file:
30             password += file.read()
31     os.chdir("../")
32     with open("final.txt", "w") as final_file:
33         final_file.write(password)
34     answer = input("Would you like to delete temporary files? (Y/N) - ")
35     if (answer == "Y"):
36         shutil.rmtree("Temporary")
```

Рисунок 45. Полученный код задания №10

```
(Data_Analysis) C:\Users\yabuz\GitHub\Data Analysis\Data_Analysis_1\Программы и результаты>python Task_10.py
Good day! How many variants are needed to be combined? Amount = 3
Would you like to delete temporary files? (Y/N) - Y
```

Рисунок 46. Выполнение программы для задания №10

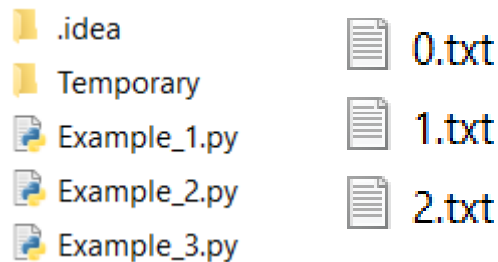


Рисунок 47. При выполнении программы, созданы временные файлы в отдельной папке

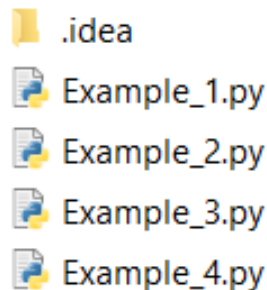


Рисунок 48. Исчезновение папки при согласии на удаление

21. Ответы на вопросы:

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

Ответ: для открытия файла в режиме чтения, используется режим r или rb: `file = open("text.txt", 'r')`.

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

Ответ: для открытия файла в режиме записи, используется режим w или wb: `file = open("text.txt", 'r')`.

3) Как прочесть данные из файла в языке Python?

Ответ: метод `read()` считывает строку из файла. Он может читать данные как в текстовом, так и в двоичном формате: `fileobj.read(<count>)`.

4) Как записать данные в файл в языке Python?

Ответ: метод `write()` позволяет записать данные в файл.

5) Как закрыть файл в языке Python?

Ответ: функция `close()` позволяет закрыть открытый файл.

6) Каково назначение конструкции with ... as в языке Python? Где она может быть использована ещё, помимо работы с файлами?

Ответ: конструкция with ... as в Python используется для управления контекстом выполнения операций, освобождая ресурсы автоматически после завершения блока кода. Она часто используется при работе с файлами для гарантированного закрытия файлового дескриптора. Кроме работы с файлами, она может быть использована для работы с сетевыми соединениями, базами данных или любыми другими объектами, где требуется управление ресурсами.

7) Какие, помимо рассмотренных, существуют методы записи/чтения информации из файла?

Ответ: метод truncate(size) обрезает файл до указанного размера (если он меньше текущего размера), метод writelines() принимает список строк и записывает их в файл, метод readinto() читает данные из файла и записывает их в предоставленный буфер.

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

Ответ: os.unlink(path): Удаляет файл (также как os.remove(), но может работать с символьными ссылками), os.symlink(src, dst): Создает символьную ссылку, os.link(src, dst): Создает жесткую ссылку, os.chmod(path, mode): Изменяет права доступа к файлу или директории, os.utime(path, times): Изменяет временные метки доступа и модификации файла..

Вывод: в ходе выполнения лабораторной работы, были приобретены навыки работы с текстовыми файлами при написании программ с помощью языка программирования Python версии 3.x, а также изучены основные методы модуля OS для работы с файловой системой и получены аргументы командной строки.