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»

Выполнила:
Кубанова Ксения Олеговна
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и вычислительная техника», очная форма обучения

_____
(подпись)

Руководитель практики:
Воронкин Р. А.

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2024 г.

Тема: работа с файлами

Цель: 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.

Порядок выполнения работы

Примеры.

В соответствие с методическими указаниями были реализованы примеры работы с файлами.

```
1   #!/usr/bin/env python3
2   # -*- coding: utf-8 -*-
3
4   #open the file2.txt in append mode. Create a new file
5   with open("file2.txt", "w") as fileptr:
6       #appending the content to the file
7       fileptr.write(
8           "Python is the modern day language. It makes things so simple.\n"
9           "It is the fastest growing programing language."
10      )
```

Рисунок 1 – пример 1

```
file2.txt
1   Python is the modern day language. It makes things so simple.
2   It is the fastest growing programing language.
```

Рисунок 2 – файл примера 1

```
file2.txt
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 user-frendly interaction.
```

Рисунок 3 – результат работы второго примера в файле

prim3.py > ...

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

#open the file2.txt in read mode. causes error if no such file ex
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)
```

Рисунок 4 – пример 3

```
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-frendly
 interaction.
```

Рисунок 5 – результат работы примера 3

prim4.py > ...

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

#open the file2.txt in read mode. causes error if no such file ex
with open("file2.txt", "r") as fileptr:
    content = fileptr.readlines()
    print(content)
```

Рисунок 6 – пример 4

```
['Python is the modern day language. It makes things so simple.\n', 'It is the fastest g
rowing programing language.Python has an easy syntax and user-frendly interaction.']
```

Рисунок 7 – результат примера 4

prim5.py > ...

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

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

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

Рисунок 8 - пример 5

```
<_io.TextIOWrapper name='newfile.txt' mode='x' encoding='cp1251'>
file created successfully
```

Рисунок 9 – результат примера 5

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

if __name__ == "__main__":
    with open("text.txt", "w", encoding = "utf-8") as fileptr:
        print(
            "UTF-8 is a variable-width charecter encoding used fo
            file = fileptr
        )
        print(
            "UTF-8 is capable of encoding all 1,112,064 valid cha
            file = fileptr
        )
        print(
            "in unicode using one to four one-byte (8-bit) code u
            file = fileptr
        )
```

Рисунок 10 – пример 6

```
UTF-8 is a variable-width charecter encoding used for electronic
UTF-8 is capable of encoding all 1,112,064 valid charecter code p
in unicode using one to four one-byte (8-bit) code units.

```

Рисунок 11 – результат примера 6

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

if __name__ == "__main__":
    with open("text.txt", "r", encoding = "utf-8") as fileptr
        sentences = fileptr.readlines()

    for sentence in sentences:
        if "," in sentence:
            print(sentence)
```

Рисунок 12 – пример 7

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

Рисунок 13 – результат примера 7

prim8.py > ...

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  with open("file2.txt", "r") as fileptr:
5      print("the filepointer is at byte:", fileptr.tell())
6
7      fileptr.seek(10);
8
9      print("after reading, the fillpointer is at:", fileptr.tell()
```

Рисунок 14 – пример 8

```
the filepointer is at byte: 0
after reading, the fillpointer is at: 10
```

Рисунок 15 – результат примера 8

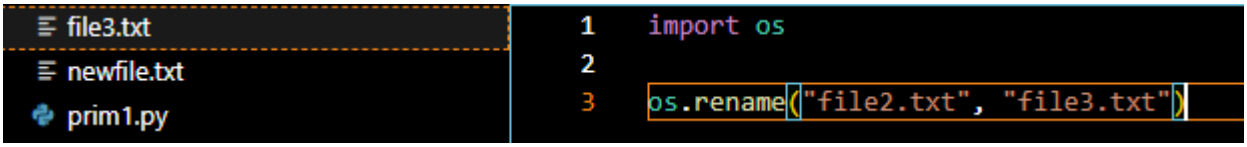

Рисунок 16 – пример 9 и результат

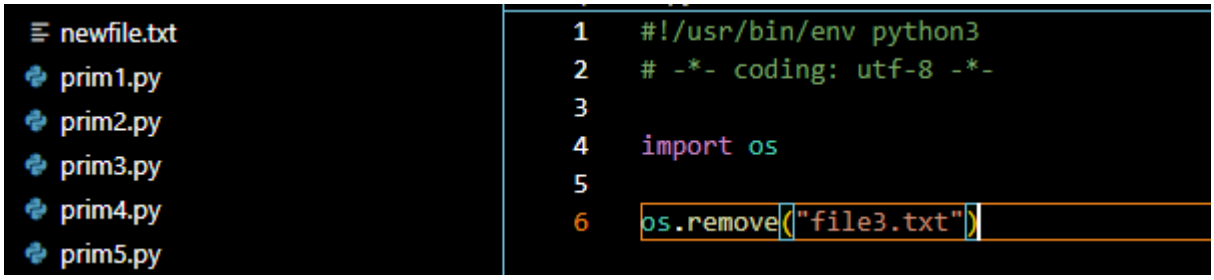

Рисунок 17 – пример 10 и результат

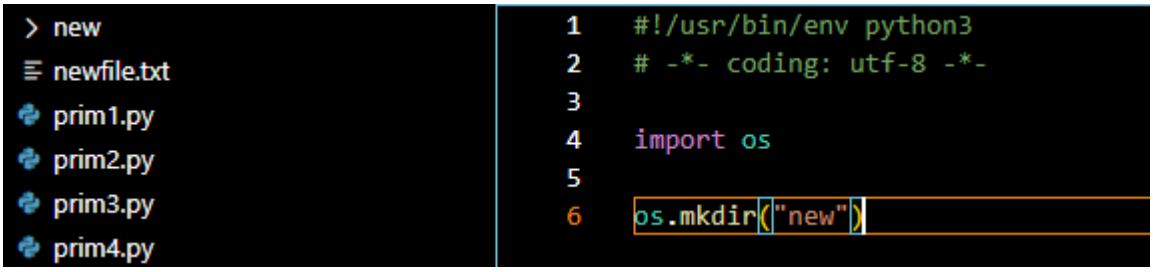

Рисунок 18 – пример 11 и результат

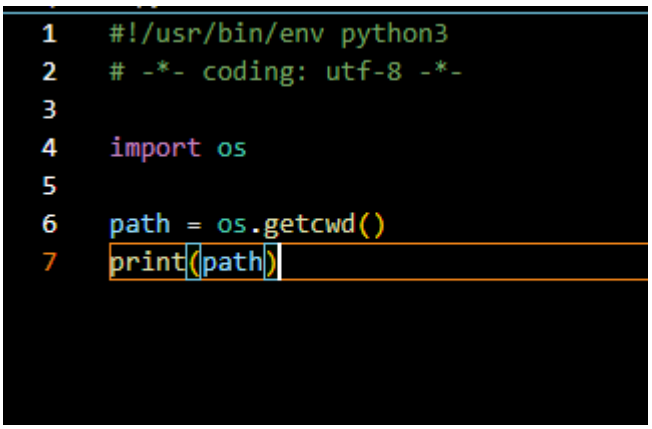

Рисунок 19 – пример 12

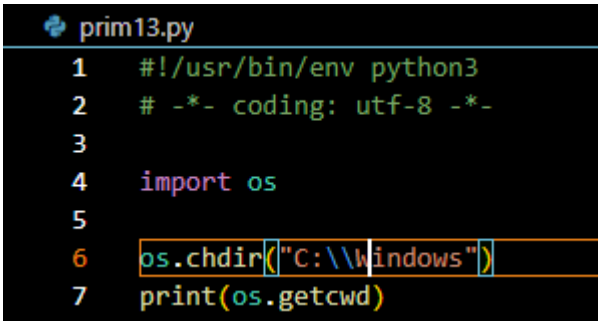

Рисунок 20 – пример 13

Рисунок 21 – результат примера 13

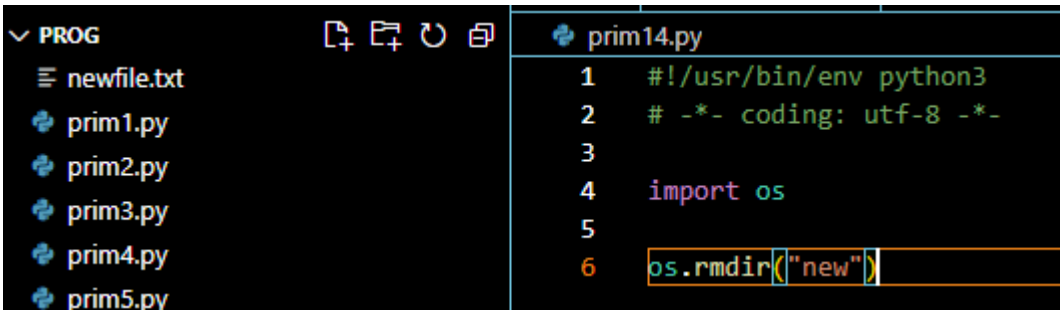

Рисунок 22 – пример 14 и результат

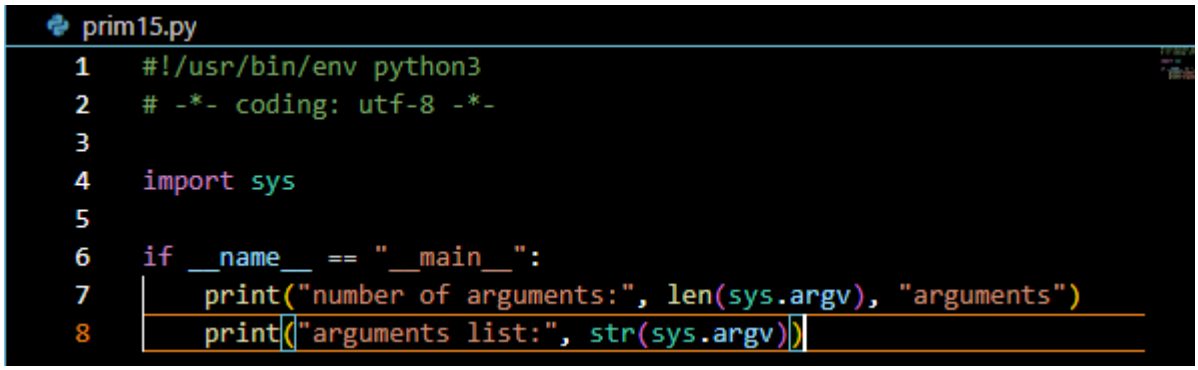

Рисунок 23 – пример 15

```
number of arguments: 1 arguments
arguments list: ['c:/Users/student-09-330/Desktop/AD/1/Lab1_Ad/prog/prim15.py']
```

Рисунок 24 – результат примера 15

prim16.py > ...

```python
#!/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))
```

Рисунок 25 – пример 16

```
no. of arguments passed is  1
```

Рисунок 26 – результат примера 16

prim17.py > ...

```python
#!/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.std
        sys.exit()

    chars = string.ascii_letters + string.punctuation + string.di
    length_pwd = int(sys.argv[1])

    result = []
    for _ in range(length_pwd):
        idx = secrets.SystemRandom().randrange(len(chars))
        result.append(chars[idx])

    print(f"secrets password: {''.join(result)}")
```

Рисунок 27 – пример 17

```
the password length is not given!
```

Рисунок 28 – результат примера 17

Индивидуальное задание 1.

Написать программу, которая считывает текст из файла и выводит на экран все его предложения в обратном порядке.

Код.

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


def read_sentences_from_file(file2):
    with open("file2.txt", "r") as file:
        text = file.read()
        sentences = text.split(".")
        reversed_sentences = list(reversed(sentences))
        return reversed_sentences


def main():
    filename = "file2.txt"
    reversed_sentences = read_sentences_from_file(filename)

    for sentence in reversed_sentences:
        print(sentence + ".")


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

Рисунок 29 – код ИДЗ1

Решение.

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

Рисунок 30 – результат работы ИДЗ1

Индивидуальное задание 2.

Разработайте программу, которая будет показывать слово (или слова), чаще остальных встречающееся в текстовом файле. Сначала пользователь должен ввести имя файла для обработки. После этого вы должны открыть файл и проанализировать его построчно, разделив при этом строки по словам и исключив из них пробелы и знаки препинания. Также при подсчете частоты появления слов в файле вам стоит игнорировать регистры.

Код.

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


def analyze_file(filename):
    wcounts = {}

    with open(filename, "r") as file:
        for line in file:
            words = line.split()
            for word in words:
                word = word.lower()
                if word:
                    if word in wcounts:
                        wcounts[word] += 1
                    else:
                        wcounts[word] = 1

    wmost = max(wcounts, key=wcounts.get)
    f = wcounts[wmost]

    return wmost, f


def main():
    filename = input("Введите имя файла для обработки: ")
    wmost, f = analyze_file(filename)

    print(f"Слово '{wmost}' встречается чаще всего - {f} раз(а).")


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

Рисунок 31 – код ИДЗ2

Решение.

```
Введите имя файла для обработки: file2.txt
Слово 'is' встречается чаще всего - 2 раз(а).
```

Рисунок 32 – результат работы ИДЗ2

Контрольные вопросы

1 Как открыть файл в языке Python только для чтения?

Чтобы открыть файл в языке Python только для чтения, можно использовать функцию open() в режиме чтения 'r'.

2 Как открыть файл в языке Python только для записи?

Чтобы открыть файл только для записи, можно использовать режим 'w' или 'a' в функции open(). Режим 'w' будет перезаписывать файл, а 'a' будет дописывать в конец файла

3 Как прочитать данные из файла в языке Python?
Для чтения данных из файла в Python можно использовать методы read(), readline(), readlines().

4 Как записать данные в файл в языке Python?
Чтобы записать данные в файл в Python, можно использовать метод write()

5 Как закрыть файл в языке Python?
Чтобы закрыть файл в Python после работы с ним, вы можете использовать метод close(), но в рекомендуется использовать блок with, который автоматически закроет файл по завершении работы в нем.

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 управления контекстом выполнения, гарантируя правильное открытие и закрытие ресурсов. Кроме работы с файлами, with может быть использована для работы с сетевыми соединениями, базами данных и другими ресурсами.

7 Изучите самостоятельно документацию Python по работе с файлами. Какие помимо рассмотренных существуют методы записи/чтения информации из файла?
Помимо уже рассмотренных методов, существуют также методы для работы с файлами в Python, такие как flush(), seek(), tell() и другие. У каждого метода свои особенности и сферы применения.

8 Какие существуют, помимо рассмотренных, функции модуля os для работы с файловой системой?
Помимо рассмотренных функций модуля os для работы с файловой системой, существуют другие полезные функции, такие как os.rename(), os.remove(), os.mkdir() для переименования файлов, удаления файлов и создания директорий соответственно.

Вывод: были приобретены навыки по работе с текстовыми файлами при написании программ с помощью языка программирования Python версии 3.x,

изучены основные методы модуля os для работы с файловой системой, получение аргументов командной строки.