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 ПРАКТИЧЕСКОЙ РАБОТЕ №1
дисциплины «Анализ данных»
Вариант №2

Выполнила:
Беседина Инга 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., канд. технических наук, доцент, доцент кафедры инфокоммуникаций

_____
(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 основных методов модуля os для работы с файловой системой, получение аргументов командной строки.

Ход работы

Пример 1. Запись текста в файл.

Запись текста в файл без использования оператора 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.
   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 programing language"
   )
   # closing the opened the file
   fileptr.close()
```

Запись текста в файл с использованием оператора 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"
        )
```

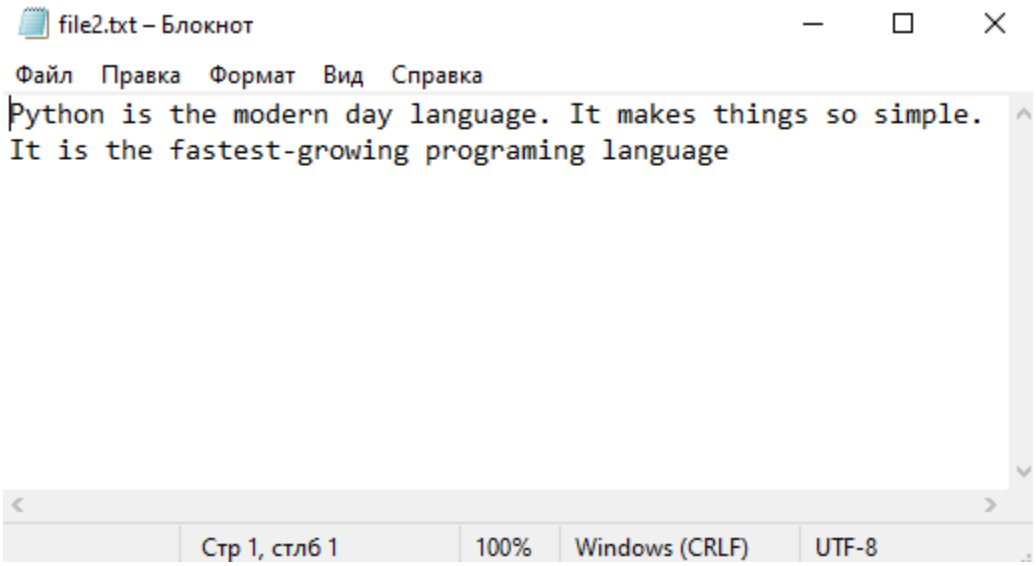

Рисунок 1. Содержимое файла file2.txt

Пример 2. Добавление содержимого в существующий файл.

Без использования оператора with:

```
#!/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()
```

С использованием оператора 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.")
```

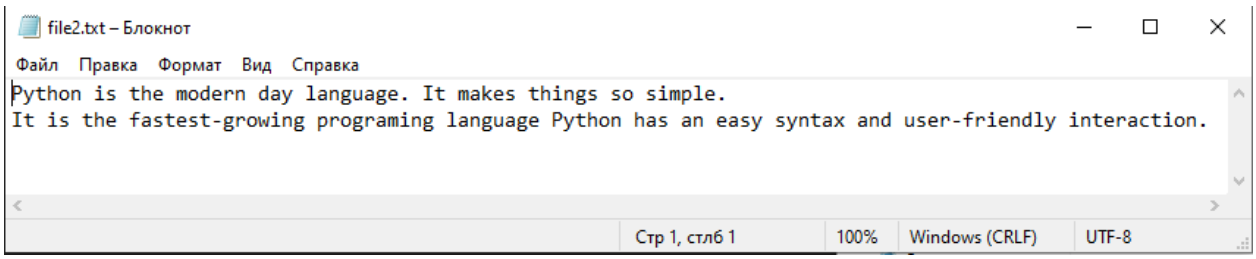

Рисунок 2. Содержимое файла file2.txt

Пример 3. Чтение из файла.

Без использования оператора 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.
    fileptr = open("file2.txt", "r")
    # stores all the data of the file into the variable content
    content = fileptr.read(10)
    # prints the type of the data stored in the file
    print(type(content))
    # prints the content of the file
    print(content)
    # closes the opened file
    fileptr.close()
```

С использованием оператора 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.read(10)
        # prints the type of the data stored in the file
        print(type(content))
        # prints the content of the file
        print(content)
```

```
<class 'str'>
Python is
```

Рисунок 3. Результаты работы программы

Пример 4. Чтение строк с помощью метода readline().

Без использования оператора 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.
    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()
```

С использованием оператора 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.
```

Рисунок 4. Результаты работы программы

Пример 5. Чтение строк с помощью функции readlines()

Без использования оператора with:

```
#!/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()
```

С использованием оператора 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)
```

```
['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 5. Результаты работы программы

Пример 6. Создание нового файла.

Без использования оператора 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.
    fileptr = open("newfile.txt", "x")
    print(fileptr)

    if fileptr:
        print("File created successfully")

    # closes the opened file
    fileptr.close()
```

С использованием оператора 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")
```

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

Рисунок 6. Результаты работы программы

Пример 7. Изменение кодировки файла.

```
#!/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
        )
```

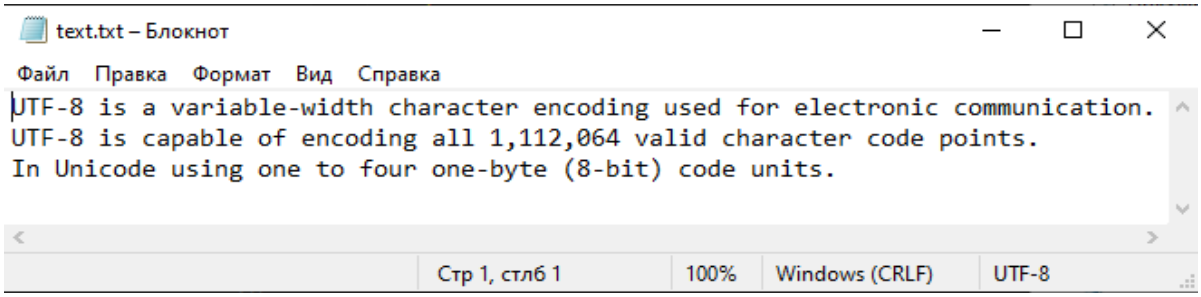

Рисунок 7. Содержимое файла text.txt

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.*

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


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

Пример 9. Позиция указателя файла

```
#!/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 9. Результаты работы программы

Пример 10. Модуль os. Переименование файла

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


import os


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

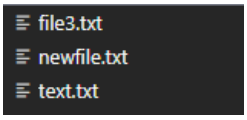

Рисунок 10. Результаты работы программы

Пример 11. Модуль os. Удаление файла

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


import os


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

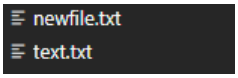

Рисунок 11. Результаты работы программы

Пример 12. Модуль os. Создание нового каталога

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


import os


if __name__ == "__main__":
    #creating a new directory with the name new
    os.mkdir("new")
```

Рисунок 12. Созданный каталог new

Пример 13. Модуль os. Получение текущего рабочего каталога

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


import os


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

```
D:\Rep\Data_analysis_1\Project
```

Рисунок 13. Текущий рабочий каталог

Пример 14. Модуль os. Изменение текущего рабочего каталога

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


import os


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

```
C:\Windows
```

Рисунок 14. Текущий рабочий каталог

Пример 15. Модуль os. Удаление каталога

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


import os


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

Пример 16. Доступ к элементам командной строки

```
#!/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))
```

```
Number of arguments: 4 arguments
Argument List: ['example16.py', 'arg1', 'arg2', 'arg3']
```

Рисунок 15. Результаты работы программы

Пример 17. Доступ к элементам командной строки

```
#!/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))
```

```
Argument #0 is example17.py
Argument #1 is Knowledge
Argument #2 is Hut
Argument #3 is 21
No. of arguments passed is  4
```

Рисунок 16. Результаты работы программы

Пример 17. *Написать программу для генерации пароля заданной длины. Длина пароля должна передаваться как аргумент командной строки сценария.*

```
#!/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)}")
```

```
Secret Password: ^F9.pDv:AAN5
```

Рисунок 17. Результаты работы программы

Индивидуальное задание:

2. Написать программу, которая считывает текст из файла и выводит на экран только предложения, содержащие введенное с клавиатуры слово.

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


if __name__ == "__main__":
    word = input()

    with open("text.txt", "r") as text:
        sentences = text.readlines()

    for sentence in sentences:
        if word in sentence:
            print(sentence)
```

```
units
In Unicode using one to four one-byte (8-bit) code units.
```

Рисунок 18. Результаты работы программы

2. Продолжая тему предыдущего упражнения, в тех же операционных системах на базе Unix обычно есть и утилита с названием tail, которая отображает последние десять строк содержимого файла, имя которого передается в качестве аргумента командной строки. Реализуйте программу, которая будет делать то же самое. В случае отсутствия файла, указанного пользователем, или аргумента командной строки вам нужно вывести соответствующее сообщение.

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

```python
import sys


def tail(filename):
    try:
        with open(filename, "r", encoding="utf-8") as file:
            sentences = file.readlines()
            if len(sentences) > 10:
                for sentence in sentences[-10:]:
                    print(sentence, end="")
            else:
                for sentence in sentences:
                    print(sentence, end="")
    except FileNotFoundError:
        print(f"File {filename} not found")


def main():
    if len(sys.argv) != 2:
        print("Использование: python indv2.py <имя_файла>")
    else:
        tail(sys.argv[1])


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

```
PS D:\Rep\Data_analysis_1\Project> python indv2.py indv2.txt
Строка 16
Строка 17
Строка 18
Строка 19
Строка 20
Строка 21
Строка 22
Строка 23
Строка 24
Строка 25
```

Рисунок 19. Результаты работы программы

```
PS D:\Rep\Data_analysis_1\Project> python indv2.py new.txt
File new.txt not found
```

Рисунок 20. Результаты работы программы

Задание 3. Найти все файлы с расширением .txt в указанной директории и ее поддиректориях, а затем вывести список найденных файлов.

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

```python
import os


def find_files(directory):
    txt_files = []
    for root, dirs, files in os.walk(directory):
        for file in files:
            if file.endswith(".txt"):
                txt_files.append(os.path.join(root, file))
    return txt_files


def main():
    directory = input("Введите путь к директории: ")
    if os.path.isdir(directory):
        txt_files = find_files(directory)
        if txt_files:
            print("Найдены следующие файлы с расширением .txt:")
            for file in txt_files:
                print(file)
        else:
            print("В указанной директории и ее поддиректориях не найдено файлов с
расширением .txt")
    else:
        print("Указанная директория не существует")


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

```
Введите путь к директории: D:\Rep\Data_analysis_1\Project\files
Найдены следующие файлы с расширением .txt:
D:\Rep\Data_analysis_1\Project\files\indv2.txt
D:\Rep\Data_analysis_1\Project\files\newfile.txt
D:\Rep\Data_analysis_1\Project\files\text.txt
```

Рисунок 21. Результаты работы программы

Контрольные вопросы:

1. Установить режим доступа r, который открывает файл в режиме только для чтения

2. Установить режим доступа w – только для записи

3. Для чтения данных из файла в Python используется функция open(), которая открывает файл и возвращает объект файла. Далее можно

использовать методы этого объекта, такие как read(), readline(), readlines() для чтения данных из файл

4. Для записи данных в файл в Python также используется функция open(), но с указанием режима записи (например, "w" для записи или "a" для добавления в конец файла). Затем можно использовать методы объекта файла, такие как write() для записи данных

5. Файл следует закрывать после завершения работы с ним, для этого используется метод close() объекта файл

6. Конструкция with ... as в Python используется для автоматического управления ресурсами, например, для открытия файлов, работы с сетевыми соединениями и т.д. Она гарантирует, что ресурсы будут корректно освобождены после завершения блока with. Кроме работы с файлами, она может быть использована, например, при работе с базами данных

7. Помимо уже рассмотренных методов чтения/записи информации из файла, существуют такие методы как fileno() - возвращает целочисленный файловый дескриптор файла; isatty() - возвращает True, если файловый объект связан с терминалом; writable() - возвращает True, если файл открыт для записи; writelines(lines) - записывает список строк в файл

8. Модуль os в Python предоставляет множество функций для работы с файловой системой. Некоторые из них: listdir(path) - возвращает список файлов и подкаталогов в указанном каталоге; walk(top, topdown=True, onerror=None, followlinks=False) - генератор, позволяющий рекурсивно обходить дерево каталогов, возвращая для каждого каталога кортеж (dirpath, dirnames, filenames); stat(path) - возвращает информацию о файле или каталоге

Вывод: в ходе выполнения лабораторной работы были приобретены навыки по работе с текстовыми файлами при написании программ с помощью языка программирования Python версии 3.x, были изучены основные методы модуля os для работы с файловой системой, получение аргументов командной строки.