

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 №5
дисциплины «Языки программирования»

Выполнила:
Иващенко Олеся Игорьевна
2 курс, группа ИТС-б-о-21-1,
11.03.02 «Инфокоммуникационные
технологии и системы связи»,
направленность (профиль)
«Инфокоммуникационные системы и
сети», очная форма обучения

(подпись)

Руководитель практики:
Воронкин Р. А., доцент кафедры
инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2022 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:

1. Оператор with. Запись файла.

Чтобы записать текст в файл, нам нужно открыть файл с помощью метода open с одним из следующих режимов доступа.

1. 'w': он перезапишет файл, если какой-либо файл существует. Указатель файла находится в начале файла.

2. 'a': добавит существующий файл. Указатель файла находится в конце файла. Он создает новый файл, если файл не существует.

Чтобы прочитать файл с помощью сценария Python, Python предоставляет метод read(). Метод read() считывает строку из файла. Он может читать данные как в текстовом, так и в двоичном формате.

Мы можем прочитать файл, используя цикл for.

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  # open the file1.txt in append mode. Create a new file if no such file exists.
5  with open("file1.txt", "w") as fileptr:
6
7      # appending the content to the file
8      fileptr.write(
9          "Python is the modern day language. It makes things so simple.\n"
10         "It is the fastest-growing programming language"
11     )
```

Рисунок 1. Пример 1 с использованием "w"

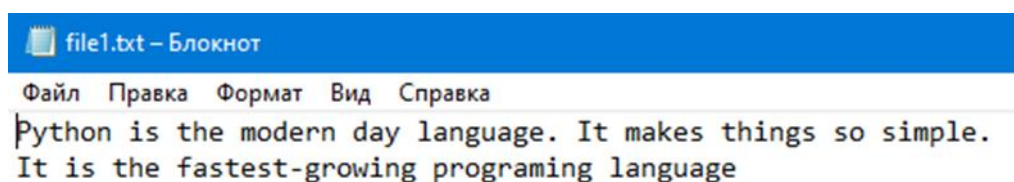


Рисунок 2. Результат работы

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  # open the file1.txt in write mode.
5  with open("file1.txt", "a") as fileptr:
6      # overwriting the content of the file
7      fileptr.write(" Python has an easy syntax and user-friendly interaction.")
8      # open the file1.txt in read mode. causes error if no such file exists.
9  with open("file1.txt", "r") as fileptr:
10     # stores all the data of the file into the variable content
11     content = fileptr.read(14)
12     # prints the type of the data stored in the file
13     print(type(content))
14     # prints the content of the file
15     print(content)
16     # open the file2.txt in read mode. causes error if no such file exists.
17 with open("file1.txt", "r") as fileptr:
18     # running a for loop
19     for i in fileptr:
20         print(i) # i contains each line of the file

```

Рисунок 3. Пример 1 с использованием “a”, “r” и построчным чтением содержимого файла

```

<class 'str'>
Python is the
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-friendl
Process finished with exit code 0

```

Рисунок 4. Результат выполнения

file1.txt – Блокнот

Файл Правка Формат Вид Справка

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 5. Результат работы

2. Построчное чтение содержимого файла с помощью методов файлового объекта.

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  # open the file1.txt in read mode. causes error if no such file exists.
5  with open("file1.txt", "r") as fileptr:
6      # stores all the data of the file into the variable content
7      content1 = fileptr.readline(14)
8      content2 = fileptr.readline(47)
9
10 # prints the content of the file
11 print(content1)
12 print(content2)

```

Рисунок 6. Код (Пример 3)

```

Python is the
modern day language. It makes things so simple.

Process finished with exit code 0

```

Рисунок 7. Результат выполнения (Пример 3)

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  # open the file1.txt in read mode. causes error if no such file exists.
5  with open("file1.txt", "r") as fileptr:
6      # stores all the data of the file into the variable content
7      content = fileptr.readlines()
8
9  # prints the content of the file
10 print(content)

```

Рисунок 8. Код (Пример 4)

```

['Python is the modern day language. It makes things so simple.\n', 'It is the fastest-

Process finished with exit code 0

```

Рисунок 9. Результат выполнения (Пример 4)

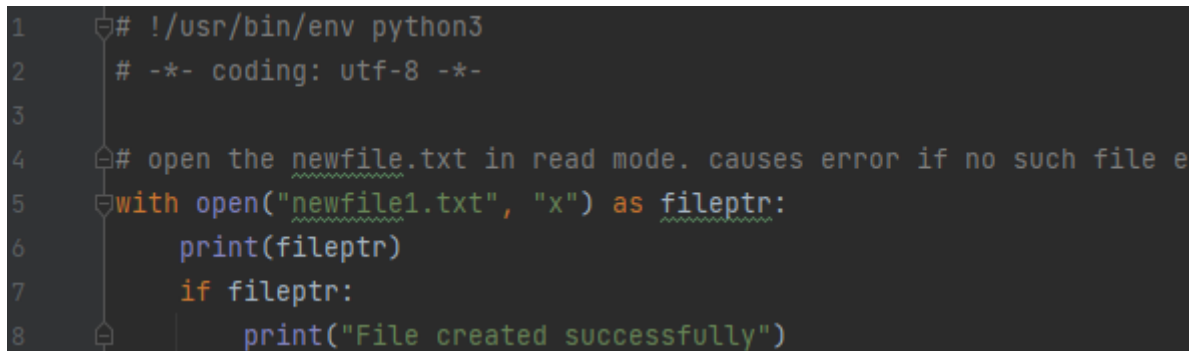
3. Создание нового файла.

Новый файл можно создать, используя один из следующих режимов доступа с функцией `open()`.

1. `x`: создает новый файл с указанным именем. Вызовет ошибку, если существует файл с таким же именем.
2. `a`: создает новый файл с указанным именем, если такого файла не существует. Он добавляет содержимое к файлу, если файл с указанным

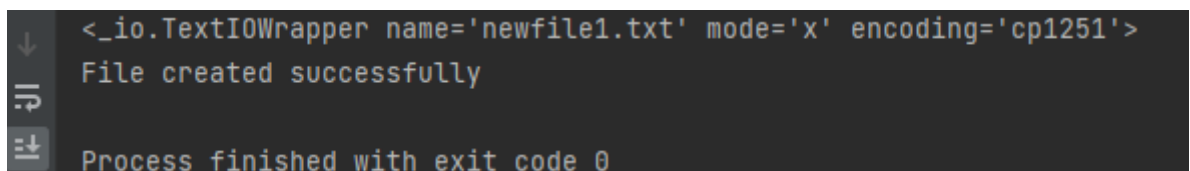
именем уже существует.

3. `w`: создает новый файл с указанным именем, если такого файла не существует. Он перезаписывает существующий файл.

A screenshot of a code editor showing a Python script. The script is as follows:

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 # open the newfile.txt in read mode. causes error if no such file e
5 with open("newfile1.txt", "x") as fileptr:
6     print(fileptr)
7     if fileptr:
8         print("File created successfully")
```

Рисунок 10. Код (Пример 5)

A screenshot of a terminal window showing the execution of the Python script. The output is as follows:

```
<_io.TextIOWrapper name='newfile1.txt' mode='x' encoding='cp1251'>
File created successfully
Process finished with exit code 0
```

Рисунок 11. Результат выполнения (Пример 5)

4. Изменение кодировки файла.

При работе с несколькими операционными системами, а также при передаче данных по сети одной из наиболее распространенных кодировок является кодировка UTF-8. Следующий пример показывает, как осуществить запись данных в файл с использованием кодировки UTF-8.

```

▶ #!/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 12. Код (Пример 6)

text.txt – Блокнот

Файл Правка Формат Вид Справка

UTF-8 is a variable-width character encoding used for electronic communication.
 UTF-8 is capable of encoding all 1,112,064 valid character code points.
 In Unicode using one to four one-byte (8-bit) code units.

Рисунок 13. Результат работы (Пример 6)

Написать программу, которая считывает текст из файла и выводит на экран только предложения, содержащие запятые. Каждое предложение в файле записано на отдельной строке.

```

1 ▶ #!/usr/bin/env python3
2   # -*- coding: utf-8 -*-
3 ▶ if __name__ == "__main__":
4       with open("text.txt", "r", encoding="utf-8") as fileptr:
5           sentences = fileptr.readlines()
6           # Вывод предложений с запятыми.
7       for sentence in sentences:
8           if "," in sentence:
9               print(sentence)

```

Рисунок 14. Код (Пример 7)

```
UTF-8 is capable of encoding all 1,112,064 valid character code points.  
Process finished with exit code 0
```

Рисунок 15. Результат выполнения (Пример 7)

5. Позиция указателя файла.

Python предоставляет метод `tell()`, который используется для печати номера байта, в котором в настоящее время существует указатель файла.

В реальных приложениях иногда нам нужно изменить расположение указателя файла извне, поскольку нам может потребоваться прочитать или записать контент в разных местах. Для этой цели Python предоставляет нам метод `seek()`, который позволяет нам изменять положение указателя файла извне.

Метод `seek()` принимает два параметра:

1. `offset` – относится к новой позиции указателя файла в файле.
2. `from` – указывает ссылочную позицию, из которой должны быть перемещены байты. Если он установлен на 0, начало файла используется в качестве позиции ссылки. Если он установлен на 1, текущая позиция файлового указателя используется как ссылочная. Если установлено значение 2, конец указателя файла используется в качестве позиции ссылки.

```
1 ▶ #!/usr/bin/env python3  
2   # -*- coding: utf-8 -*-  
3  
4   # open the file file1.txt in read mode  
5   with open("file1.txt", "r") as fileptr:  
6       # initially the filepointer is at 0  
7       print("The filepointer is at byte :", fileptr.tell())  
8  
9       # changing the file pointer location to 10.  
10      fileptr.seek(10)  
11  
12      # tell() returns the location of the fileptr.  
13      print("After reading, the filepointer is at:", fileptr.tell())
```

Рисунок 16. Код (Пример 8)


```
The filepointer is at byte : 0
After reading, the filepointer is at: 10
Process finished with exit code 0
```

Рисунок 17. Результат выполнения (Пример 8)

6. Модуль os. Переименование файла.

Модуль Python os обеспечивает взаимодействие с операционной системой. Модуль os предоставляет функции, которые участвуют в операциях обработки файлов, таких как переименование, удаление и т.д. Он предоставляет нам функцию `rename()` для переименования указанного файла в новое имя.

```
1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3
4      import os
5
6      # rename file2.txt to file3.txt
7      os.rename("file2.txt", "file3.txt")
```

Рисунок 18. Код (Пример 9)

7. Удаление файла.

Модуль os предоставляет функцию `remove()`, который используется для удаления указанного файла.

```
1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3
4      import os
5
6      # deleting the file named file3.txt
7      os.remove("file3.txt")
```

Рисунок 19. Код (Пример 10)

8. Создание нового каталога.

Функция `mkdir()` используется для создания каталогов в текущем рабочем каталоге.


```

1 ▶ 1 #!/usr/bin/env python3
2   2 # -*- coding: utf-8 -*-
3
4   3 import os
5
6   4 # creating a new directory with the name new
7   5 os.mkdir("new")

```

Рисунок 20. Код (Пример 11)

9. Получение текущего рабочего каталога.

Эта функция возвращает текущий рабочий каталог.

```

1 ▶ 1 #!/usr/bin/env python3
2   2 # -*- coding: utf-8 -*-
3
4   3 import os
5
6   4 path = os.getcwd()
7   5 print(path)

```

Рисунок 21. Код (Пример 12)

10. Изменение текущего рабочего каталога.

Функция `chdir()` используется для изменения текущего рабочего каталога на указанный каталог.

```

1 ▶ 1 #!/usr/bin/env python3
2   2 # -*- coding: utf-8 -*-
3
4   3 import os
5
6   4 # Changing current directory with the new directory
7   5 os.chdir("C:\\MonstR")
8   6 # It will display the current working directory
9   7 print(os.getcwd())

```

Рисунок 22. Код (Пример 13)

11. Удаление каталога.

Функция `rmdir()` используется для удаления указанного каталога.

```

1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3
4      import os
5
6      # removing the new directory
7      os.rmdir("new")

```

Рисунок 23. Код (Пример 14)

12. Доступ к элементам командной строки в языке программирования Python.

Для доступа к элементам командной строки используется список `sys.argv`, который содержит все аргументы командной строки, переданные программе на языке Python. Это важно при работе с аргументами командной строки. Рассмотрим подробнее на нескольких примерах. С помощью функции `len(sys.argv)` Вы можете подсчитать количество аргументов.

```

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))

```

Рисунок 24. Код (Пример 15)

```

Number of arguments: 4 arguments
Argument List: ['G:/Языки/5/lb5prim15.py', 'arg1', 'arg2', 'arg3']

```

Рисунок 25. Результат выполнения (Пример 15)

При этом следует обратить внимание, что `sys.argv[0]` содержит имя файла программы на языке Python. Следующие элементы списка `sys.argv` содержат переданные программе параметры командной строки.

```

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

```

Рисунок 26. Код (Пример 16)

```

Argument #0 is G:/Языки/5/lb5prim16.py
Argument #1 is Knowledge
Argument #2 is Hut
Argument #3 is 21
No. of arguments passed is 4

```

Рисунок 27. Результат выполнения (Пример 16)

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

Решение. Перед написанием программы необходимо учесть тот факт, что все элементы списка `sys.argv` являются строками, поэтому необходимо преобразовать значение `sys.argv[1]` к типу `int`. Напишем программу для решения поставленной задачи.

```

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

```

Рисунок 28. Код (Пример 17)

```
Secret Password: K*[ZEMb"]X,c=m
```

Рисунок 29. Результат выполнения (Пример 18)

Индивидуальное задание

Вариант №17

Задание 1

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

Рисунок 30. Условия задания

```

1 ▶ 1 #!/usr/bin/env python3
2   2 # -*- coding: utf-8 -*-
3
4 ▶ 3 if __name__ == "__main__":
5   4     with open('text.txt', 'r', encoding='utf-8-sig') as fileptr:
6   5         sentence = fileptr.readlines()
7   6         reverse_sentence = list(reversed(sentence))
8   7         print(reverse_sentence)

```

Рисунок 31. Код задания 1

```
['In Unicode using one to four one-byte (8-bit) code units.\n', 'UTF-8 is capable of encoding all 1,112,064 valid character code points.\n', 'UTF-8 is a variable-width character\n']\nProcess finished with exit code 0
```

Рисунок 32. Результат выполнения задания 1

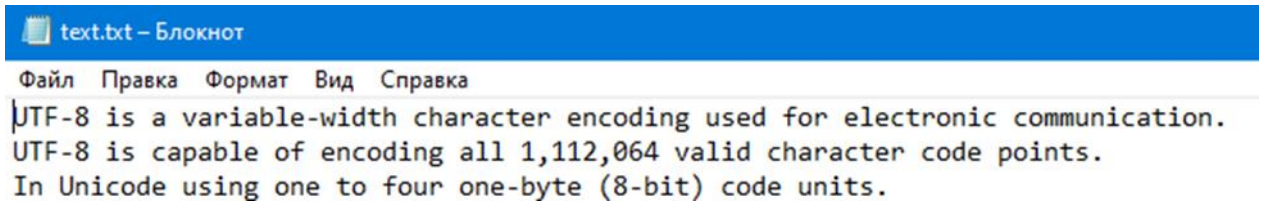


Рисунок 33. Файл для открытия

Задание 2

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.

Рисунок 34. Условия задания

```
1  ▶  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4      import sys
5  ▶  if __name__ == "__main__":
6
7          if len(sys.argv) < 10:
8              print("Недостаточно строк", file=sys.stderr)
9
10         else:
11             print("Последние 10 строк", str(sys.argv[-10::]))
```

Рисунок 35. Код для задания 2

```
C:\Users\MonstR>python G:/Языки/5/iz2.py 14 1 a1 a d f w e fwef wqw qwf weg qw q u o p [ l m j h f t h k
Последние 10 строк ['p', '[', 'l', 'm', 'j', 'h', 'f', 't', 'h', 'k']
```

Рисунок 36. Результат выполнения задания 2

Ответы на контрольные вопросы:

1. Как открыть файл в языке Python только для чтения?
 - `fileptr = open("Имяфайла", "r")`
 - `with open("file1.txt", "r") as fileptr:`
2. Как открыть файл в языке Python только для записи?
 - `fileptr = open("Имяфайла", "w")`
 - `with open("file1.txt", "w") as fileptr:`
3. Как прочитать данные из файла в языке Python?

```
# 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()
```

4. Как записать данные в файл в языке Python?

```
# 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()
```

5. Как закрыть файл в языке Python?

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

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` используется для оборачивания выполнения блока инструкций менеджером контекста. Иногда это более удобная

конструкция, чем try...except...finally. Самый распространённый пример использования этой конструкции - открытие файлов. `with ... as`, как правило, является более удобной и гарантирует закрытие файла в любом случае.

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

- `r` – открывает файл в режиме только для чтения. Указатель файла существует в начале. Файл по умолчанию открывается в этом режиме, если не передан режим доступа.
- `rb` – открывает файл в двоичном формате только для чтения. Указатель файла существует в начале файла.
- `r+` – открывает для чтения и записи. Указатель файла также существует в начале.
- `rb+` – открывает в двоичном формате. Указатель файла присутствует в начале файла.
- `w` – только для записи. Он перезаписывает файл, если он существовал ранее, или создает новый, если файл с таким именем не существует. Указатель имеется в начале файла.
- `wb` – открывает файл для записи только в двоичном формате. Перезаписывает файл, если он существует ранее, или создает новый, если файл не существует. Указатель файла существует в начале файла.
- `w+` – для записи и чтения обоих. Он отличается от `r+` в том смысле, что он перезаписывает предыдущий файл, если он существует, тогда как `r+` не перезаписывает ранее записанный файл. Он создает новый файл, если файл не существует. Указатель файла существует в начале файла.
- `wb+` – открывает файл для записи и чтения в двоичном формате. Указатель файла существует в начале файла.
- `a` – открывает файл в режиме добавления. Указатель файла существует в конце ранее записанного файла, если он существует. Он создает новый файл, если не существует файла с таким же именем.
- `ab` – открывает файл в режиме добавления в двоичном формате. Указатель существует в конце ранее записанного файла. Он создает новый файл в двоичном формате, если не существует файла с таким же именем.
- `a+` – открывает файл для добавления и чтения. Указатель файла остается в конце файла, если файл существует. Он создает новый файл, если не существует файла с таким же именем.
- `ab+` – открывает файл для добавления и чтения в двоичном формате. Указатель файла остается в конце файла.

Вывод: в ходе лабораторной работы 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.