

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций

Отчет по лабораторной работе № 17 (2.15)

Работа с файлами в языке Python.

по дисциплине «Технологии программирования и алгоритмизации»

Выполнил студент группы ИВТ-б-о-20-1

Злыгостев И.С. « » _____ 20__ г.

Подпись студента _____

Работа защищена « » _____ 20__ г.

Проверил Воронкин Р.А. _____

(подпись)

Ставрополь 2021

Цель работы: 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. Первым делом, после ознакомления с методическими указаниями, перешёл к разбору примером.

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

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 programming language!")
```

Рисунок 17.1 – Запись файла

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

# 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.")

# 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)

# open the file2.txt in read mode. causes error if no such file exists.
with open("file2.txt", "r") as fileptr:
    # running a for loop
    for i in fileptr:
        print(i) # i contains each line of the file
```

Рисунок 17.2 – Чтение файла

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

# 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)
```

Рисунок 17.3 – Чтение строк при помощи readline()

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

# 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)
```

Рисунок 17.4 – Чтение строк с помощью функции readlines()

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

# 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")
```

Рисунок 17.5 – Создание нового 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 17.6 – Изменение кодировки файла

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

if __name__ == "__main__":
    with open("text.txt", "r", encoding="utf-8") as f:
        sentences = f.readlines()
        # Вывод предложений с запятыми.
        for sentence in sentences:
            if "," in sentence:
                print(sentence)
```

Рисунок 17.7 – Вывод предложений, содержащих запяты

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

# 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())
```

Рисунок 17.8 – Позиция указателя файла

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

import os

# rename file2.txt to file3.txt
os.rename("file2.txt", "file3.txt")
```

Рисунок 17.9 – Переименование файла

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

import os

# deleting the file named file3.txt
os.remove("file3.txt")
```

Рисунок 17.10 – Удаление файла

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

import os

#creating a new directory with the name new
os.mkdir("new")
```

Рисунок 17.11 – Создание новой директории

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

import os

path = os.getcwd()
print(path)
```

Рисунок 17.12 – Получение пути текущего рабочего каталога

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

import os

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

Рисунок 17.13 – Изменение текущего рабочего каталога

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

import os

# removing the new directory
os.rmdir("new")
```

Рисунок 17.14 – Удаление каталога

```
#!/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))
```

Рисунок 17.15 – Подсчёт аргументов 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))
```

Рисунок 17.16 – Иной способ подсчёта аргументов командной строки

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

import ...

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)}")
```

Рисунок 17.17 – Программа по созданию пароля

2. После разбора примеров, приступил к выполнению индивидуальных заданий.

Вариант 5

Задание 1.

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

"""
Написать программу, которая считывает текст из файла и выводит его на экран, меняя
местами каждые два соседних слова.
"""

if __name__ == "__main__":
    with open("ind1.txt", "r", encoding='utf-8') as f:
        file = (f.read()).split(' ')
        print('Содержимое файла:', ' '.join(file))
        try:
            for i, n in enumerate(file):
                if i % 2 == 0:
                    file[i], file[i + 1] = file[i + 1], file[i]
            print("После выполнения перестановки: ", ' '.join(file))
        except IndexError:
            print("После выполнения перестановки: ", ' '.join(file))
```

Рисунок 17.18 – Код первого задания

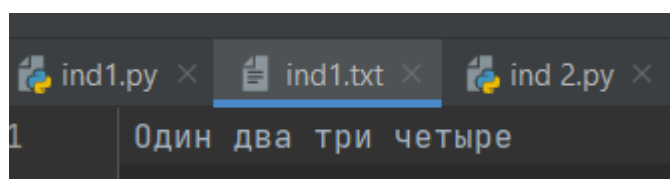


Рисунок 17.19 – Содержимое текстового файла к заданию

```
Содержимое файла: Один два три четыре
После выполнения перестановки: два Один четыре три
```

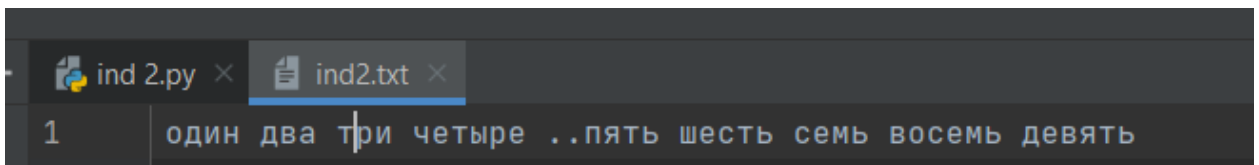
Рисунок 17.20 – Результат выполнения

Задание 2

```
"""
В данном упражнении вы должны написать программу, которая будет находить самое
длинное слово в файле. В качестве результата программа
должна выводить на экран длину самого длинного слова и все слова такой длины. Для
простоты принимайте за значимые буквы любые непробельные символы, включая цифры и
знаки препинания.
"""

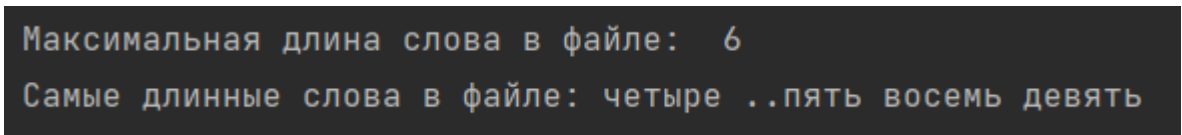
if __name__ == "__main__":
    with open("ind2.txt", "r", encoding='utf-8') as f:
        words = []
        count = 0
        file = (f.read()).split(' ')
        for w in file:
            length = len(w)
            if length > count:
                count = length
        for w in file:
            if len(w) == count:
                words.append(w)
        print('Максимальная длина слова в файле: ', count)
        print('Самые длинные слова в файле:', " ".join(words))
```

Рисунок 17.21 – Код второго индивидуального задания



```
ind 2.py x ind2.txt x
1  один два три четыре ..пять шесть семь восемь девять
```

Рисунок 17.22 – Содержимое файла к заданию



```
Максимальная длина слова в файле:  6
Самые длинные слова в файле: четыре ..пять восемь девять
```

Рисунок 17.23 – Результат выполнения кода

Задание 3

Напишите программу, которая будет выполнять действия над файлами/каталогами в виде: “команда” “наименование файла/каталога”.

```
ind with os.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import os
5
6
7  if __name__ == "__main__":
8      while True:
9          message = input("Введите команду: ").lower()
10         task = message.split(' ')
11         if message.startswith("mkdir "):
12             os.mkdir(task[1])
13         elif message.startswith("rmdir "):
14             os.rmdir(task[1])
15         elif message.startswith("create "):
16             open(task[1], 'w')
17         elif message.startswith("remove "):
18             os.remove(task[1])
19         elif message.startswith("rename "):
20             os.rename(task[1], task[2])
21         elif message.startswith("exit"):
22             exit()
23         elif message.startswith("open "):
24             os.system(f"notepad.exe {task[1]}")
25         else:
26             print("Неизвестная команда")
27
```

Рисунок 17.24 – Код третьего индивидуального задания

```
Введите команду: mkdir test
Введите команду: create test.py
Введите команду: create test.txt
Введите команду: exit
```

Рисунок 17.25 – Ввод команд

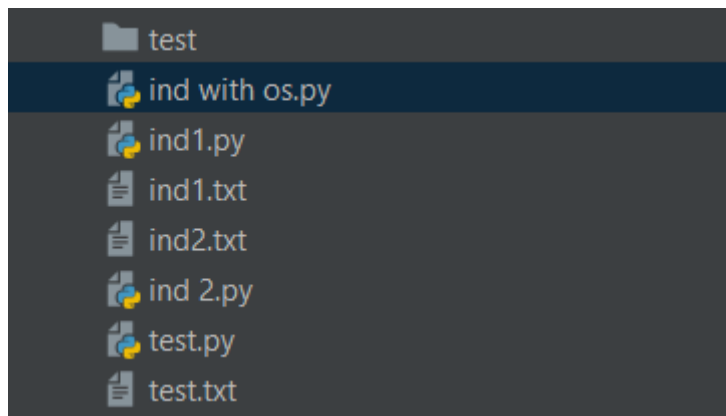


Рисунок 17.25 – Результат выполнения программы

Контрольные вопросы

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

Используя функцию `open()`, после ввода имени файла через запятую указать режим “r”.

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

Используя функцию `open()`, после ввода имени файла через запятую указать режим “w”.

3. Как прочитать данные из файла в языке Python?

Сначала необходимо открыть файл, вызвав функцию `open()`, затем использовать метод `read()`.

4. Как записать данные в файл в языке Python?

Сначала необходимо открыть файл, вызвав функцию `open()`, затем использовать метод `write()`.

5. Как закрыть файл в языке Python?

Использовать метод `close()` или открывать файл при помощи оператора `with`, который закрывает файл, после окончания работы с ним.

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` гарантирует, что критические функции выполнятся в любом случае. В основном она используется для работы с файлами разного типа, но также может использоваться для фиксации или

отката транзакции базы данных, для перенаправления стандартного вывода однопоточных программ.

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

Метод `writelines()` – добавляет последовательность строк в файл.

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

`os.name` - имя операционной системы.

`os.environ` - словарь переменных окружения.

`os.getpid()` - текущий `id` процесса.

`os.uname()` - информация об ОС.

`os.access(path, mode, *, dir_fd=None, effective_ids=False, follow_symlinks=True)` - проверка доступа к объекту у текущего пользователя.

`os.chdir(path)` - смена текущей директории.

`os.chmod(path, mode, *, dir_fd=None, follow_symlinks=True)` - смена прав доступа к объекту.

`os.link(src, dst, *, src_dir_fd=None, dst_dir_fd=None, follow_symlinks=True)` - создаёт жёсткую ссылку.

`os.listdir(path=".")` - список файлов и директорий в папке.

`os.makedirs(path, mode=0o777, exist_ok=False)` - создаёт директорию, создавая при этом промежуточные директории.

`os.symlink(source, link_name, target_is_directory=False, *, dir_fd=None)` - создаёт символическую ссылку на объект.

`os.truncate(path, length)` - обрезает файл до длины `length`.

`os.utime(path, times=None, *, ns=None, dir_fd=None, follow_symlinks=True)` - модификация времени последнего доступа и изменения файла.

`os.walk(top, topdown=True, onerror=None, followlinks=False)` - генерация имён файлов в дереве каталогов.

os.system(command) - исполняет системную команду, возвращает код её завершения.

os.urandom(n) - n случайных байт.

os.path - модуль, реализующий некоторые полезные функции на работы с путями.

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