Лабораторное занятие 8
Работа с файлами

1 Цель работы

1.1 Изучить процесс работы с файлами на Python.

2 Литература

2.1 Прохоренок, Н.А. Python 3. Самое необходимое / Н.А. Прохоренок, В.А. Дронов. – Санкт-Петербург: БХВ-Петербург, 2016. – с.18-50.

3 Подготовка к работе

3.1 Повторить теоретический материал (см. п.2).

3.2 Изучить описание лабораторной работы.

4 Основное оборудование

4.1 Персональный компьютер.

5 Задание

5.1 Создайте файл, в котором записано два целых числа, каждое в отдельной строке. Выведите в выходной файл их сумму.

5.2 Имеется входной файл и введенный с клавиатуры символ (например, “!”). Определить, есть ли данный символ в файле. Выведите с качестве результата слово “Yes” или ”No”.

5.3 Дан файл, каждая строка которого может содержать одно или несколько целых чисел, разделенных одним или несколькими пробелами. Вычислите сумму чисел в каждой строке и выведите эту сумму (для каждой строки выводится сумма чисел в этой строке).

5.4 Создайте текстовый файл. Определите сколько в нем букв (латинского алфавита), слов, строк

5.5 Имеется текстовый файл (создать самостоятельно). Напечатать:
а) его первую строку;
б) его пятую строку;
в) его первые 5 строк;
г) его строки с s1-й по s2-ю;
д) весь файл.

6 Порядок выполнения работы

6.1 Запустить Python IDLE и выполнить все задания из п.5.

6.2 Ответить на контрольные вопросы.

7 Содержание отчета

7.1 Титульный лист

7.2 Цель работы

7.3 Ответы на контрольные вопросы

7.4 Вывод

8 Контрольные вопросы

8.1 **Как считать файл построчно?**
8.2 **В чем разница между текстовыми и бинарными файлами?**

9 Приложение

При работе с текстовыми файлами необходимо выполнять следующую последовательность действий.

***Первый шаг* – это открытие файла с помощью функции open.**

Функция **open** возвращает ссылку на файловый объект, которую нужно записать в переменную, чтобы потом через данный объект использовать методы вводавывода:

f = open(<file>, <mode>)

<file> - путь к файлу.

Путь файла может быть **абсолютным**, то есть начинаться с буквы диска, например, *C://Temp/namefile.txt*. Либо можно быть **относительным**, например, *Temp/namefile.txt* - в этом случае поиск файла будет идти относительно расположения запущенного скрипта Python.

<mode> - устанавливает режим открытия файла в зависимости от того, что необходимо сделать с ним.

Возможны следующие режимы:

'r' открытие на чтение (является значением по умолчанию); если файл не найден, то генерируется исключение FileNotFoundError

'w' открытие на запись; если файл отсутствует, то он создается; если подобный файл уже есть, то он создается заново, и соответственно старые данные в нем стираются.

'x' открытие на запись, если файла не существует, иначе исключение.

'a' открытие на дозапись, информация добавляется в конец файла; если файл отсутствует, то он создается.

'b' открытие в двоичном режиме.

't' открытие в текстовом режиме (является значением по умолчанию).

'+' открытие на чтение и запись.

Режимы могут быть объединены, то есть, к примеру, **'rb'** - чтение в двоичном режиме. По умолчанию режим равен **'rt'**.

Пример. Открытие файла на для записи:

f = open('hello.txt', 'w')

***Второй шаг* – это собственно работа с файлом.**

«Работа» – это прежде всего чтение символов/строк из файла и запись

символов/строк в файл. Чтение из текстового файла может быть реализовано
одним из следующих способов.

Чтение файла

1. Метод **read()** считывает все содержимое из файла и возвращает строку, которая может содержать символы ‘\n’. Если методу **read** передать целочисленный параметр (read(n)), то будет считано не более (n) заданного количества символов.

```
# метод read без параметров считывает ВЕСЬ файл
f = open(‘hello.txt','r')
line = f.read()
print(line, end='')
f.close() # закрыть файл

# метод read с параметром считывается 10 первых символов
f = open('hello.txt','r')
line = f.read(10)
print(line, end='')
f.close() # закрыть файл
```

2. Метод **readline()** считывает одну строку из файла (до символа конца строки ‘\n’, возвращается считанная строка вместе с символом ‘\n’, поэтому для удаления символа ‘\n’ из конца файла удобно использовать метод строки **rstrip()**). Если считывание не было успешно (достигнут конец файла), то возвращается пустая строка.

```
# метод readline
f = open('hello.txt','r')
line = f.readline()
print(line, end='')
f.close() # закрыть файл
```

3. Метод **readlines()** считывает все строки из файла и возвращает список из всех считанных строк (одна строка — один элемент списка). При этом символы ‘\n’ остаются в концах строк.

При чтении файла можно столкнуться с тем, что его кодировка не совпадает с ASCII. В этом случае мы явным образом можем указать кодировку с помощью параметра **encoding**:

```
filename = ‘hello.txt’
with open(filename, encoding=’utf8’) as file:
text = file.read()
```

Конструкция **with**

При открытии файла или в процессе работы с ним мы можем

столкнуться с различными исключениями, например, к нему нет доступа и т.д. В этом случае программа выдаст в ошибку, а ее выполнение не дойдет до вызова метода close, и соответственно файл не будет закрыт. В этом случае мы можем обрабатывать исключения

```
try:
f = open('hello.txt', 'w')
try:
    f.write('hello world')
except Exception as e:
    print(e)
    finally:
    f.close()
except Exception as ex:
    print(ex)
```

В данном случае вся работа с файлом идет во вложенном блоке **try**. И если вдруг возникнет какое-либо исключение, то в любом случае в блоке
finally файл будет закрыт.
Однако есть и более удобная конструкция - конструкция **with**:

```
with open(<file>, <mode>) as file_obj:
инструкции
```

Эта конструкция определяет для открытого файла переменную file_obj и выполняет набор инструкций. После их выполнения файл автоматически
закрывается. Даже если при выполнении инструкций в блоке **with** возникнут
какие-либо исключения, то файл все равно закрывается.
Так предыдущий пример выглядеть следующим образом:

```
with open('hello.txt', 'w') as f:
f.write('hello world')
```

Запись в текстовый файл.
Чтобы открыть текстовый файл на запись, необходимо применить режим **'w'** (перезапись) или **'a'**