Практическая работа № 5. Работа с файлами Ввод-вывод данных.

Что рассмотрим?

- ⇨ Понятие файла в Python
- ⇨ основные способы ввода и вывода данных в *Python* с использованием консоли
- ⇨ работа с файлами: открытие, закрытие, чтение и запись.

Что мы знаем?

ВЫВОД ДАННЫХ В КОНСОЛЬ

Для вывода данных в консоль используется функция *print*.

Вспомним основные способы использования данной функции.

```
>>> print("Hello")
Hello
>>> print("Hello, " + "world!")
Hello, world!
>>> print("Age: " + str(23))
Age: 23
```

По умолчанию, в функции ***print*** используется ***пробел.***

```
>>> print("A", "B", "C")
A B C
```

Для ***замены разделителя*** необходимо использовать параметр ***sep*** функции ***print.***

```
>>> print("A", "B", "C", sep="#")
A#B#C
```

В ***качестве конечного элемента выводимой строки*** используется символ перевода строки.

```
>>> for i in range(3):
    print("i: " + str(i))
i: 0
i: 1
i: 2
```

Чтобы вывести текст в одну строчку используется параметр end=" слово ".

Примеры:

1)
```
for i in range(3):
   print("[i: " + str(i) + "]", end="->")
```

На консоли:

```
 [i: 0]->[i: 1]->[i: 2]->
>>> for i in range(3):
   print("[i: " + str(i) + "]", end="-это число")
```

На консоли:
[i: 0]-это число[i: 1]-это число[i: 2]-это число

! Как видите, индексация начинается с 0!

ВВОД ДАННЫХ С КЛАВИАТУРЫ

Для считывания вводимых с клавиатуры данных используется функция *input()*.

```
>>> input()
test
'test'
```

Для сохранения данных в переменной используется следующий синтаксис.

```
>>> a = input()
hello
>>> print(a)
hello
```

Если считывается с клавиатуры целое число, то строку, получаемую с помощью функции *input()*, можно передать сразу в функцию *int()*.

```
>>> val = int(input())
123
>>> print(val)
123
>>> type(val)
<class 'int'>
```

Для вывода строки-приглашения, в качестве аргумента функции input() ***используется текст, заключенный в кавычки (" ")***

```
>>> tv = int(input("input number: "))
input number: 334
>>> print(tv)
334
```

Работа с файлами

Файлом называется совокупность данных, сохраненная под определенным именем. Каждый файл на диске имеет обозначение, состоящее из двух частей: имени и расширения.

Имя состоит из символов (прописные и строчные латинские буквы, цифры и символы: !, @, #, %, ^, &, (,), ‘, ~, -, _).

Расширение отделяется от имени точкой, содержит до трех разрешенных символов, например: spisok.txt, kontakty.dat, go_my.exe.

Различают файлы двух видов: последовательного (текстовые) и произвольного (типизированные) доступа.

Последовательные файлы состоят из совокупности элементов различной длины, отделенных друг от друга разделителями (чаще используется пробел). Чтобы найти элемент в таком файле, надо просмотреть все элементы, предшествующие критерию поиска.

Для файла произвольного доступа можно организовать доступ к каждой компоненте файла.

Текстовые файлы наиболее распространены и в дальнейшем речь пойдет только о файлах данного типа.

Работа с файлом складывается из трех пунктов:

- открытие файла;
- чтение или запись;
- закрытие файла.

Файл в данный момент времени может быть в одном из двух состояний: либо открыт для записи, либо только для чтения.

1)***Открытие файла***

Для открытия файла используется функция ***open(),*** которая возвращает файловый объект.

Формат функции: **open(имя_файла, режим_доступа).**

Для указания режима доступа используется следующие символы:

r – открыть файл для чтения;

w – открыть файл для записи;

x – создать файл. При этом, если файл существует, то вызов функции ***open()*** завершится с ошибкой;

a – открыть файл для записи, при этом новые данные будут добавлены в конец файла, без удаления существующих;

b – бинарный режим;

t – текстовый режим;

+ – открывает файл для обновления.

По умолчанию файл открывается на чтение в текстовом режиме.

У файлового объекта имеет следующие ***атрибуты***.

file.closed – возвращает *true* если файл закрыт и *false* в противном случае;

file.mode – возвращает режим доступа к файлу, при этом файл должен быть открыт;

file.name – возвращает имя файла.

```
>>> f = open("test.txt", "r")
>>> print("file.closed: " + str(f.closed))
file.closed: False
>>> print("file.mode: " + f.mode)
file.mode: r
>>> print("file.name: " + f.name)
file.name: test.txt
```

***Для закрытия файла используется метод* close().**

Чтение данных из файла

Чтение данных из файла осуществляется с помощью методов **read(размер)** и **readline().**

Метод ***read(размер)*** считывает из файла определенное количество символов, переданное в качестве аргумента.

Если использовать этот метод без аргументов, то будет считан весь файл.

```
>>> f = open("test.txt", "r")
>>> f.read()
'1 2 3 4 5\nWork with file\n'
>>> f.close()
```

В качестве аргумента метода можно передать количество символов, которое нужно считать.

```
>>> f = open("test.txt", "r")
>>> f.read(5)
'1 2 3'
>>> f.close()
```

Метод ***readline()*** позволяет считать строку из открытого файла.

```
>>> f = open("test.txt", "r")
>>> f.readline()
'1 2 3 4 5\n'
>>> f.close()
```

Построчное считывание можно организовать с помощью оператора ***for*.**

```
>>> f = open("test.txt", "r")
>>> for line in f:
        print(line)
1 2 3 4 5
Work with file
>>> f.close()
```

Запись данных в файл

Для записи данных файл используется метод **write(строка)**, при успешной записи он вернет количество записанных символов.

```
>>> f = open("test.txt", "a")
>>> f.write("Test string")
11
>>> f.close()
```

Дополнительные методы для работы с файлами

Метод **tell()** возвращает текущую позицию “условного курсора” в файле. Например, если вы считали пять символов, то “курсор” будет установлен в позицию 5.

```
>>> f = open("test.txt", "r")
>>> f.read(5)
'1 2 3'

>>> f.tell()
5
>>> f.close()
```

Метод **seek(позиция)** выставляет позицию в файле.

```
>>> f = open("test.txt", "r")
>>> f.tell()
0
>>> f.seek(8)
8
>>> f.read(1)
'5'
>>> f.tell()
9
>>> f.close()
```

Оператор **with** обеспечивает автоматическое закрытие файла после завершения работы с ним.

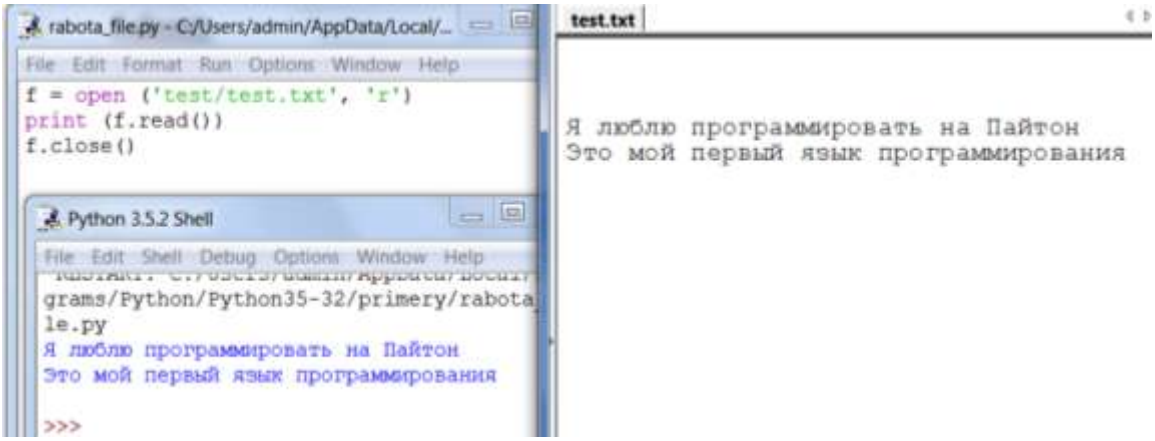
```
>>> with open("test.txt", "r") as f:
...     for line in f:
...             print(line)
...
1 2 3 4 5
Work with file
Test string
>>> f.closed
True
```

Рассмотрим примеры работы с файлами.

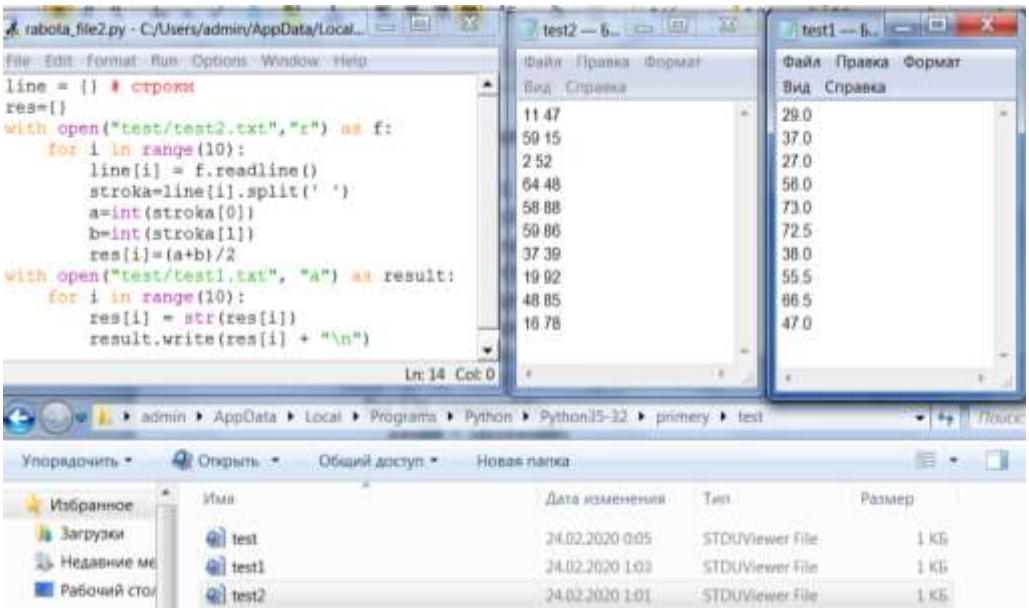
Пример_1. Создадим текстовый файл, записав в него текст.

```
print ('Введите текст: ')
text = input()
f = open ('text.txt', 'w')
f.write (text)
f.close()
```

Пример_2. Прочитать строки из текстового файла **test.txt**, находящегося в папке **test** и вывести их на экран



Пример *3.* Имеется файл test2.**txt** с двумя столбцами и десятью строками однозначных и двузначных чисел, разделенных пробелом. ***Найти среднее арифметическое*** этих чисел и записать их в новый текстовый файл test1.**txt.** ***Показать содержимое файлов test2.txt и test1.txt.***



ВОПРОСЫ И ЗАДАНИЯ

1. Что такое файл?
2. Какие требования предъявляются к именам файлов?
3. Зачем используется файловая переменная?
4. Назначение r, w, t, a, b, +, close?
5. Создать файл, содержащий 10 значений типа float. Вычислите произведение его элементов.
6. Создать файл, содержащий таблицу умножения 10 значений типа integer.
7. Составьте программу, которая считывает текст из файла.

Задание:

1) **Отработать Пайтоне все методы по теме**
2) **Переписать в конспект на память**
3) **Оформить в виде отчета в Ворде(скрины)**
4) **Сохранить отчет в PDFформате с титульным листом**
5) **Отправить до 9-00 следующего дня**

С уважением Баян Е