Урок 7

Цели урока

- Работа с файлами
- JSON
- lambda functions
- map, reduce, filter

Файлы

Python дает возможность работать с любыми типами файлов: txt, doc, docx, xlsx, и т.д. Но для некоторых требуются специальные пакеты.

Файл - это объект. Этот объект открывается, проделываются операции над ними, и закрываются. Все три части обязательны, особенно, закрытие файла.

Операции над файлами

file = open(“filename”, “mode”)
Это конструкция открывает файл. Для этого мы используем функцию open с названием имени файла ‘filename’. Если файл находится в другой папке, нужно указать путь. Второй аргумент - режим открытия файла.

Есть три основные типы файла: r - читать, w - write, a - append. Можно ещё указывать расширяющий режим + либо байтовый режим b.

file = open(‘my_txt.txt’, ‘w’)

Закрытие файла:
file.close()

Читать с файла

Каждую строку:

```
for line in file:
    print(line)
```

Все строки сразу:

```
lines = file.readlines()
```

Определенное количество символов или одну линию:

```
line = file.read([size])
```

Запись в файл

Запись в файл происходит функцией write.
file.write(‘string’)

write() принимает только тип string.

with statement

Чтобы быть уверенным, что файл закроется после завершения определенных операции над ними, можно пользоваться конструкцией with:

```
with open(“filename”, “mode”) as file:
    file.read(1024)
```

JSON

JSON - JavaScript Object Notation - формат файла для передачи данных.

```
{
    "book": [
        {
          "id":"01",
          "language": "Java",
          "edition": "third",
          "author": "Herbert Schildt"
        },

        {
          "id":"07",
          "language": "C++",
          "edition": "second“,
          "author": "E.Balagurusamy"
        }]
}
```

JSON and dict

Он по структуре похож на dictionary. Разница в том, что dictionary - это тип данных, json - формат файла. Соответственно, для работы с json мы транслируем словарь в json и обратно.

```
import json
d = {‘name’:’Python’, ‘version’: ‘3’}
file = open(‘my_json.json’, ‘w’)
json.dump(d, file) # запись в файл
file.close()

json_str = json.dumps(d) # перевод в формат JSON string

json_str = json.load(file) # прочитать из файла
```

Lambda functions

В Python реализовал возможность использования анонимных функции - lambda functions. Это такие функции, которые не имеют названия.
Синтаксис: lambda params: operation
Пример: lambda x: x**2

Такая функция может храниться в переменной. Почему? Потому что функция - это тоже объект.

```
square = lambda x: x**2
print(square(2))

max_value = lambda x,y: x if x> y else y
print(max_value(2, 3))
```

map, filter, reduce

Lambda функции приобретают свою мощь с такими функциями как map, filter, reduce.

map - функция, которая принимает два аргумента: другую функцию и любой итерируемый тип (лист, словарь, кортеж, т.д.). Его деятельность: перебрать все элементы данных и произвести операцию, указанный в виде функции.

filter - функция, которая фильтрует данные на основании ответа от функции.

reduce - функция, которая аккумулирует результат на основе всех элементов массива. Пример: сумма всех элементов, умножение всех элементов, и т.д.

Примеры map, filter, reduce

```
squares = list(map(lambda x:x**2, [1, 2, 3, 4, 5]))
print(squares)

even = list(filter(lambda x: True if x%2==0 else False, [1, 2, 3, 4, 5]))
print(even)

from functools import reduce
s = reduce(lambda x,y: x+y, [1, 2, 3, 4, 5])
print(s)
```