InnoDom

Ещё немного про файлы

Файлы (.json)

JSON – это **текстовый** формат для хранения данных, который позволяет быстро и просто обмениваться данными, формат позволяет просто кодировать и декодировать данные.

Этот формат основан на JavaScript, но он полностью *независимый* и может быть использован другими языками.

Файлы (.json)

В Python словари и json-формат имеют похожую структуру.

Объект в json формате - это неупорядоченный набор данных ключ/значение. Объект начинается с открывающейся **фигурной скобки** ({) и заканчивается закрывающейся **фигурной скобкой** (}).Такие файлы используются для того, чтобы не хранить данные в памяти, а освобождать её и хранить на диске.

Каждый **ключ** отделяется от значения двоеточием (`:`), каждые пары ключ/значение разделяются **запятой**.
Все любят `.json` формат за его понимание людьми и машинами.

Файлы (.json)

Пример json-файла:

```
{
 "films": [
   {"name": "Interstellar"...},
   {"name": "The Conjuring"...},
   {"name": "La La Land"...},
   {"name": "Pulp Fiction"...},
   {
     "name": "Spirited Away",
     "release": "2001",
     "producer": "Hayao Miyazaki",
     "evaluations": [
       {
         "name": "IMDb",
         "result": 8.6
       }
       {"name": "metacritic"...},
       {"name": "rotten tomatoes"...}
     ],
     "actors": ["Daveigh Chase","Suzanne Pleshette", "Jason Marsden"]
   }
 ]
}
```

Сериализаторы и десериализаторы

Json написан на JavaScript, поэтому и типы в таком формате отличаются от типов Python. Когда говорим про хранение данных и работе с файлом, в первую очередь имеется в виду *запись в файл* и *чтение из файла*.

Для того, чтобы записать данные требуется перевести данные в формат `json`, а для того чтобы прочитать - перевести из формата `json`. Такой процесс называется **кодированием** и **декодированием**. А часть кода, которая эти действия выполняет называется **сериализатором** и **десериализатором** соответственно.

Сериализаторы и десериализаторы

Сериализатор – это написанная инструкция, которая кодирует часть данных в определенный формат, для успешной отправки и декодирования. **Десериализатор** наоборот, декодирует полученные данные для работы или хранения их.

Сериализация типов в Python и Json:

Python	JSON
dict	object
list, tuple	array
str	string
int, long, float	number
True	true
False	false
None	null

Модуль json

позволяет работать с `.json` файлами. Модуль входит в стандартную библиотеку python, для подключения нужно ввести команду:

```
import json
```

Запись и чтение json-файлов

Для того, чтобы записать данные в json, требуется использовать функцию dump. Для этого нужно открыть файл, в который будем записывать и задать словарь, из которого записываем.

```
data = {
    "man":"Bob",
    "age": 22,
}


with open("man.json", 'w') as file:
    json.dump(data, file)
```

! Такие функции имеют много параметров, и знать все не обязательно.
К функции `dump`, существует параметр `indent`, контролирующий отступы в файле; `sort_keys`, задающий сортировку ключам в файле. Про остальные можно почитать в *документации*.

Запись и чтение json-файлов

Для того, чтобы прочитать данные из json-файла, требуется прочитать файл и использовать функцию `load()`:

```
import json


with open("class.json", 'r') as file:
    data = json.load(file)
print(data)
```

Для того чтобы сериализовать и десериализовать строку, и в строку данные формата json, можно использовать функции `dumps()` и `loads()` соответственно.

Файлы (.csv) и модуль csv

Формат `.csv` – один из самых распространенных форматов для импорта и экспорта электронных таблиц. В большинстве случаев формат универсален. Однако для обработки данных есть некоторые различие форматирования. К примеру csv легко переводится в формат **.xls** (**.xlsx**).

Файлы (.csv) и модуль csv

Для того, чтобы записывать в файлы данные, можно использовать `writer` модуля csv.
`csv.writer` – это объект, который позволяет записывать данные в csv файл. Данные передаются в виде **двумерного списка**, каждый список - это **строка**, каждый элемент - **столбец** таблицы. Далее создаем `writer` и с помощью его записываем данные в файл.

```
import csv

data = [["film", "release", "producer"],
        ['Spirited Away', '2001', 'Hayao Miyazaki'],
        ['Interstellar', '2014', 'Christopher Nolan']]

with open('films.csv', 'w') as file:
    writer = csv.writer(file)
    writer.writerows(data)
```

Файлы (.csv) и модуль csv

Для **записи** данных в csv файл, можно ещё использовать *словари*.
Для этого необходимо создать список, в котором будут названия столбцов; далее создать словарь, в котором **ключ** - это название столбца, а **значение** - то, что записать в этот столбец.
Используем объект `csv/DictWriter`. Словари составляем построчно:

```
import csv
with open('films.csv', 'w') as csvfile:
    fieldnames = ["film", "release", "producer"]
    writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
    writer.writeheader()
    writer.writerow({'film': 'Interstellar', 'release': '2014', 'producer': 'Christopher Nolan'})
    writer.writerow({'film': 'The Conjuring', 'release': '2013', 'producer': 'James Wan'})
    writer.writerow({'film': 'La La Land', 'release': '2016', 'producer': 'Damien Chazelle'})
    writer.writerow({'film': 'Pulp Fiction', 'release': '1994', 'producer': 'Quentin Tarantino'})
```

Файлы (.csv) и модуль csv

Для того, чтобы прочитать данные из csv файла, существует csv.reader:

```
import csv


with open('films.csv', 'r', newline='') as file:
    reader = csv.reader(file)
    for row in reader:
        print(', '.join(row))
```

! Для того, чтобы прочитать данные из csv файла безопасно (корректно), следует указывать аргумент `newline = ' '`

Файлы (.csv) и модуль csv

Для того, чтобы **прочитать** данные из csv-файла по именам столбцов, существует `csv.DictReader`:

```
import csv

with open('films.csv') as file:
    reader = csv.DictReader(file)
    for row in reader:
        print(row['film'], row['release'])
```