Практическая работа № 3

Работа с данными в Python.
Структуры для хранения данных в файлах.

Задание.

1. JSON-файл. Скачать файл **data.json**
2. Открытие JSON -файла в Python. Библиотека **json** и функция **open**
3. Работа с данными. библиотека Collections метод Counter().
4. Анализ данных.
5. Написать отчет.
6. Написать программу.
7. Выполнить задание.

Формат данных JSON

JSON (JavaScript Object Notation) это легковесный формат обмена данными. Людям его легко читать и вести в нем записи, а компьютеры запросто справляются с его синтаксическим анализом и генерацией.

JSON основан на языке программирования JavaScript. Но этот текстовый формат не зависит от языка и среди прочих может использоваться в Python и Perl. В основном его применяют для передачи данных между сервером и веб-приложением.

JSON построен на двух структурах:

Набор пар «имя-значение». Они могут быть реализованы как объект, запись, словарь, хеш-таблица, список «ключей-значений» или ассоциативный массив.

Упорядоченный список значений. Его реализуют в виде массива, вектора, списка или последовательности.

Работа с файлами JSON в Python

В Python есть ряд пакетов, поддерживающих JSON, в частности metamagic.json, jyson, simplejson, Yajl Py, ultrajson, и json. В этом руководстве мы будем использовать json, имеющий «родную» поддержку в Python.

Ниже приведен пример записи JSON. Как видим, представление данных очень похоже на словари Python.

```
{
 "article": [
      {
           "id": "01",
           "language": "JSON",
           "edition": "first",
           "author": "Derrick Mwiti"
      },
      {
           "id": "02",
           "language": "Python",
           "edition": "second",
           "author": "Derrick Mwiti"
      }
 ],
 "blog": [
      {
           "name": "Datacamp",

           "URL": "datacamp.com"
      }
 ]
}
```

Конвертируем JSON в объекты Python

Вышеуказанную JSON-строку мы можем извлечь при помощи метода json.loads() из модуля json. В итоге получим словарь Python.

```
import json
my_json_string = """{
   "article": [

      {
         "id":"01",
         "language": "JSON",
         "edition": "first",
         "author": "Derrick Mwiti"
      },

      {
         "id":"02",
         "language": "Python",
         "edition": "second",
         "author": "Derrick Mwiti"
      }
   ],

   "blog":[
   {
       "name": "Datacamp",
       "URL":"datacamp.com"
   }
   ]
}
"""
to_python = json.loads(my_json_string)
to_python['blog']
[{'URL': 'datacamp.com', 'name': 'Datacamp'}]
```

Конвертируем объекты Python в JSON

Используя json.dumps(), мы можем сконвертировать объекты Python в формат JSON.

```
blog = {'URL': 'datacamp.com', 'name': 'Datacamp'}
```

```
to_json= json.dumps(blog)
to_json
'{"URL": "datacamp.com", "name": "Datacamp"}'
```

Теперь давайте сравним типы данных в Python и JSON.

Python	JSON
dict	Object
list	Array
tuple	Array
str	String
int	Number
float	Number
True	true
False	false
None	null

Ниже мы покажем, как сконвертировать некоторые объекты Python в типы данных JSON.

Кортеж Python — в массив JSON

```
tuple_example = 'Mango', 'Banana', 'Apple'
print(json.dumps(tuple_example))
["Mango", "Banana", "Apple"]
```

Список Python — в массив JSON

```
list_example = ["Mango", 1, 3, 6, "Oranges"]
print(json.dumps(list_example))
["Mango", 1, 3, 6, "Oranges"]
```

Строка Python — в строку JSON

```
string_example = "This is a cool example."
print(json.dumps(string_example))
"This is a cool example."
```

Булевы значения Python — в булевы значения JSON

```
boolean_value = False
print(json.dumps(boolean_value))
false
```

Запись в файл JSON

Модуль json позволяет также записывать данные JSON в файл. Такие файлы сохраняют с расширением .json.

Давайте посмотрим, как это сделать. Для этого воспользуемся функцией **open**() с параметром w, сигнализирующим о том, что мы хотим записать в файл.

```
my_json_string = """{
   "article": [

      {
         "id":"01",
         "language": "JSON",
         "edition": "first",
         "author": "Derrick Mwiti"
      },

      {
         "id":"02",
         "language": "Python",
         "edition": "second",
         "author": "Derrick Mwiti"
      }
   ],

   "blog":[
   {
       "name": "Datacamp",
       "URL":"datacamp.com"
   }
   ]
}
"""
with open('test_file.json', 'w') as file:
    json.dump(my_json_string, file)
```

Чтение файлов JSON

Теперь продемонстрируем, как прочитать только что созданный нами файл JSON. Для его загрузки вызовем **json.load**().

```
with open('test_file.json', 'r') as j:
    json_data = json.load(j)
    print(json_data)
{
   "article": [

      {
```

```
            "id":"01",
            "language": "JSON",
            "edition": "first",
            "author": "Derrick Mwiti"
        },

        {
            "id":"02",
            "language": "Python",
            "edition": "second",
            "author": "Derrick Mwiti"
        }
    ],

    "blog":[
    {
        "name": "Datacamp",
        "URL":"datacamp.com"
    }
    ]
}
```

json.load vs json.loads

json.load используют для загрузки файла, а json.loads – для загрузки строки (loads расшифровывается как «load string»).

json.dump vs json.dumps

Аналогично, json.dump применяется, если нужно сохранить JSON в файл, а json.dumps (dump string) – если данные JSON нам нужны в виде строки для парсинга или вывода.

Работа с данными JSON в Data Science

Иногда при работе над проектами, связанными с data science, требуется загрузить данные в формате JSON. Библиотека для анализа данных Pandas предоставляет для этого функцию .**read_json**. Как только данные загружены, мы конвертируем их в объект dataframe при помощи атрибута pandas.DataFrame.

```
import pandas as pd
data = pd.read_json("https://api.github.com/users")
df = pd.DataFrame(data)
df
```

Пример 3.1

Открытие файла **data.json** и получение основной информации из файла

```
import json
with open('data.json', 'rb') as infile:
    data = json.load(infile)
print(type(data))    # dict
print(data.keys())  # dict_keys(['events_data'])
print('DATA    ',data['events_data'])    #  просмотр содержимого
data_list = data['events_data']
print(len(data_list)) # количество событий хранится в нашем логе
```

Содержимое файла data.json

{"events_data": [{"id": 47946124, "client_id": 62526, "user_id": 110777, "category": "page", "action": "enter", "options": {"name": "landing"}}, {"id": 47947076, "client_id": 61944, "user_id": 108565, "category": "page", "action": "enter", "options": {"name": "landing"}},

…

{"id": 47962218, "client_id": 27115, "user_id": 49440, "category": "table", "action": "sort", "options": {"dir": "ASC", "name": "deal_date"}},

…

{"id": 57540495, "client_id": 49700, "user_id": 84946, "category": "page", "action": "enter", "options": {"name": "landing"}}]}

Пример 3.2

Подсчитаем частоту встречаемости каждого из событий

```
categories = []
for item in data_list:
    category = item['category']
    categories.append(category)
print ('Categories  ',categories)

import collections
c = collections.Counter()
for category in categories:
    c[category] += 1
print (' Частота встречаемости  событий  ',c)
```

Пример 3.3

Теперь давайте посмотрим, какие клиенты совершают **события table**. Для этого пройдемся по каждому словарю из списка *data_list* и добавим значение *client_id* в новый список *table_clients*, но только в тех случаях, где *category = 'table'*:

```
table_clients = []
for item in data_list:
    client_id = item['client_id']
    category = item['category']
    if category == 'table':
        table_clients.append(client_id)
print (table_clients)
```

Пример 3.4

Выполните на основе набора данных data.json задание: Сколько клиентов не совершали действий с категориями (category) = datepicker и table?

```
import json
with open('data.json', 'rb') as infile:
    data = json.load(infile)
data_list = data['events_data']
spisok = []
spisok_pt = []
data_list = data['events_data']
for item in data_list: ## item is a dictionary
    category = item['category']
    client = item['client_id']
    if client not in spisok:
        spisok.append(client) ## all clients, unique
    if (category == 'datepicker' or category == 'table') and (client not in spisok_pt):
        spisok_pt.append(client) ## special clients unique

k1 = len(spisok)
k2 = len(spisok_pt)
print('Число клиентов не совершавших действий с категориями (category) = datepicker и table', k1-k2)
```

Пример 3.5

Выполните на основе набора данных data.json задание:

Напишите идентификатор клиента, который совершил максимальное количество действий с категорией (category) = page.

```
import json
with open('data.json', 'rb') as infile:
    data = json.load(infile)
data_list = data['events_data']
slovar = {}
for item in data_list: ## item is a dictionary
    category = item['category']
    client = item['client_id']
    if category == 'page':
        if client in slovar.keys():
            slovar[client] += 1
        else:
            slovar[client] = 1
max1 = 0
clientmax = ' '
for client in slovar.keys():
    if slovar[client] > max1:
        clientmax = client
        max1 = slovar[client]

print('client_id с максимальным количеством действий
с категорией (category) = page ', clientmax)
```

Пример 3.6

Выполните на основе набора данных data.json задание: Сколько клиентов совершили только одно действие с категорией (category) = page?

```
import json
with open('data.json', 'rb') as infile:
    data = json.load(infile)
data_list = data['events_data']
k = 0  ## number of clients
slovar = {}
for item in data_list: ## item is a dictionary
    category = item['category']
    client = item['client_id']
    if category == 'page':
        if client in slovar.keys():
            slovar[client] += 1
```

```
            else:
                slovar[client] = 1


    for client in slovar.keys():
        if slovar[client] == 1:
            k+=1

    print('Число клиентов совершивших только одно
действие с категорией (category) = page ', k)
```

Пример 3.7

Выполните на основе набора данных data.json задание: Сколько клиентов совершили оба действия с категориями (category) = page и report?

```
    import json
    with open('data.json', 'rb') as infile:
        data = json.load(infile)
    data_list = data['events_data']
    spisok_pages = []
    spisok_reports = []
    k=0
    for item in data_list: ## item is a dictionary
        category = item['category']
        client = item['client_id']
        if category == 'page' and (client not in
spisok_pages): ## append only unique numbers
            spisok_pages.append(client) ## clients unique
        if category == 'report' and (client not in
spisok_reports):
            spisok_reports.append(client)    ##    clients
unique

    for client in spisok_pages:
        if client in spisok_reports:
            k=k+1

    print('Число клиентов, совершивших оба действия с
категориями (category) = page & report',k)
```

Пример 3.8

Напишите программу, которая на основе списков городов и стран из произвольно текста формирует словарь с перечнем стран и городов, встретившихся в нем.

```
spisok = ['Moscow', 'Russia', 'Kiev', 'Ukraine',
          'Ivanovo',   'Russia',   'Berlin','Germany',
'Kharkov', 'Ukraine']

slovar = {}

L = len(spisok)

for i in range(1,L,2):
    country = spisok[i]
    city = spisok[i-1]
    if country not in slovar.keys():
        slovar[country] = []
    slovar[country].append(city)

print(slovar)
```

```
{'Russia': ['Moscow', 'Ivanovo'], 'Ukraine': ['Kiev', 'Kharkov'], 'Germany':
['Berlin']}
```

Пример 3.9

Напишите программу, которая анализирует текст на основе словаря, содержащий арифметическое выражение над числами, записанное словами. Вычислите это выражение.

```
slovar = {'two plus two': 'four',
          'two plus three': 'five',
          'three plus seven': 'ten'}

text = 'two plus three is equal ?'

stext = text.split()

for key in slovar.keys():
    if stext[0:3]==key.split():
        print(text)
        print(slovar[key])
```

```
two plus three is equal ?
five
```