hillel

Python Basic

Заняття 9

Import datetime

```
import datetime

dt_now = datetime.datetime.now()
print(dt_now)
```

```
from datetime import date

current date = date.today()
print(current_date)
```

```
import datetime

current date time = datetime.datetime.now()
current_time = current_date_time.time()
print(current_time)
```

Створення об'єктів **datatime**

```
import datetime

timeobj= datetime.time(8,48,45)
print(timeobj)
```

```
import datetime

date obj = datetime.datetime(2020,10,17)
print(date_obj)
```

timedelta

```
from datetime import date


first_date = date(2020, 10, 2)
second_date = date(2020, 10, 30)
delta = second_date - first_date
print(delta)
```

```
import datetime


now = datetime.time(9, 31, 0)
next_hour = datetime.time(10, 31, 0)
print('now < next_hour:', now < next_hour)
today = datetime.date.today()
next_week = datetime.date.today() + datetime.timedelta(days=7)
print('today > next_week:', today > next_week)
```

Import time

sleep()
localtime()
mktime()
asctime()
strftime()
strptime()

time.struct_time()

Индекс	Атрибут	Значения
0	tm_year	0000, ..., 2019, ..., 9999
1	tm_mon	1, 2, ..., 12
2	tm_mday	1, 2, ..., 31
3	tm_hour	0, 1, ..., 23
4	tm_min	0, 1, ..., 59
5	tm_sec	0, 1, ..., 61
6	tm_wday	0, 1, ..., 6; Monday is 0
7	tm_yday	1, 2, ..., 366
8	tm_isdst	0, 1 or -1

- JSON (JavaScript Object Notation) – простий формат обміну даними, заснований на підмножині синтаксису JavaScript. Модуль json дозволяє кодувати та декодувати дані у зручному форматі.

Де використовують JSON?

JSON зазвичай використовують для надсилання та отримання даних між сервером та клієнтом, де клієтом виступає вебсторінка або вебзастосунок.

Десеріалізація – процесом перетворення рядка в об'єкт.

json.load - метод зчитує файл у форматі JSON та повертає об'єкти Python

json.loads - метод зчитує рядок у форматі JSON та повертає об'єкти Python

dump відрізняється від dumps тим, що dump записує об'єкт Python у файл JSON, а dumps серіалізує об'єкт Python і зберігає його у вигляді рядка.

JSON	Python
object	dict
array	list
string	str
number (int)	int
number (real)	float
true	True
false	False
null	None

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

Table 4.1: Basic data types

Type	JSON	YAML
Integers	Yes	Yes
Floats	Scientific notation	Scientific notation
Number specifics	Not infinity	Also octal/Hexadecimal
Strings	Yes (Unicode)	Yes (Unicode)
Booleans	Yes	Yes
Arrays	Yes (sequences)	Yes
Associative arrays	Yes (objects)	Yes (mappings)
Null	Yes	Yes
Timestamps	As strings	Yes

Avaibility of basic data types in both formats (naming in parentheses).

	XML	JSON	YAML
Adoption and support	★★★★	★★★★	★★
Readability	★★	★★★	★★★★
Read and Write Speed	★★	★★★★	★
File Size	★	★★★	★★

```
pip install pyyaml
```

read_yaml.py

```
#!/usr/bin/python

import yaml

with open('items.yaml') as f:

    data = yaml.load(f, Loader=yaml.FullLoader)
    print(data)
```

read_docs.py

```
#!/usr/bin/python

import yaml

with open('data.yaml') as f:

    docs = yaml.load_all(f, Loader=yaml.FullLoader)

    for doc in docs:

        for k, v in doc.items():
            print(k, "->", v)
```

dumping.py

```
#!/usr/bin/python

import yaml

users = [{'name': 'John Doe', 'occupation': 'gardener'},
         {'name': 'Lucy Black', 'occupation': 'teacher'}]

print(yaml.dump(users))
```

dumping.py

```
#!/usr/bin/python

import yaml

users = [{'name': 'John Doe', 'occupation': 'gardener'},
         {'name': 'Lucy Black', 'occupation': 'teacher'}]

print(yaml.dump(users))
```

writing.py

```
#!/usr/bin/python

import yaml

users = [{'name': 'John Doe', 'occupation': 'gardener'},
         {'name': 'Lucy Black', 'occupation': 'teacher'}]

with open('users.yaml', 'w') as f:

    data = yaml.dump(users, f)
```