Serializacja

Serializacja

Serializacja w Python'ie:

- serializacja:
  - reprezentacja danych strukturalnych (obiektów, list, słowników, itp..) w postaci ciągów tekstowych lub binarnych,
  - umożliwia wymianę takich danych pomiędzy aplikacjami poprzez strumieniowanie danych (pliki, komunikacja bezpośrednia, sieciowa, itp),
  - przykładowe formaty serializacji dostępne w Python'ie to: json, pickle, yaml,
- json (JavaScript Object Notation):
  - 'lekki', niezależny od języka format wymiany danych strukturalnych,
  - wykorzystuje serializację tekstową,
  - obłsuga w Pythonie - modułu json (domyślnie instalowany),
- pickle:
  - format przeznaczony do wykorzystania w Pythoni’e,
  - wykorzystuje serializację binarna (bajtową),
  - jest 'szybszy' i zajmuje mniej miejsca w stosunku do serializacji tekstowej,
  - daje większe możliwości serializacji (potrafi serializować bardziej skomplikowane obiekty),
  - niebezpieczny w przypadku deserializacji nieznanych/niepewnych strumieni,
  - obsługa w Pythonie - modułu pickle (domyślnie instalowany),

Serializacja

Serializacja w Python'ie (cd.):

- yaml:
  - wykorzystuje serializację tekstową,
  - format jest w zamierzeniu 'bardzo czytelny' dla człowieka, przez co jest często wykorzystywany do zapisu plików konfiguracyjnych,
  - stanowi nadzbiór formatu json – oprogramowanie yaml powinno też przeczytać format json,
  - obłsuga w Pythonie – np. moduł pyyaml (trzeba doinstalować),
- cPickle:
  - jest implementacją w języku C obsługi formatu pickle (występują pewne drobne różnice), przez co jest wielokrotnie szybszy niż implementacja w Python'ie,
  - występował w Python 2.x jako oddzielny moduł,
  - w Pythonie 3.x jest częścią modułu pickle i jest wykorzystywany w sposób automatyczny (jak tylko interpreter wykryje 'odpowiednie warunki').

Serializacja

Przykład 1 – moduł json:

```
import json

slownik = {
    'k1':'w1',
    'k2':2,
    3:[1,2,3],
}
print(slownik)
print(type(slownik))

json_str=json.dumps(slownik)     # dict -> str
print(json_str)
print(type(json_str))

slownik2=json.loads(json_str)     # str -> dict
print(slownik2)
print(type(slownik2))
```

```
#output:
{'k1': 'w1', 'k2': 2, 3: [1, 2, 3]}
<class 'dict'>

{"k1": "w1", "k2": 2, "3": [1, 2, 3]}
<class 'str'>

{'k1': 'w1', 'k2': 2, '3': [1, 2, 3]}
<class 'dict'>
```

Serializacja

Przykład 2 – moduł pickle:

```
import pickle

slownik = {
   'k1':'w1',
   'k2':2,
   3:[1,2,3],
}
print(slownik)
print(type(slownik))

pickle_bts=pickle.dumps(slownik) # dict -> bytes
print(pickle_bts)
print(type(pickle_bts))

slownik2=pickle.loads(pickle_bts) # bytes -> dict
print(slownik2)
print(type(slownik2))
```

```
#output:
{'k1': 'w1', 'k2': 2, 3: [1, 2, 3]}
<class 'dict'>

b'\x80\x03}q\x00(X\x02\x00\x00\x00k1q\x01
X\x02\x00\x00\x00w1q\x02X\x02\x00\x00\x
00k2q\x03K\x02K\x03]q\x04(K\x01K\x02K\x0
3eu.'
<class 'bytes'>

{'k1': 'w1', 'k2': 2, 3: [1, 2, 3]}
<class 'dict'>
```

Zadanie 1

Napisz skrypt, który przechowuje profile osób w postaci listy (np. lista ‚persons'). Profil osoby to obiekt typu dict w postaci (mynums – moje ulubione liczby):

{ 'f_name' : '', 'l_name' : '', 'age' : 0, ‚mynums' : [0,0,0,…] }.

Dodaj trzy przykładowe profile do listy, a następnie zapisz listę z profilami do pliku w formacie JSON (np. plik 'persons.json').

Zadanie 2

Rozbuduj skrypt z zadania 1 następująco:

- lista profili jest odczytywana z pliku ('persons.json') ,
- możliwe jest dodawanie/usuwanie rekordów z linii komend (np. w pętli),
- przed zakończeniem działania skrypt aktualizuje plik profili.

Zadanie 3

Napisz skrypt, który umożliwia konwersję pliku profili z zadań 1 i 2 pomiędzy formatami JSON i PICKLE.

Przetestuj działanie.

Zadanie 4

W systemie Ubuntu do przechowywania statycznej konfiguracji sieci wykorzystuje się pliki w formacie 'yaml'. Napisz skrypt, który zaktualizuje (odczyta i zapisze korzystając z obsługi yaml) przykładowy plik konfiguracyjny i zamieni w nim adres serwera DNS 192.168.1.1 na 8.8.8.8.

Zadanie 4 – przykładowy plik

```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s25:
      addresses: [192.168.0.100/24]
      gateway4: 192.168.0.1
      nameservers:
        search: [example.com, sales.example.com, dev.example.com]
        addresses: [1.1.1.1, 192.168.1.1, 4.4.4.4]
```