Python średnio zaawansowany

Dzień 7

Blok nr 3:

Przetwarzanie danych

AGENDA

- Plain text
- Format CSV
- Format JSON

Plain text

Plain text

Zalety:

- dobry do zapisu pojedynczego rekordu, wczesnego prototypu lub prostego testu

Wady:

- brak metody opisu struktury (podziały między polami w rekordzie, podziały między rekordami)
- brak informacji o kodowaniu pliku

Dobrze pamiętać o wymuszeniu kodowania pliku podczas jego otwierania lub konwersji podczas przetwarzania.

Odczyt

```
with open('plain.txt', 'r', encoding=„UTF-8”) as file_in:
    data = file_in.read()
```

lub konstrukcja (decode()) znana z Python 2.x:

```
file_in = open('plain.txt', 'rb')
data_raw = file_in.read()
file_in.close()
data = data.decode('UTF-8')
```

Format CSV

Format CSV

CSV (ang. **c**omma-**s**eparated **v**alues, wartości rozdzielone przecinkiem) to prosty format przechowywania danych w plikach tekstowych.

Zalety: Ze względu na prostotę (czytelność) i względną uniwersalność używany jest do eksportu i importu danych między systemami IT.

Wady:

- Brak możliwości zapisu zagnieżdżonych struktur
- Mimo ustanowienia standardu RFC 4180, nie wszystkie implementacje są kompatybilne
- Nie niesie informacji o systemie kodowania (np. UTF-8 czy Win1250)

CSV

- Delimiter (separator) może być innym znakiem niż ',' - średniki i tabulatory są równie często wykorzystywane; zwróć na to uwagę podczas cytowania (!)
- Jeden wiersz w pliku to jedna encja
- Kolejność danych odpowiada przypisaniu do kolumny
- Jeżeli danych dla danej kolumny brakuje, pozostawia się pustą wartość pomiędzy separatorami
- Wczytując dane CSV będą one typu tekstowego ***(str),*** jeżeli typ ma znaczenie, należy **samodzielnie** dokonać detekcji typu danych

CSV - Przykłady

```
id,imie,nazwisko,wiek,nr_tel
1,jan,kowalski,33,123456789
2,pawel,nowak,18,987654321
```

W pierwszej linii najczęściej znajdują się nagłówki (nazwy) kolumn

W każdej kolejnej linii znajdują się rekordy (wpisy)

Przykład: https://www.nbp.pl/kursy/Archiwum/archiwum_tab_a_2018.csv

Zapis do CSV – writerow()

```
import csv

with open('eggs.csv', 'w', newline='') as csvfile:
    spamwriter = csv.writer(csvfile)
    spamwriter.writerow(['Spam'] * 5 + ['Baked Beans'])
    spamwriter.writerow(['Spam', 'Lovely Spam', 'Wonderful Spam'])
```

Odczyt z CSV

```
import csv

with open('eggs.csv', newline='') as csvfile:
    spamreader = csv.reader(csvfile, delimiter=' ')
    for row in spamreader:
        print(', '.join(row))
```

Zapis do CSV - DictWriter

```
import csv

with open('data.csv', 'w', newline='') as data_file:
    writer = csv.DictWriter(data_file, headers)
    writer.writeheader()
    writer.writerow(my_dict)
#   writer.writerows(list_of_dicts)
```

Odczyt z CSV - DictReader

```
import csv

with open('data.csv', 'r', newline='') as csv_data:
    reader = csv.DictReader(csv_data)
    # atrybut przechowujący nazwy kolumn
    headers = reader.fieldnames
    for row in reader:
        print(row)
```

Format JSON

JSON

```
{
    "name": "Luke Skywalker",
    "eye_color": "Blue",
    "active": true,
    "gender": "Male",
    "hair_color": "Blond",
    "height": 172,
    "films": [
        "https://swapi.co/api/films/1/",
    ],
    "homeworld": "https://swapi.co/api/planets/1/",
    "created": "2014-12-09T13:50:51.644000Z",
    "edited": "2014-12-10T13:52:43.172000Z",
}
```

str

boolean

int

list

datetime

Typy danych przechowywane w JSON po zdekodowaniu będą odpowiadać typom wbudowanym w język.

*** zobacz materiał z dnia drugiego**

Zapis do JSON

```
import json
import requests

url = 'http://swapi.co/api/planets/1/'
response = requests.get(url)
data = response.json()

with open('data.json', 'w') as data_file:
    json.dump(data, data_file, indent=4)
```

Odczyt z JSON

```
import json


with open('json_data.json', 'r') as data_file:
    data = json.load(data_file)
    print(data['name'])
    print(data['created'])
```

Dzięki!