Лекція 9. Сховища даних

Лектор – доцент, кандидат фізико-математичних наук
Кириченко Віктор Вікторович

ПЛАН ЛЕКЦІЇ

1. Серіалізація. Десеріалізація.
2. Збереження даних в форматі JSON.
3. Уявлення табличних даних в форматі CSV. Читання-запис файлу CSV за допомогою словника.
4. Створення власного діалекту. Сніффер, розпізнавання діалекту.
5. Формат XML файлу. Читання та запис даних в XML файл

Серіалізація. Десеріалізація.

Серіалізація — це процес перетворення складних структур даних, які використовуються в програмуванні, у форму, яку можна передавати між різними системами.

Десеріалізація - відновлення початкового стану структури даних до серіалізації.

Збереження даних в форматі JSON

збереження/зчитування об’єктів Python:

```
import json

data = {'name': 'Anna', 'age': 25, 'active': True}
```

Приклад файлу у JSON.

```
{
  "name": 'Anna',
  "age": 25,
  "active": true
}
```

Збереження даних в форматі JSON

```
import json
```

Серіалізація:

```
json_str = json.dumps(data)                    # у рядок
with open('data.json', 'w') as f:
    json.dump(data, f)                         # у файл
```

Десеріалізація:

```
data_loaded = json.loads(json_str)             # з рядка
with open('data.json') as f:
    data_from_file = json.load(f)              # з файлу
```

Уявлення табличних даних в форматі CSV. Читання-запис файлу CSV за допомогою словника.

Запис даних у CSV файл.

```
import csv


data = [
    {'Name': 'Anna', 'Age': 25},
    {'Name': 'Oleh', 'Age': 30},
]


with open('people.csv', 'w', newline='') as f:
    writer = csv.DictWriter(f, fieldnames=['Name', 'Age'])
    writer.writeheader()
    writer.writerows(data)
```

Уявлення табличних даних в форматі CSV. Читання-запис файлу CSV за допомогою словника.

Читання з файлу.

```
with open('people.csv') as f:

    reader = csv.DictReader(f)

    for row in reader:

        print(row['Name'], row['Age'])
```

Уявлення табличних даних в форматі CSV. Читання-запис файлу CSV за допомогою словника.

Діалекти CSV (CSV dialects) — це набір налаштувань, які визначають формат CSV-файлу: якими символами розділяються поля, як обрамляються рядки в лапки, які символи екрануються тощо.

Бібліотека `csv` у Python дозволяє визначати власні діалекти або використовувати вбудовані для роботи з CSV-файлами, які мають різні стилі.

`delimiter`	Символ, що розділяє стовпці (наприклад, `,` або `;`)
`quotechar`	Символ, що обрамляє текстові значення (зазвичай `"`)
`escapechar`	Символ для екранування спецсимволів
`lineterminator`	Символ наприкінці рядка (наприклад, `\n`)
`quoting`	Як обробляти лапки (наприклад, `csv.QUOTE_ALL`, `csv.QUOTE_MINIMAL`)
`skipinitialspace`	Пропуск пробілу після роздільника

Уявлення табличних даних в форматі CSV. Читання-запис файлу CSV за допомогою словника.

Для використання діалекту, його потрібно зареєструвати:

```
csv.register_dialect('my_dialect',
                     delimiter=';',
                     quotechar='"',
                     quoting=csv.QUOTE_MINIMAL)
```

Потім діалект можна використовувати під час читання файлу.

```
with open('custom.csv', 'w', newline='') as f:
    writer = csv.writer(f, dialect='my_dialect')
    writer.writerow(['Name', 'Age'])
    writer.writerow(['Ivan', 28])
```

Уявлення табличних даних в форматі CSV. Читання-запис файлу CSV за допомогою словника.

Сніффер (Sniffer) у контексті роботи з CSV у Python — це спеціальний клас `csv.Sniffer`, який автоматично визначає формат CSV-файлу, тобто його діалект.

Іноді ми не знаємо наперед: який роздільник використано (`,`, `;`, `\t`, інше), чи є заголовки в першому рядку, як обрамлено текстові поля. Сніффер може переглянути файл і автоматично визначити діалект.

Уявлення табличних даних в форматі CSV. Читання-запис файлу CSV за допомогою словника.

```
import csv

with open('data.csv', 'r') as f:
    sample = f.read(1024)        # зчитуємо зразок
    f.seek(0)                    # повертаємось на початок файлу

    sniffer = csv.Sniffer()
    dialect = sniffer.sniff(sample)
    has_header = sniffer.has_header(sample)

    print("Діалект:", dialect.delimiter)
    print("Є заголовок?", has_header)

    reader = csv.reader(f, dialect)
    for row in reader:
        print(row)
```

Формат XML файлу. Читання та запис даних в XML файл.

XML-файл (від англ. *eXtensible Markup Language*) — це текстовий файл, який використовується для структурованого зберігання та обміну даними у вигляді дерева тегів, подібно до HTML, але без фіксованого набору елементів.

```
<people>
    <person>
        <name>Anna</name>
        <age>25</age>
    </person>
    <person>
        <name>Oleh</name>
        <age>30</age>
    </person>
</people>
```

Формат XML файлу. Читання та запис даних в XML файл.

Читання XML у Python.

```
import xml.etree.ElementTree as ET

tree = ET.parse('people.xml')
root = tree.getroot()

for person in root.findall('person'):
    name = person.find('name').text
    age = person.find('age').text
    print(name, age)
```

Формат XML файлу. Читання та запис даних в XML файл.

Створення та запис XML за допомогою Python.

```
import xml.etree.ElementTree as ET


people = ET.Element('people')


person1 = ET.SubElement(people, 'person')

ET.SubElement(person1, 'name').text = 'Anna'

ET.SubElement(person1, 'age').text = '25'


tree = ET.ElementTree(people)

tree.write('people.xml', encoding='utf-8', xml_declaration=True)
```

ДЯКУЮ ЗА УВАГУ