Python for Data Science

Лекція 5. Ввід/Вивід

XML

- XML — це мова розмітки для зберігання та передачі структурованих даних
- XML має ієрархічну структуру та може бути представлений у вигляді дерева
- Вузлами дерева XML можуть бути елементи, їх атрибути, вказівки щодо обробки документа

Основні поняття XML

- Коректність — коректний документ має відповідати всім синтаксичним правилам
- Валідність — документ є валідним, якщо він коректний
- Синтаксичний аналізатор — програма, що читає XML документ та відтворює його структуру

XML 0 приклад

```
<Messages>
    <Message>
        <From>Alice</From>
        <To>Bob</To>
        <Text>Hi</Text>
    </Message>
    <Message>
        <From>Bob</From>
        <To>Alice</To>
        <Text>Hello</Text>
    </Message>
</Messages>
```

JSON

- JSON — текстовий формат обміну даними
- Базується на тексті та може бути прочитаний людиною
- Дозволяє описувати об’єкти та структури
- Використовується для передачі структурованої інформації

JSON — синтаксис

JSON використовує дві структури:

- Набір пар назва:значення
- Впорядкований список значень (аналог списку)

Значеннями можуть бути:

- Об’єкт — пара {назва:значення}
- Масив — послідовність змінних у квадратних дужках: [1, 2, 3]
- Змінна — число, рядок, логічне true\false, об’єкт, масив
- Рядок — послідовність символів у подвійних лапках: "string"

JSON — приклад

```
{
    "name": "John",
    "address": {
        "street": "First Street 12",
        "city": "Cityname"
    },
    "numbers": [
        "+123456712",
        "+765432112"
    ]
}
```

YAML

- YAML — формат серіалізації даних зручний для читання людиною
- Орієнтований на введення-виведення типових для мов програмування структур

YAML — послідовності

- YAML підтримує блочний та однорядковий формати:

```
--- # block
- first
- second
- third

--- # one-line
[first, second, first]
```

YAML - ім’я:значення

```
--- # block
name: John
surname: Johnson

--- # one-line
{name: John, surname: Johnson}
```

TOML

- TOML — формат файлів конфігурацій
- Забезпечує легкість читання людиною
- Базується на асоціативних масивах

TOML — синтаксис

Синтаксис складається з:

- Пар `ключ="значення"`
- `[розділів]`
- `# коментарів`

TOML — приклад

```
title = "Example TOML"

[creator]
name = "Admin"
ip = "8.8.8.8"

[messages]
    [message.m001]
    text = "Hello World"
    hidden = false

    [message.m002]
    text = "Hello World 2"
    hidden = true
```

Задачі

- За допомогою XML опишіть короткий список товарів інтернет-магазину смартфонів

- За допомогою JSON опишіть обліковий запис користувача уявної соц. мережі

- За допомогою YAML опишіть список мов програмування, які ви знаєте

Стандартне введення-виведення

- `stdin`
- `stdout`
- `stderr`

Стандартне введення та виведення – це ті потоки, з якими ми працюємо в консолі

Стандартне введення-виведення

- Файл можна відкрити у режимі роботи з рядками (str) чи у режимі роботи з байтами (bytes)

- Ці типи мають функції `str.encode()` і `bytes.decode()` для зворотної конвертації

- Можна імітувати роботу з файлом, зберігаючи при цьому дані у пам’яті за допомогою `io.BytesIO` чи `io.StringIO`

Введення-виведення

Найпростіші команди для роботи зі звичайними потоками - це `print` та `input`:

- `print(*objects, sep='', end='\n')`

  Роздруковує будь-які об'єкти як рядки

- `input([prompt])`

  Починає режим очікування вводу, зчитує рядок та повертає його

Файлова система

- Для роботи з файловою системою використовують модулі `os` і `os.path`, а ткож `shutil`

- Модуль `pathlib` було включено в стандартну бібліотеку з версії 3.4

Модуль os

Модуль `os` — надає функції для роботи з операційною системою.

`os.name` — ім’я операційної системи ('posix', 'nt', 'mac', 'os2', 'ce', 'java')
`os.environ` — словник змінних оточення
`os.listdir(path="")` — список файлів у директорії
`os.getcwd()` — поточна директорія
`os.system(command)` — виконує системну команду

Модуль os.path

Модуль `os.path` — вбудований підмодуль модуля `os`, призначений для роботи зі шляхами файлів.

`os.path.abspath(path)` — вертає нормалізований повний шлях
`os.path.basename(path)` — базове ім’я шляху
`os.path.exists(path)` — вертає True, якщо об’єкт існує
`os.path.getsize(path)` — розмір файлу у байтах
`os.path.isfile(path)` — чи є шлях файлом
`os.path.isdir(path)` — чи є шлях папкою

Модуль shutil

Модуль `shutil` — набор функцій високого рівня для роботи з елементами файлової системи. Часто використовується разом з `os`.

Файли

- Для роботи з вмістом файлів використовується функція `open()`:

```
open(file, mode='r', …)
```

- За замовчуванням файл відкривається у режимі читання

Режими відкриття файлів

'r'	читання (режим за замовчуванням)
'w'	запис (видаляє існуючий вміст)
'x'	створення (помилка, якщо файл вже існує)
'a'	запис у кінець
'b'	бінарний режим (без перекодування)
't'	текстовий режим (режим за замовчуванням)
'+'	update (читання та запис)

Контекстний менеджер

- Відкриті файли завжди треба закривати після завершення роботи з ними
- Для цього використовується контекстний менеджер `with`, аналог `try except finally`

```
with open('file.txt') as f:
    print(f.read())
```

- Файл, що був відкритий за допомогою with, у кінці роботи закривається

Основні методи файлового об’єкту

- читання: `readable(), read(), readline(), readlines()`
- запис: `writable(), write(), writelines()`
- навігація: `seekable(), seek(), tell()`

Задачі

- Створити функцію `last_lines()` яка приймає шлях до файлу та кількість останніх рядків та виводить їх у консолі

- Написати програму, яка створює новий файл і записує у нього усі числа від 0 до 100, що кратні 5

Задачі

- Створити функцію, яка отримує шлях до файлу та повертає кількість рядків у ньому

- Створити функцію, яка приймає шлях до каталогу та повертає список з його вмістом