Python & Data - Week 16

格式與資料交換 (下)

1. XML 格式介紹
2. JSON 格式介紹

XML 是甚麼?

- XML 與 HTML 具有相類似的語法
- 歷史上是由 HTML 變化出來
- XML 是一種較一般化的格式。一般而言，在 XML 所定義的 Tag 沒有特別意思
- XML 是早期電腦系統之間用於交換資料的其中一種共通語言

例如：

我們有一個用戶資料表，用戶資料表中有多個用戶 (User):

```
用戶資料表 (Users)
    用戶 (User)
        名 (First Name)
        姓 (Last Name)
        年齡 (Age)
```

我們可以用這樣的方式以 XML 格式表示

```xml
<?xml version="1.0" encoding="UTF-8"?>
<users>
    <user>
        <first-name>Peter</first-name>
        <last-name>Chan</last-name>
        <age>15</age>
    </user>
    <user>
        <first-name>Mary</first-name>
        <last-name>Lui</last-name>
        <age>20</age>
    </user>
</users>
```

可以有另一種方式表示相同的資料:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<users>
    <user first-name="Peter" last-name="Chan" age="15" />
    <user first-name="Mary" last-name="Lui" age="20" />
</users>
```

💡 兩種方式沒有優劣之分，取決於提供資料的人如何規劃

使用 Untangle 解析 XML 資料

- 有多種方法如 Pandas 都可以解析 XML 資料
- Untangle 相對簡單

In [ ]:

```
# 範例

import untangle

# content 是 string, 可以使用 requests / file system 直接讀取後使用 Untangle 處
content = """<?xml version="1.0" encoding="UTF-8"?>
<users>
    <user>
        <first-name>Peter</first-name>
        <last-name>Chan</last-name>
        <age>15</age>
    </user>
    <user>
        <first-name>Mary</first-name>
        <last-name>Lui</last-name>
        <age>20</age>
    </user>
</users>
"""

doc = untangle.parse(content)

# 直接使用 doc.XXX 取得 tag
# 使用 .cdata 直接存取 tag 內容
print(doc.users.user[0].first_name.cdata) # should show Peter
```

```
Peter
```

In [ ]:

```
# 範例 (讀取 Attribute)

import untangle

# content 是 string, 可以使用 requests / file system 直接讀取後使用 Untangle 處
content = """<?xml version="1.0" encoding="UTF-8"?>
<users>
    <user first-name="Peter" last-name="Chan" age="15" />
    <user first-name="Mary" last-name="Lui" age="20" />
</users>
"""

doc = untangle.parse(content)

# 直接使用 doc.XXX[attr-name] 取得 Attributes 內容
print(doc.users.user[0]['first-name']) # should show Peter
```

```
Peter
```

使用迴圈讀取資料

In [ ]:

```
# 範例 (讀取 Attribute)
import untangle

# content 是 string, 可以使用 requests / file system 直接讀取後使用 Untangle 處
content = """<?xml version="1.0" encoding="UTF-8"?>
<users>
    <user first-name="Peter" last-name="Chan" age="15" />
    <user first-name="Mary" last-name="Lui" age="20" />
</users>
"""

doc = untangle.parse(content)

for user in doc.users.user:
    print(f"Hi {user['first-name']} {user['last-name']}, your age is {user[
```

```
Hi Peter Chan, your age is 15
Hi Mary Lui, your age is 20
```

JSON 格式

- JSON 發展比 XML 發展較後
- JSON 是一種基於文字 (Text-based) 的格式
- JSON 格式基於 JavaScript 的 Object 表示方式 (故此叫 JavaScript Object Notation)
- 產生 JSON 和讀取都比較簡單，對於一些低效能設備會很適合 (IoT)
- JSON <-> 程式原生 Object 互相轉換 (在這裏是 Python)

範例

```
{
  "users": [
    {
      "first_name": "Peter",
      "last_name": "Chan",
      "age": 15
    },
    {
      "first_name": "Mary",
      "last_name": "Lui",
      "age": 20
    }
  ]
}
```

讀取 JSON

Deserializing JSON

Great, looks like you've captured yourself some wild JSON! Now it's time to whip it into shape. In the `json` library, you'll find `load()` and `loads()` for turning JSON encoded data into Python objects.

Just like serialization, there is a simple conversion table for deserialization, though you can probably guess what it looks like already.

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

Deserializing JSON

In [ ]:

```
# 範例 (讀取 JSON)

import json

json_doc = """{
  "users": [
    {
      "first_name": "Peter",
      "last_name": "Chan",
      "age": 15
    },
    {
      "first_name": "Mary",
      "last_name": "Lui",
      "age": 20
    }
  ]
}"""

# print(json_doc)

# print(json_doc)
users_doc = json.loads(json_doc)

print(type(users_doc))
print(users_doc['users'][0]['first_name']) # 直接存取 dict
```

```
<class 'dict'>
Peter
```

總結

上述直接讀取的方式較適合做小型的資料處理，遇上較大量的資料處理和分析，我們會使用 Pandas 等工具幫助。