LAB: CHUYỂN ĐỔI CÁC ĐỊNH DẠNG DỮ LIỆU API SỬ DỤNG PYTHON

I. Mô tả

- Máy tính cần cài đặt Python 3.

- Trong bài lab này, ta sẽ làm quen cách chuyển đổi các định dạng dữ liệu bằng cách điền thêm nội dung vào đoạn scipt, sử dụng các thư viện như ruamel, ElementTree, miniDOM.

II. Yêu cầu kỹ thuật

- Cài đặt thư viện.

- Viết code bằng Python thực hiện yêu cầu:

+ Chuyển đổi nội dung file user.yaml sang đối tượng user.

+ Chuyển đối tượng user trên sang định dạng JSON và xuất ra file.

+ Chuyển đổi nội dung file user.xml sang cấu trúc dạng tag:text.

III. Các bước thực hiện

***Bước 1*:** Cài đặt thư viện

- Vào cmd gõ lệnh: py –m pip install ruamel.yaml.

- Thư viện ElementTree, miniDOM đã được cài đặt sẵn khi cài Python.

***Bước 2*:** Điền vào đoạn script để thực hiện bài lab

Chuyển đổi file YAML:

Trong phần này, bạn sẽ phát triển đoạn script Python để chuyển đổi file YAML bằng cách sử dụng module ruamel.yaml. Đoạn code tổng thể gồm class User và các funtion đã được chuẩn bị để bạn sử dụng. Bạn sẽ học cách sử dụng mô-đun ruamel.yaml trong Python và có thể chuyển đổi các đối tượng trong YAML.

Bạn sẽ dựa vào cấu trúc dựng sẵn, đọc comment và điền vào những phần thiếu trong quá trình làm lab này.

Trong phần đầu đoạn code, import mô-đun ruamel.yaml:

```
# Import modules
import sys
from helper import *
from ruamel import yaml
```

Mở file user.yaml và xem qua cấu trúc của file. Đổi lại file lab01.py, trong chức năng chính, sử dụng function open để mở luồng đọc file user.yaml:

```
 # Open the user.yaml file as read only
 with open('user.yaml','r') as stream:
```

Sử dụng function safe_load từ thư viện ruamel.yaml để tải luồng. Lưu đối tượng vào biến user_yaml:

```
# Load the stream using safe_load
user_yaml = yaml.safe_load(stream)
```

Xác định kiểu của đối tượng trong biến user_yaml, sử dụng type function. In kết quả:

```
# Print the object type
print("Type of user_yaml variable:")
print(type(user_yaml))
```

Cho chạy code và kiểm tra kiểu của đối tượng:

```
##################
###### YAML ######
##################
Type of user_yaml variable:
<class 'dict'>
----------------------
```

Biến user_yaml là kiểu từ điển. Lặp lại các key trong biến user_yaml và in chúng ra màn hình. Các key từ điển giống như trong file user.yaml và ta có thể dùng chúng để truy nhập value tương ứng trong biến user_yaml:

```
# Iterate over the keys of the user_yaml and print them
print('Keys in user_yaml:')
for key in user_yaml:
      print(key)
```

Chạy code:

```
Keys in user_yaml:
id
first_name
last_name
birth_date
address
score
----------------------
```

Thêm một ví dụ mới của class User và đặt tên biến là user. Đối tượng User có nhiều thuộc tính đã được định nghĩa trong class User, được đặt ở trong file helper.py. Ta sẽ truyền giá trị từ biến user_yaml vào trong đối tượng:

```
# Create a new instance of class User
user = User()
```

Truyền giá trị từ user_yaml sang đối tượng user:

```
# Assign values form the user_yaml to the object user
user.id = user_yaml['id']
user.first_name = user_yaml['first_name']
user.last_name = user_yaml['last_name']
user.birth_date = user_yaml['birth_date']
user.address = user_yaml['address']
user.score = user_yaml['score']
```

In đối tượng user:

```
# Print the user object
print('User object:')
print(user)
```

Chạy code. Đối tượng user được in bằng định dạng là dictionary:

```
User object:
{'id': 3242, 'first_name': 'Ray', 'last_name': 'Smith', 'birth_date': datetime.date(1979, 8, 15), 'address':
[{'street': '94873 Ledner Rue', 'city': 'Royal Oak', 'postal_code': 44663, 'state': 'OH', 'primary': 1}, {'st
reet': '832 William Ave', 'city': 'Elnaville', 'postal_code': 17319, 'state': 'EL', 'primary': 0}], 'score':
18.3}
```

Viết và quan sát file JSON

Trong phần kế tiếp, chúng ta sẽ tiếp tục viết code dựa trên kết quả phần trước. Ta sẽ chuẩn bị đối tượng tới JSON serializable và tạo cấu trúc JSON sử dụng function dumps từ json package. Bạn sẽ viết cấu trúc JSON này ra file.

Tại đầu đoạn code import thêm mô-đun JSON:

```
# Import modules
import sys
from helper import *
from ruamel import yaml
import json
```

Hàm dumps có thể có nhiều thông số. Hai thông số cần thiết để tạo cấu trúc JSON từ đối tượng user là thông số cho đối tượng user và argument default để nối tiếp dữ liệu đã thông qua. Tuy nhiên để tạo cấu trúc JSON, phương thức chỉ có thể lấy được một vài kiểu đối tượng nhất định. Vì đối tượng user không phải argument hợp lệ cho function dumps nên cần phải chuyển đối tượng user và hàm *serializeUser* từ mô-đun helper sang cấu trúc JSON. Cuối cùng là lưu cấu trúc này vào biến user_json:

```
# Create JSON structure from the user object
user_json = json.dumps(user, default = serializeUser)
```

In biến user_json ra màn hình:

```
# Print the created JSON structure
print('Print user_json:')
print(user_json)
```

Chạy code để xem cấu trúc JSON:

```
Print user_json:
{"id": 3242, "first_name": "Ray", "last_name": "Smith", "birth_date": "1979-08-15", "address": [{"street": "9
4873 Ledner Rue", "city": "Royal Oak", "postal_code": 44663, "state": "OH", "primary": 1}, {"street": "832 Wi
lliam Ave", "city": "Elnaville", "postal_code": 17319, "state": "EL", "primary": 0}], "score": 18.3}
----------------------
```

Đầu ra của cấu trúc JSON là dạng có thể đọc được và sắp xếp theo thứ tự alpha bằng cách dùng sort_keys = True. Đặt tham số indent (thụt lề) bằng 4 vào phương thức dumps:

```
# Create JSON structre with indents and soreted keys
print('JSON with indents and sorted keys')
user_json = json.dumps(user, default = serializeUser, indent = 4,
                       sort_keys = True)
print(user_json)
```

```
JSON with indents and sorted keys
{
    "address": [
        {
            "city": "Royal Oak",
            "postal_code": 44663,
            "primary": 1,
            "state": "OH",
            "street": "94873 Ledner Rue"
        },
        {
            "city": "Elnaville",
            "postal_code": 17319,
            "primary": 0,
            "state": "EL",
            "street": "832 William Ave"
        }
    ],
    "birth_date": "1979-08-15",
    "first_name": "Ray",
    "id": 3242,
    "last_name": "Smith",
    "score": 18.3
}
```

Xuất ra file JSON:

```
# Print to file user.json

file = open("user.json","w")

file.write(user_json)

file.close()
```

So sánh bộ chuyển đổi XML khác nhau

Trong phần này, ta sẽ sử dụng hai bộ chuyển đổi XML là ElementTree và MiniDOM, để chuyển đổi file XML và nhận được kết quả giống nhau. Chúng ta sẽ học cách hoạt động của cả hai, khác biệt giữa chúng và khi đó bạn có thể quyết định sử dụng cái mình thích hơn. Mở file user.xml và xem qua nó. File XML bao gồm các thông tin giống như file YAML ta đã xem trước đó.

- Chuyển đổi file XML với ElementTree:

Mở lab01.py. Trên đầu đoạn code, import thư viện và đặt tắt là ET:

```
import xml.etree.ElementTree as ET
```

Parse file user.xml và đặt vào biến tree:

```
# Parse the user.xml file
tree = ET.parse('user.xml')
```

Lấy thành phần root:

```
# Get the root element
root = tree.getroot()
```

In ra màn hình những nhãn (sử dụng vòng lặp for):

```
# Print the tags
print('Tags in the XML:')
for element in root:
    print(element.tag)
```

```
Tags in the XML:
id
first_name
last_name
birth_date
address
address
score
```

In giá trị nhãn ID:

```
# Print the value of id tag
print('id tag value:')
print(root.find('id').text)
```

Sau đó chúng ta cho chạy file:

```
id tag value:
3242
----------------------
```

Tìm tất cả các thành phần với nhãn address trong root:

```
# Find all elements with the tag address in root
addresses = root.findall('address')
```

In các address trong file xml:

```
# Print the adresses in the xml
print('Addresses:')
for address in addresses:
     for i in address:
          print(i.tag + ':' + i.text)
```

```
Addresses:
street:94873 Ledner Rue
city:Royal Oak
postal_code:44663
state:OH
primary:1
street:832 William Ave
city:Elnaville
postal_code:17319
state:EL
primary:0
----------------------
```

Khác với ở trên là tìm và in các address, tiếp theo chúng ta sẽ in tất cả các thành phần trong root bao gồm các nhãn và các giá trị:

```
# Print the elements in root with their tags and values
print('Print the structure')
for k in root.iter():
     print(k.tag + ':' + k.text)
```

```
Print the structure
user:

id:3242
first_name:Ray
last_name:Smith
birth_date:1979-08-15
address:

street:94873 Ledner Rue
city:Royal Oak
postal_code:44663
state:OH
primary:1
address:

street:832 William Ave
city:Elnaville
postal_code:17319
state:EL
primary:0
score:18.3
```

- Chuyển đổi file XML với MiniDOM:

Trên phần đầu đoạn code ta sẽ import thêm thư viện và gọi tắt là MD:

```
import xml.dom.minidom as MD
```

Parse file XML và đặt vào biến dom:

```
# Parse the user.xml file
dom = MD.parse('user.xml')
```

In ra các nhãn trong file XML:

```
# Print the tags
print('Tags in the XML:')
for node in dom.childNodes:
    printTags(node.childNodes)
```

```
Tags in the XML:
id
first_name
last_name
birth_date
address
address
score
```

Truy cập vào giá trị ID và in nó ra màn hình:

```
# Accessing element value
print('Accessing element value')
idElements = dom.getElementsByTagName('id')
print(idElements)
elementId= idElements.item(0)
print(elementId.childNodes)
idValue = elementId.firstChild.data
print(idValue)
```

```
Accessing element value
[<DOM Element: id at 0x283ba52e408>]
[<DOM Text node "'3242'">]
3242
```

In các thành phần có nhãn 'address':

```
# Print elements from the DOM with tag name 'address'
print('Addresses:')
for node in dom.getElementsByTagName('address'):
      printNodes(node.childNodes)
```

```
Addresses:
street:94873 Ledner Rue
city:Royal Oak
postal_code:44663
state:OH
primary:1
street:832 William Ave
city:Elnaville
postal_code:17319
state:EL
primary:0
----------------------
```

In cả cấu trúc file XML bằng printNodes:

```
# Print the entire structure with printNodes
print('The structure:')
for node in dom.childNodes:
     printNodes(node.childNodes)
```

```
The structure:
id:3242
first_name:Ray
last_name:Smith
birth_date:1979-08-15
address:

  street:94873 Ledner Rue
  city:Royal Oak
  postal_code:44663
  state:OH
  primary:1
address:

  street:832 William Ave
  city:Elnaville
  postal_code:17319
  state:EL
  primary:0
score:18.3
```

- Sử dụng Namespaces:

Parse file item.xml vào biến itemTree:

```
# Parse the item.xml file
itemTree = ET.parse('item.xml')
```

Lấy thành phần root:

```
# Get the root element
root = itemTree.getroot()
```

Định nghĩa namespaces:

```
# Define namespaces
namespaces = {'a':'https://www.example.com/network','b':'https://www.example.com/furniture'}
```

Đặt từ khóa table trong namespaces và đặt vào biến elementsInNSa,b:

```
# Set table as the root element
elementsInNSa = root.findall('a:table',namespaces)
elementsInNSb = root.findall('b:table',namespaces)
```

In các thành phần trong namespace a:

```
# Elements in NS a
print('Elements in NS a:')
for e in elementsInNSa:
    for i in e.iter():
        print(i.tag + ':'+ i.text)
```

```
Elements in NS a:
{https://www.example.com/network}table:

{https://www.example.com/network}tr:

{https://www.example.com/network}td:Router
{https://www.example.com/network}td:Switch
----------------------
```

In các thành phần trong namespace b:

```
# Elements in NS b
print('Elements in NS b:')
for element in list(elementsInNSb[0]):
     print(element.tag + ":" + element.text)
```

```
Elements in NS b:
{https://www.example.com/furniture}name:Coffee Table
{https://www.example.com/furniture}length:180
{https://www.example.com/furniture}width:80
```