Lab: Sử dụng thư viện trong Python để lấy danh sách thông tin các thiết bị trong Fabric SD-WAN của Cisco

1. Mô tả

- Học viên thực hiện kết nối đến Sandbox SD-WAN (vManage), viết code để lấy danh sách thông tin các thiết bị bằng ngôn ngữ Python.

2. Yêu cầu kĩ thuật

- Cài đặt thư viện requests trên máy tính.
- Kết nối đến SD-WAN vManage, thông tin Vmanage trong bài lab:
  - ❖ Host: https://sandbox-sdwan-1.cisco.com
  - ❖ Username: devnetuser
  - ❖ Password: RG!_Yw919_83
- Viết code bằng Python thực hiện yêu cầu:

  + Đăng nhập và xác thực.

  + GET requests, POST requests.

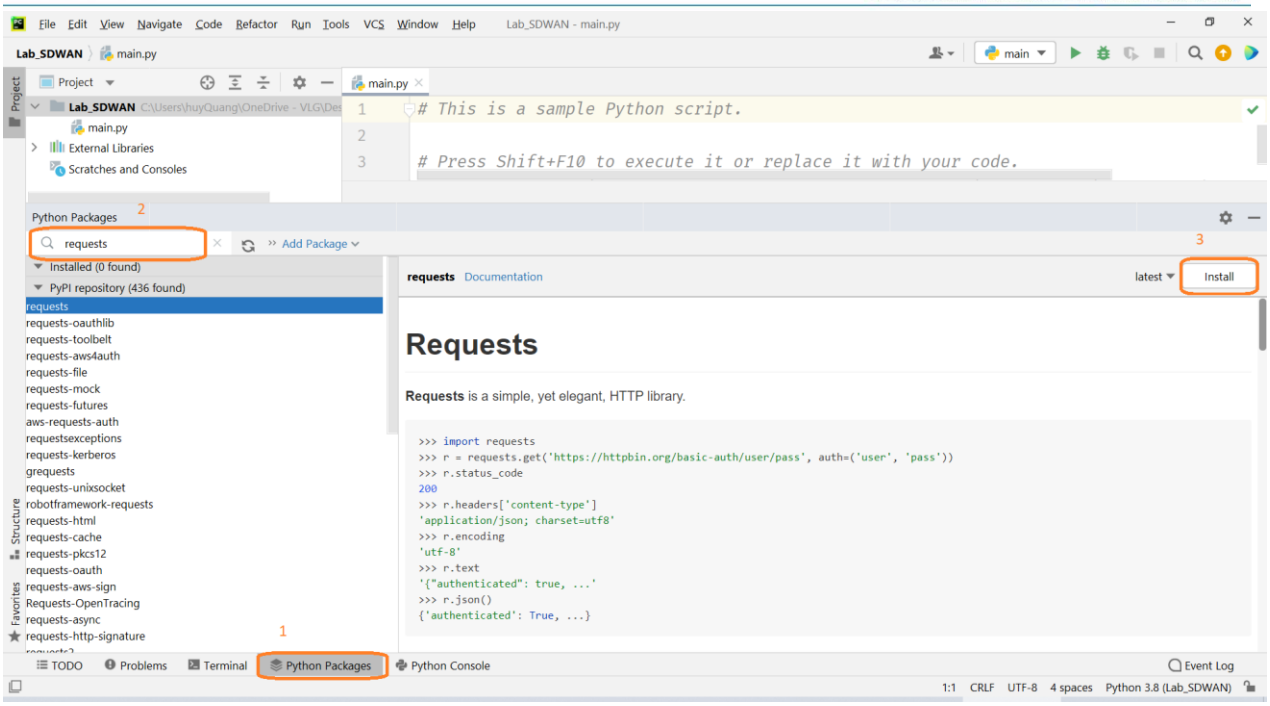
  + Lấy danh sách thông tin các thiết bị trong Controller.

3. Các bước thực hiện

❖ Cài đặt thư viện requests

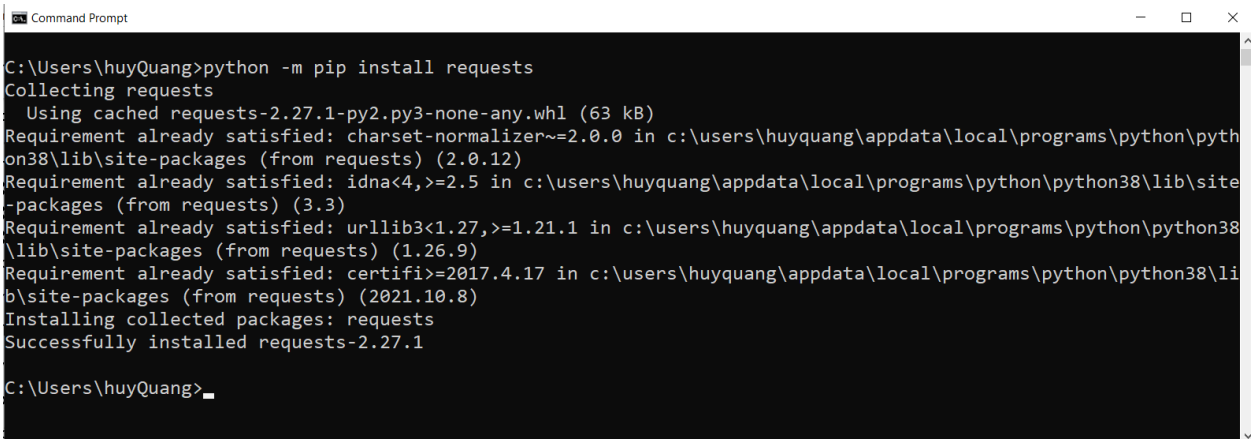
Bài lab sử dụng **IDE Pycharm**

Vào phần **Python Packages** tìm kiếm thư viện **requests** và click install, thư viện sẽ được install vào **virtualenv**



Nếu bạn sử dụng IDE khác, không phải Pycharm và **virtualenv** trong Pycharm

Truy cập CMD nhập lệnh python -m pip install requests, cẩn thận chú ý phiên bản python và pip sử dụng để cài đặt thư viện



❖ Đăng nhập và xác thực bằng Python code

Chúng ta sẽ xây dựng các hàm và class trong Python để đăng nhập và xác thực, lợi thế của việc sử dụng hàm và class là việc tái sử dụng được code.

Xây dựng hàm authentication

Bước 1: Đầu tiên bạn cần xác định api endpoint cần trỏ đến, https://{{vmanage}} và sau đó đến phần resource /j_security_check. Giá trị đó sẽ được lưu trong một biến riêng biệt:

- Biến base_url_str được dùng để chưa vManage endpoint ở định dạng chuỗi (string)
- Biến login_action chứa địa chỉ resource login

Bước 2: Chỉ định username và password trong một biến khác tên là login_data, cả 2 được lưu dưới dạng cặp key value hay còn gọi là dictionary trong Python.

Tổng kết chúng ta có 3 biến được định nghĩa:

```
base_url_str = f'https://{vmanage_ip}'

login_action = '/j_security_check'

login_data = {'j_username' : username, 'j_password' : password}
```

Endpoint và resource được chia thành hai biến khác nhau. Nếu một trong hai cần thay đổi, bạn chỉ cần thay đổi ở một biến cụ thể.

Bước 3: Tiếp theo bạn cần kết hợp 2 biến đó lại để hoàn thành login URL. Tến biến là login_url được kết hợp lại bởi base_url_str và login_action.

```
login_url = base_url_str + login_action
```

Bước 4: Để mở ra một session mới trong Python, chúng ta sử dụng phương thức session trong thư viện requests. Cùng với phương thức POST và thông tin j_username, j_password

của biến login_data để gửi request đến vManage. Và nếu bạn không muốn kiểm tra chứng chỉ tự ký (self-signed certificates) thì có thể chuyển option verify thành False.

```
sess = request.session()

login_response = sess.post(url=login_url, data=login_data, verify=False)
```

Bước 5: Chúng ta cần chỉ định rõ khi hàm login gửi request đi sẽ trả về thành công hay thất bại. Khi xác thực thành công hàm sẽ trả về 200 OK nhưng không có bất kỳ dữ liệu nào, khi xác thực thất bại Python sẽ trả về Login Failed trên màn hình và thoát khỏi hàm.

```
if b'<html>' in login_response.content:
   print ("Login Failed")
   sys.exit(0)
```

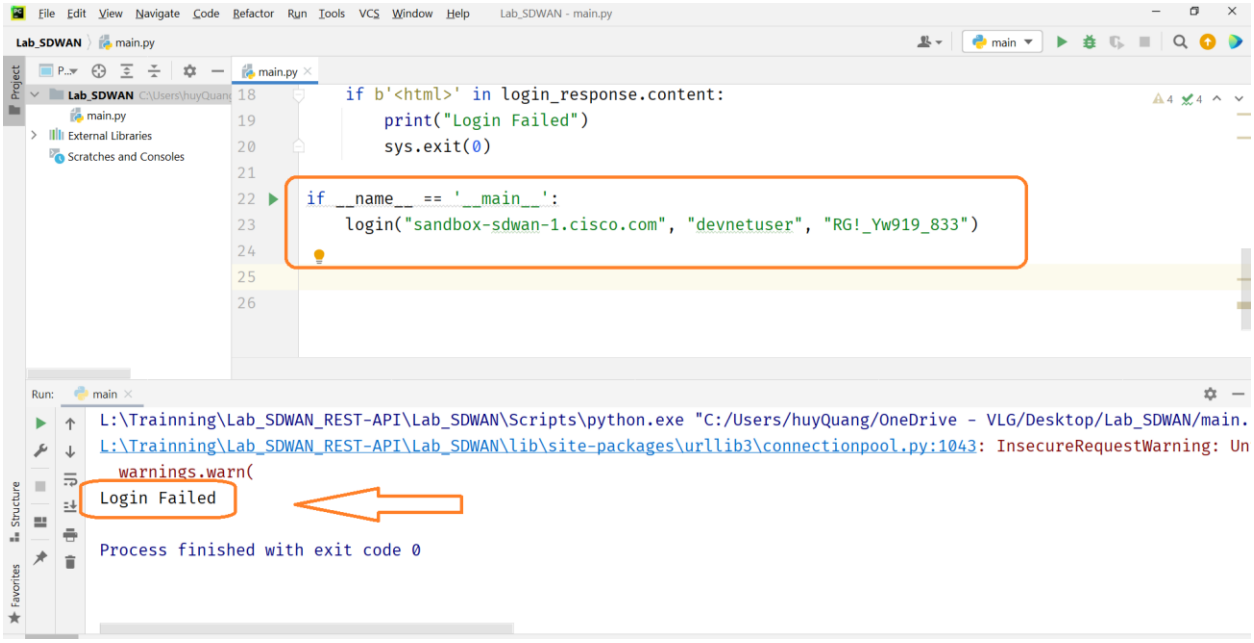
Hàm authentication sau khi hoàn thành

Bạn đã xây dựng được một hàm để login và xác thực với Cisco SD-WAN. Hàm sẽ cần những tham số đầu vào sau: địa chỉ vManage, uername và password

Code:

```
def login(vmanage_ip, username, password):
    """Login to vmanage"""
    base_url_str = f'https://{vmanage_ip}'

    login_action = '/j_security_check'

    login_data = {'j_username' : username, 'j_password' : password}

    login_url = base_url_str + login_action

    sess = requests.session()

    login_response = sess.post(url=login_url, data=login_data, verify=False)

    if b'<html>' in login_response.content:
        print ("Login Failed")
        sys.exit(0)
```

Kết quả trả về khi khai báo sai password và login thất bại:



❖ GET requests trong Python

Phần này sẽ xây dựng một hàm tên get_request() gọi dữ liệu từ vManage REST API. Hàm sẽ nhận vào 2 tham số đó là địa chỉ ip của vManage và resource để trỏ đến data muốn lấy. phần resource đó sẽ được đặt trong tham số tên mount_point.

Bước 1: Định nghĩa URL

```
url = f'https://{vmanage_ip}/dataservice/{mount_point}'
```

Bước 2: Sử dụng phương thức GET để thực hiện call api và lưu trữ trong biến response

```
response = requests.request("GET", url, verify=False)
data = response.content
return data
```

Bước 3: Kết hợp lại để xây dựng ra một hàm get_request() hoàn chỉnh:

```
def get_request(vmanage_ip, mount_point):
  """GET request"""
  url = f'https://{vmanage_ip}/dataservice/{mount_point}'

  response = requests.request("GET", url, verify=False)
  data = response.content
  return data
```

❖ POST requests trong Python

Tạo một hàm post_request(), hàm này sẽ nhận vào những tham số sau:

- Địa chỉ ip vManage
- Resource để gọi API
- Payload
- Phần Content-Type ở phần header được thực thi dưới dạng application/json

Bước 1: Định nghĩa URL

```
url = f'https://{vmanage_ip}/dataservice/{mount_point}'
```

Bước 2: Payload data sẽ được gửi đi cùng với request được lấy tự tham số đầu vào của hàm post_requests()

```
def post_request(vmanage_ip, mount_point, payload, headers={'Content-Type':
'application/json'}):
```

```
  url = f'https://{vmanage_ip}/dataservice/{mount_point}'
  payload = json.dumps(payload)
```

Bước 3: Sử dụng phương thức POST gọi api và dữ liệu trả về được lưu trữ trong biến response. Phương thức json() được sử dụng để định dạng lại dữ liệu trả về cho người dùng.

```
response = requests.request("POST", url, data=payload, headers=headers,
verify=False)
data = response.json()
return data
```

Bước 4: Kết hợp lại để xây dựng ra một hàm post_request() hoàn chỉnh:

```
def post_request(vmanage_ip, mount_point, payload, headers={'Content-Type':
'application/json'}):
  """POST request"""
  url = f'https://{vmanage_ip}/dataservice/{mount_point}'
  payload = json.dumps(payload)

  response = requests.request("POST", url, data=payload, headers=headers,
verify=False)
  data = response.json()
  return data
```

❖ Xây dựng rest_api_lib Python class

Chúng ta đã xây dựng được 3 hàm để có thể tái sử dụng lại khi tương tác với vManage REST API:

- login()
- get_request()
- post_request()

Trong phần này chúng ta sẽ tạo ra class đầu tiên trong chương trình.

Class trong Python được định nghĩa cho một đối tượng bao gồm một tập các attributes (thuộc tính) đặc trưng cho tất cả các đối tượng của lớp. Attributes gồm: data members và methods được gọi thông qua ký hiệu dấu chấm.

Hàm __init__ là method đặc biệt, gọi là constructor. Hàm này được Python tự động gọi khi một instance mới được tạo ra. Self đại diện cho một instance của một class, bạn có thể truy cập các thuộc tính và các phương thức của class thông qua self.

Thực hiện:

Bạn sẽ khởi tạo phương thức __init__ cùng với các thuộc tính cho địa chỉ ip vManage, một dict rỗng được tạo ra dành cho REST API **session** và sẽ được khởi tạo ở hàm login.

Tiếp theo, bạn cần định nghĩa các phương thức của class rest_api_lib. Bạn có thể sử dụng lại các hàm đã được định nghĩa ở phần trước login(), get_request(), post_request().

Bạn sẽ cần phải chỉnh sửa lại các hàm đó để có thể ghép vào class. Mỗi phương thức trong Python class cần có tham số self được truyền vào input. Bạn cũng cần phải lưu giữ lại giá trị session để có thể sử dụng nó cho các phương thức sau của class.

Sau khi đã kết hợp tất cả các hàm cũng như tham số, ta sẽ có được class rest_api_lib như sau:

```
class rest_api_lib:
```

```
    def __init__(self, vmanage_ip, username, password):
        self.vmanage_ip = vmanage_ip
        self.session = {}
        self.login(self.vmanage_ip, username, password)

    def login(self, vmanage_ip, username, password):
        """Login to vmanage"""
        base_url_str = f'https://{vmanage_ip}/dataservice/{mount_point}'



        login_action = '/j_security_check'

        #Format data for loginForm
        login_data = {'j_username' : username, 'j_password' : password}

        #Url for posting login data
        login_url = base_url_str + login_action
        url = base_url_str + login_url

        sess = requests.session()
        #If the vmanage has a certificate signed by a trusted authority change verify to
True
        login_response = sess.post(url=login_url, data=login_data, verify=False)



        if b'<html>' in login_response.content:
            print ("Login Failed")
            sys.exit(0)
```

```
        self.session[vmanage_ip] = sess

    def get_request(self, mount_point):
        """GET request"""
        url = f'https://{self.vmanage_ip}/dataservice/{mount_point}'
        #print url
        response = self.session[self.vmanage_ip].get(url, verify=False)
        data = response.content
        return data

    def post_request(self, mount_point, payload, headers={'Content-Type':
'application/json'}):
        """POST request"""
        url = f'https://{self.vmanage_ip}/dataservice/{mount_point}'

        payload = json.dumps(payload)
        print (payload)

        response = self.session[self.vmanage_ip].post(url=url, data=payload,
headers=headers, verify=False)
        data = response.json()
        return data
```