Chapter 9

Python Network Programming

Nội dung

- Dữ liệu Internet và mô hình Request/Response
- Thư viện urllib trong python
- Rest API và Webservice

Dữ liệu Internet và mô hình Request/Response

- HTTP
  - HTTP Message
  - HTTP Request
  - HTTP Response
- Cookie

HTTP

- HTTP (Hyper-Text-Transfer-Protocol) là một giao thức mức ứng dụng cho các hệ thống phân phối, cộng tác, đa phương tiện. Nó được thiết kế bởi Tim Berners Lee (1989). Giao thức này dựa trên giao thức TCP/IP. Đây là nền tảng đã được World-Wide Web (WWW) sử dụng từ năm 1990, cho phép tải (fetching) các nguồn tài nguyên trên mạng như tài liệu HTML, text, video, ảnh.
- Nó cũng là nền tảng cho bất kỳ trao đổi dữ liệu nào được thực hiện trên Web với mô hình Client/Server. Các yêu cầu (Request/HTTP Request) từ client được gửi đến Server, sau đó Server xử lý yêu cầu và gửi lại (Response/HTTP Response) dữ liệu đến client. Các Request và Response là các Message (HTTP Message).

Stateless:

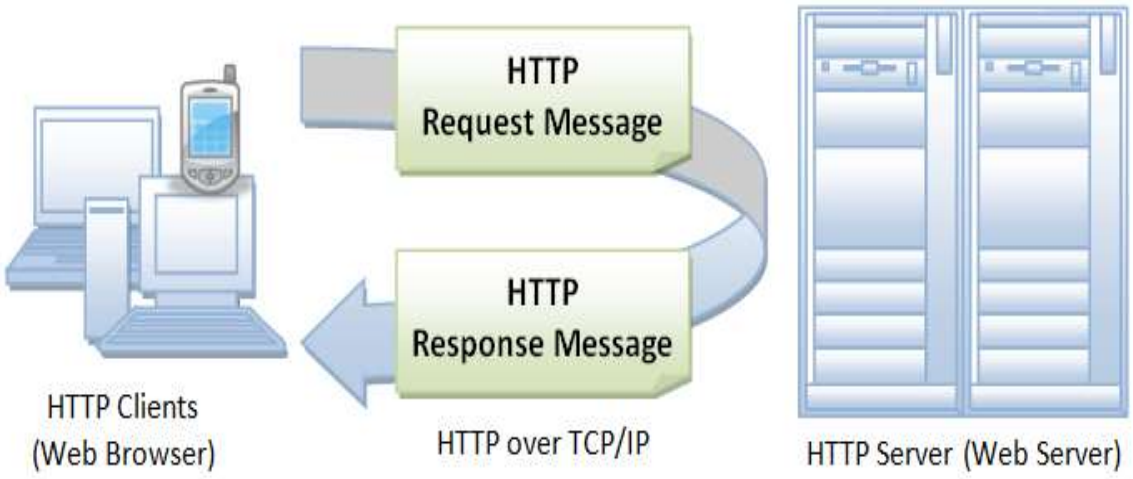
- HTTP là giao thức phi trạng thái (stateless), không có mối liên hệ nào giữa hai yêu cầu (request) được gửi đi, dù nó thực hiện trên cùng một kết nối (đến cùng 1 server). Thí dụ, client, thông qua web browser (như chrome/firefore) gửi yêu cầu đến https://www.vanlanguni.edu.vn/dao-tao/nganh-dao-tao, tiếp tục nó có thể gửi một yêu cầu khác đến https://www.vanlanguni.edu.vn/trang-chu/lich-su, cả hai yêu cầu này không có mối liên hệ gì với nhau.

Cookie:

- Đôi khi một ứng dụng cần chia sẻ trạng thái giữa những yêu cầu của phiên làm việc (session). Chẳng hạn trong ứng dụng bán hàng online, người dùng thêm 1 sản phẩm vào giỏ. Sau đó, quay lại trang chính để tìm kiếm và thêm sản phẩm khác vào giỏ. Do HTTP là stateless, nên HTTP Cookie sẽ hỗ trợ nhằm duy trì trạng thái của hai phiên làm việc trên. Phần Cookie này được thêm vào phần header của HTTP Message.

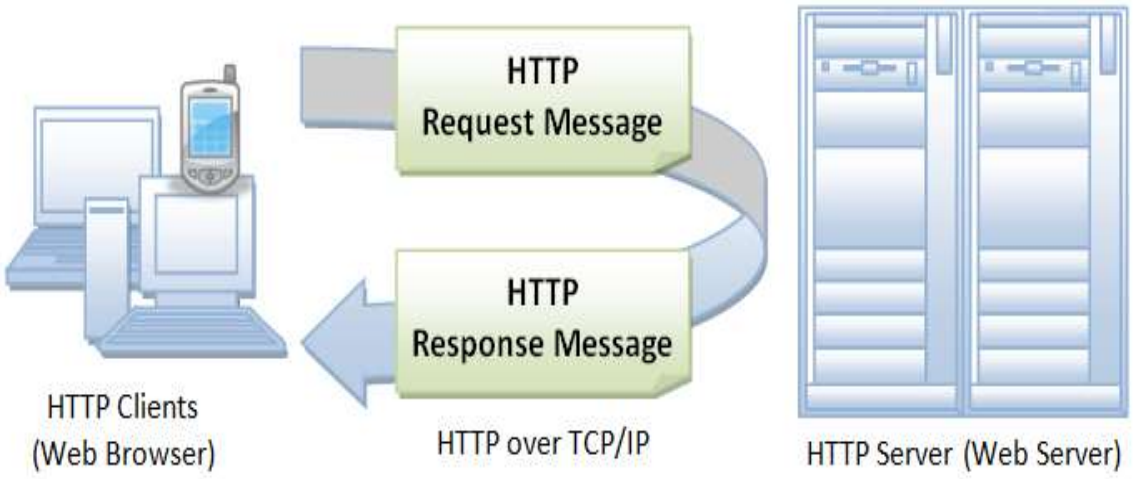
HTTP Message

- Thông điệp HTTP (HTTP Message) là thông tin được trao đổi qua lại giữa Client và Server. Trong đó, HTTP Request Message là thông điệp gửi từ Client đến Server, và ngược lại là HTTP Response Message.



HTTP Message

- Phiên bản HTTP/1.1 của HTTP Message được chuẩn hóa vào năm 1997, đây là phiên bản giao thức HTTP mặc định trong các trình duyệt. Trong khi đó, HTTP/2.0 chuẩn hóa năm 2015 là phiên bản cải tiến tiếp theo. Nếu Server hỗ trợ HTTP/2.0 thì các trình duyệt sẽ tự động sử dụng phiên bản HTTP/2.0.



HTTP Request

- HTTP Request là thông báo yêu cầu từ máy khách gửi đến máy chủ. Nội dung thông báo bao gồm: Request Line, Request Header và Request Body (tùy chọn).

Request Line:

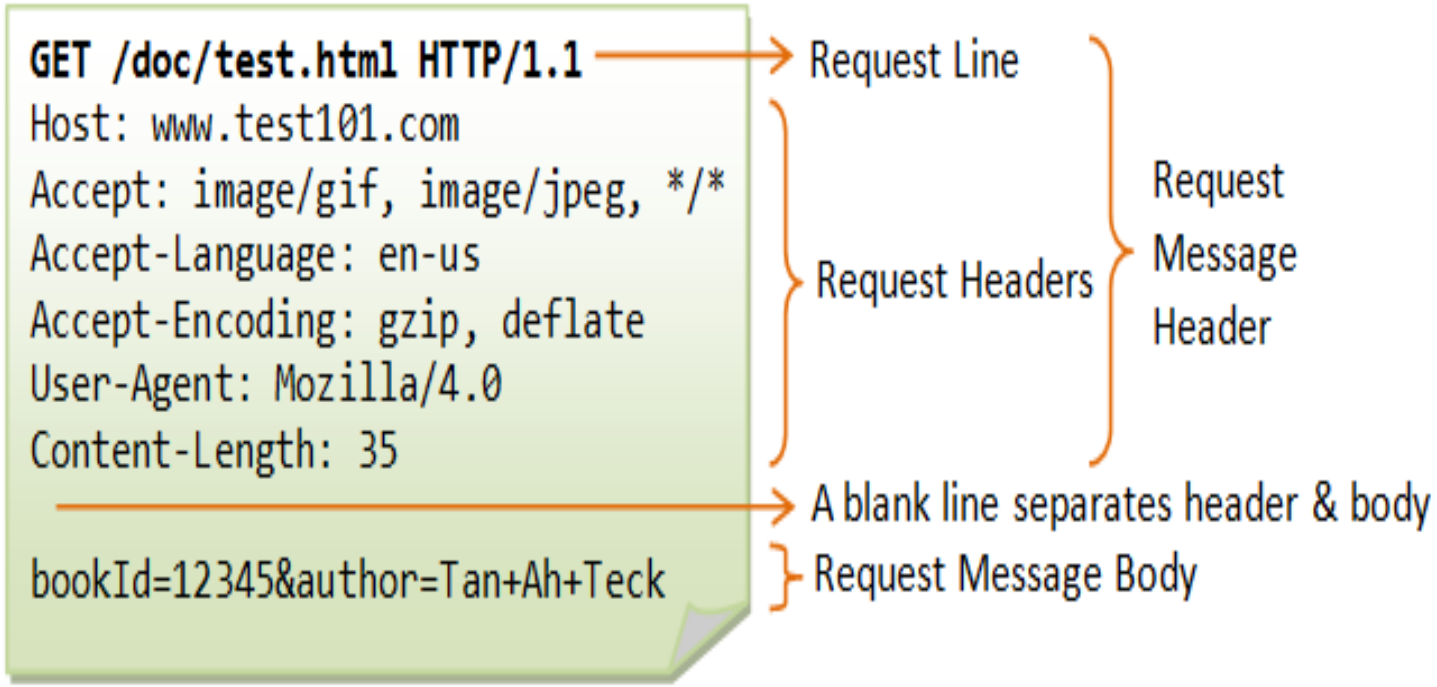
- Tên phương thức: GET, POST, PUT, DELETE
- URL: là địa chỉ định danh tài nguyên
- HTTP version: phiên bản của giao thức HTTP (ví dụ HTTP/1.0, HTTP/1.1)

Request Header:

- Cho phép client gửi thêm các thông tin bổ sung đính kèm với HTTP Request, Một số trường thông dụng như:
  - Accept loại nội dung có thể nhận được từ thông điệp response (ví dụ: text/plain, text/html…)
  - Accept-Encoding: các kiểu nén được chấp nhận (ví dụ gzip, deflate)
  - Connection: tùy chọn điều khiển cho kết nối hiện thời. Ví dụ: keepalive, Upgrade…
  - Cookie: thông tin HTTP Cookie từ server
  - User-Agent: thông tin về user agent của người dùng

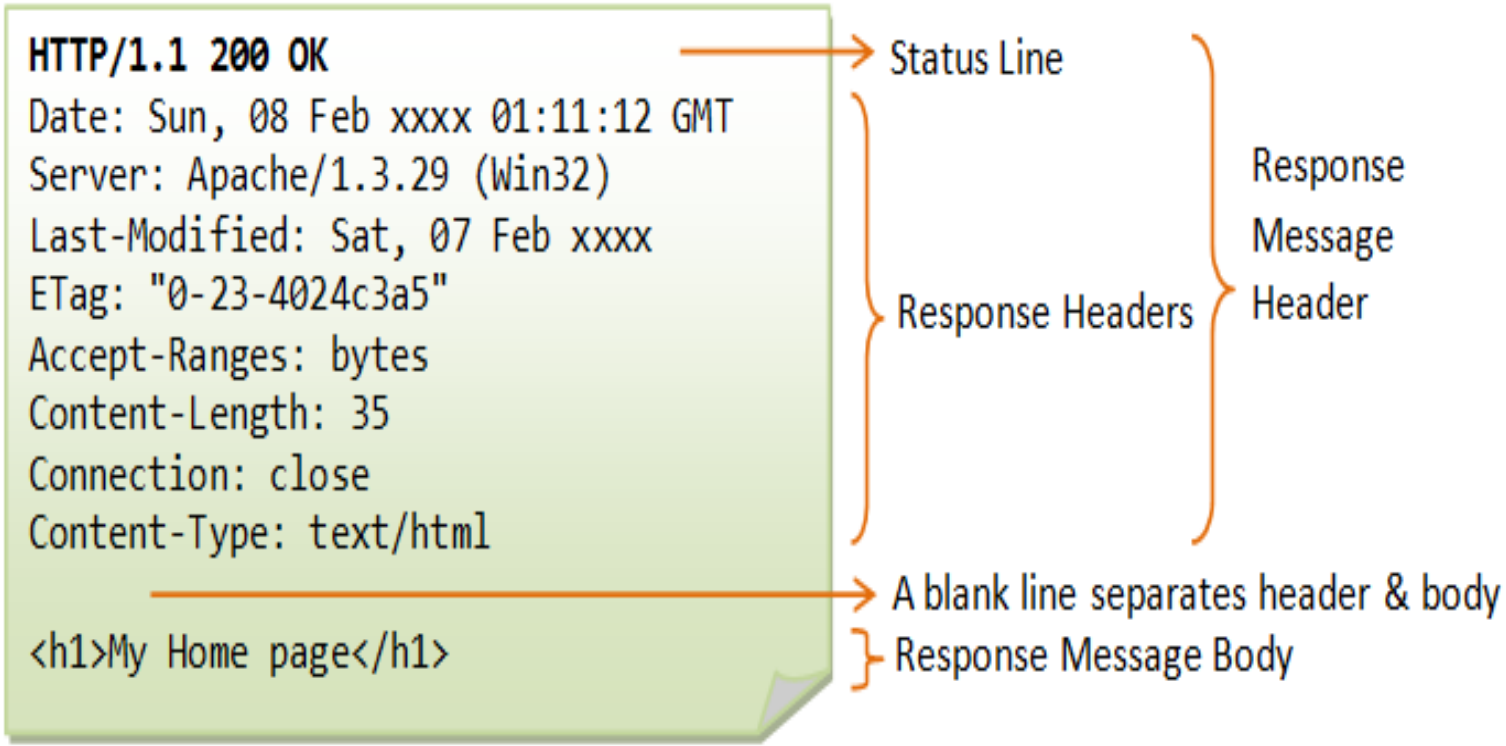
Request Body (tùy chọn):

- Các yêu cầu tìm nạp tài nguyên, như GET, HEAD, DELETE hoặc OPTIONS, thường không cần Request Body.



HTTP Response

- HTTP Response cũng là một HTTP Message, nên nó cũng có cấu trúc tương tự với HTTP Resquest, gồm các thành phần Response Line, Response Header và Response Body.



HTTP Response

- Response Line có cấu trúc như sau:

HTTP-version – Status Code – Reason Phrase

HTTP-version	**Phiên bản HTTP, thông dụng nhất là HTTP/1.1**
Status Code	Mã phản hồi do máy chủ gửi về. Ví dụ: 200, 301, 500
Reason Phrase	Giải thích ngắn gọn về mã phản hồi

Thư viện urllib trong python

- urllib request
- Xử lý ngoại lệ

Thư viện urllib trong python

- Module urllib trong Python 3 cho phép người lập trình để truy cập các trang web. Thông qua urllib, người phát triển ứng dụng có thể truy cập các trang web, tải xuống dữ liệu, phân tích cú pháp dữ liệu, sửa đổi tiêu đề và thực hiện bất kỳ yêu cầu GET và POST nào mà họ có thể cần thực hiện. Dưới đây là bảng liệt kê một số package của module urllib.

Modules	Mô tả
urllib.request	mở và đọc
urllib.parse	phân tích cú pháp URL
urllib.error	cho các trường hợp ngoại lệ được nêu ra
urllib.robotparser	phân tích cú pháp tệp robot.txt

Thư viện urllib trong python

- Cài đặt urllib:
  - pip install urllib

Thư viện urllib trong python

- urllib request:
  - Urlopen:
    - urlopen dùng để mở một đối tượng mạng thông qua chuỗi URL. Trường hợp kết nối thành công, một đối tương giống như tập tin sẽ được trả về, nếu kết nối không thể được thực hiện, ngoại lệ IOError được đưa ra. Đôi tượng trả về hỗ trợ các phương thức sau: read (), readline (), readlines (), fileno (), close (), info (), getcode () và geturl ().

Thư viện urllib trong python

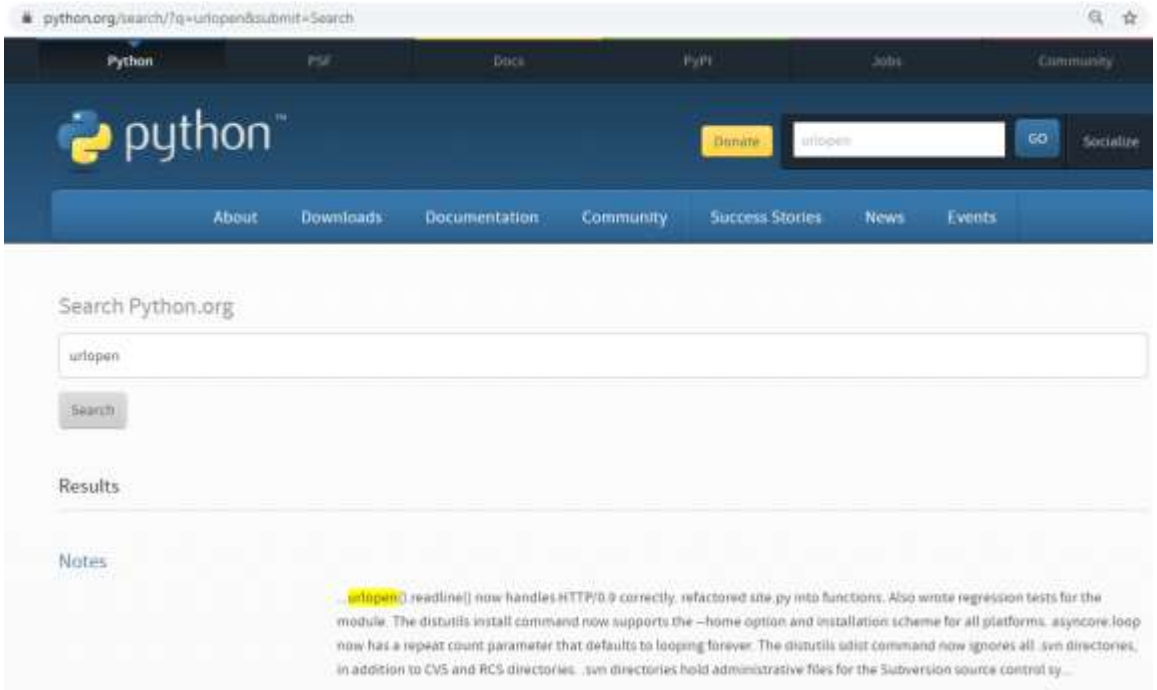
- **Thí dụ 1**:

```
1  #Used to make requests
2  import urllib.request
3
4  url_google = urllib.request.urlopen('https://www.google.com/')
5  print(url_google.read())
```

- **Thí dụ 2**: Sinh viên hãy phát triển trên thí dụ 1 bằng cách sử dụng các hàm info(), getcode() và geturl(), sau đó ghi lại kết quả và giải thích ý nghĩa của từng hàm.

urllib.parse

- Giả sử chúng ta muốn tìm kiếm thông tin ‘basic’ từ trang web www.python.org, chúng ta có thể vào trang đó và gõ từ **openurl** vào trong mục tìm kiếm, rồi nhấn vào nút **Seach,** khi đó địa chỉ trang có dạng là https://www.python.org/search/?q=urlopen&submit=Search (xem hình minh họa dưới đây).



urllib.parse

- Khi đó để lấy được nội dung của kết quả tìm kiếm, ta xem thí dụ 2 dưới đây:
- **Thí dụ 2**:

```
1  #Used to make requests
2  import urllib.request
3
4  url_google = urllib.request.urlopen('https://www.python.org/search/?q=urlopen&submit=Search')
5  print(url_google.read())
```

- Bên cạnh đó, chúng ta có thể sử dụng thư viện **urllib.parse** để thay thế, hãy xem thí dụ 3 dưới đây (kết quả thí dụ 3 hoàn toàn tương tự thí dụ 2)

urllib.parse

- **Thí dụ 3**:

```
1  import urllib.parse
2  url ='https://www.python.org/search'
3  values = {'q':'basic', 'submit':'search'}
4  data = urllib.parse.urlencode(values)
5  data = data.encode('utf-8')
6  req = urllib.request.Request(url, data)
7  resp = urllib.request.urlopen(req)
8  respData = resp.read()
9  print (respData)
```

Xử lý ngoại lệ

- urlopen có thể gây ra ngoại lệ (exception) URLError trong trường hợp máy chủ tại địa chỉ url không phản hồi kết quả. Khi đó, người phát triển ứng dụng nên nắm bắt và xử lý các ngoại lệ này. Xem thí dụ 4 và 5 dưới đây.

Xử lý ngoại lệ

- **Thí dụ 4**:

```
1  import urllib.request
2  try:
3      url_google = urllib.request.urlopen('https://www.huynhhoc.com/')
4      print(url_google.read())
5  except urllib.error.URLError as e:
6      print('Error: ', e.reason)
```

```
Error:  [Errno -2] Name or service not known
```

Xử lý ngoại lệ

- **Thí dụ 5**:

```
1  import urllib.request
2  req = urllib.request.Request('http://www.python.org/fish.html')
3  try:
4      urllib.request.urlopen(req)
5  except urllib.error.HTTPError as e:
6      print(e.code)
```

```
404
```

Rest API và Webservice

- Kiến trúc của REST
- REST APIs and Web Services
- REST và Python
- JSON

Kiến trúc của REST

- REST (representational state tranfer) là một dạng kiến trúc phần mềm xác định kiểu cho giao tiếp giữa client và server thông qua mạng. REST cung cấp một tập các ràng buộc cho kiến trúc phần mềm để thúc đẩy hiệu suất, khả năng mở rộng, tính đơn giản và độ tin cậy trong hệ thống.

Kiến trúc của REST

- REST định nghĩa các ràng buộc sau:
- **Stateless**: Máy chủ không duy trì bất cứ trạng thái nào giữa các yêu cầu từ máy khách.
- **Client-Server**: Máy chủ và máy khách phải phát triển độc lập với nhau.
- **Cacheable**: Dữ liệu được truy xuất từ máy chủ phải được máy khách hoặc máy chủ lưu vào bộ nhớ cache.

Kiến trúc của REST

- REST định nghĩa các ràng buộc sau:
- **Uniform interface**: Máy chủ sẽ cung cấp một giao diện thống nhất để truy cập tài nguyên mà không cần xác định đại diện của chúng.
- **Layered system**: Máy khách có thể truy cập tài nguyên trên máy chủ một cách gián tiếp thông qua các tầng khác nhau như proxy hoặc load balancer.
- **Code on demand (tùy chọn)**: Máy chủ có thể chuyển mã nguồn đến máy khách mà nó có thể chạy, chẳng hạn như JavaScript cho ứng dụng một trang.
- https://en.wikipedia.org/wiki/Load_balancing_(computing)

REST APIs and Web Services

- Webservice REST là bất kỳ webservice nào tuân theo các ràng buộc về kiến trúc REST. Các webservice này hiển thị dữ liệu của họ với thế giới bên ngoài thông qua một API (Application Programming Interface). API là một tập hợp các công cụ cho phép các ứng dụng khác nhau tương tác. Một số tên tuổi lớn nhất trên web (như Reddit, Spotify, Twitter và Facebook) cung cấp các API miễn phí để cung cấp dữ liệu có giá trị trên máy chủ của họ. Học cách làm việc với API sẽ chuẩn bị cho bạn làm việc trong việc truy xuất và phân tích dữ liệu để tạo ra thông tin chi tiết và giúp đưa ra các dự đoán có giá trị.

REST APIs and Web Services

- Các API REST lắng nghe các phương thức HTTP như GET, POST và DELETE để biết các thao tác nào cần thực hiện trên tài nguyên của web service. Tài nguyên là bất kỳ dữ liệu nào có sẵn trong webservice có thể được truy cập và thao tác với các yêu cầu HTTP tới API REST. Phương thức HTTP cho API biết hành động nào cần thực hiện trên tài nguyên.

REST APIs and Web Services

HTTP method	Description
GET	Retrieve an existing resource.
POST	Create a new resource.
PUT	Update an existing resource.
PATCH	Partially update an existing resource.
DELETE	Delete a resource.

- **API Endpoints** API REST hiển thị một tập hợp các URL công khai mà các ứng dụng khách sử dụng để truy cập các tài nguyên của web service. Các URL này, trong ngữ cảnh của một API, được gọi là Endpoint

REST và Python

- Để lập trình tương tác với REST API, những nhà phát triển Python có thể sử dụng requests để gửi HTTP requests. Nó là một thư viện HTTP được cấp phép của Apache2, được viết bằng Python.
- Cài đặt thư viện requests:
  - pip install requests

Phương thức GET

- GET là một trong số các phương thức HTTP phổ biến mà người lập trình sử dụng khi làm việc với REST API. Phương thức này chỉ cho phép truy xuất tài nguyên từ một API cho trước, không cho phép thay đổi hoặc chỉnh sửa tài nguyên. Người lập trình có thể sử dụng dịch vụ giả JSONPlaceholder (https://jsonplaceholder.typicode.com) để kiểm tra GET.

Phương thức GET

- **Thí dụ 6**:

```
1  import requests
2  api_url = "https://jsonplaceholder.typicode.com/todos/1"
3  response = requests.get(api_url)
4  print (response.status_code)
5  print (response.json())
```

```
200
{'userId': 1, 'id': 1, 'title': 'delectus aut autem', 'completed': False}
```

- Đoạn mã này gọi request.get () để gửi yêu cầu GET tới /todos/1, yêu cầu này sẽ phản hồi với mục **todo** với **ID** 1. Trong thí dụ này, dữ liệu trả về có định dạng JSON.
- **Câu hỏi**: làm sao để có thể xem headers của response trả về:

Phương thức POST

- POST là một phương thức khác của HTTP. Phương thức này được dùng để tạo/thêm nguồn tài nguyên mới tại địa chỉ tài nguyên (URI) đã được chỉ định. Trường hợp mã trạng thái (status code) trả về là 201, nghĩa là tài nguyên mới được tạo; trường hợp không đủ thông tin để tạo tài nguyên, giá trị trạng thái sẽ là 400 (Bad Request). Trong trường hợp có xung đột (Conflict), mã trạng thái sẽ có giá trị là 409. Trường hợp mã trạng thái trả về là 200, nghĩa là tài nguyên mới được thêm vào.

Phương thức POST

- Thí dụ 7:

```
1  import requests
2  api_url = "https://jsonplaceholder.typicode.com/todos"
3  todo = {"userId": 1, "title": "Buy milk", "completed": True}
4  response = requests.post(api_url, json=todo)
5  response.json()
```

```
{'completed': True, 'id': 201, 'title': 'Buy milk', 'userId': 1}
```

```
1  print (response.status_code)
```

```
201
```

- Thí dụ trên gọi requests.post() để tạo mới một todo trong hệ thống. Trước tiên, khai báo từ điển todo. Sau đó, gán todo vào tham số json, thao tác này sẽ được requests.post() tự động thiết lập Content-Type của headers là application/json (headers = {“Content-Type”: “application/json”}. Có thể làm theo thao tác thủ công như thí dụ 8 dưới đây:

Phương thức POST

- Thí dụ 8:

```
1  import requests
2  import json
3  api_url = "https://jsonplaceholder.typicode.com/todos"
4  todo = {"userId": 1, "title": "Buy milk", "completed": False}
5  headers =  {"Content-Type":"application/json"}
6  response = requests.post(api_url, data=json.dumps(todo), headers=headers)
7  response.json()
8  print(response.status_code)
```

```
201
```

- **Ghi chú:** json.dumps() là hàm thuộc gói json có trong thư viện chuẩn. Gói thư viện này cung cấp các phương pháp hữu ích để làm việc JSON trong Python.

Các phương thức khác

- Bên cạnh GET và POST, requests còn cung cấp các phương thức khác như PUT, PATCH và DELETE. Sinh viên có thể tự tìm hiểu thêm các phương thức này.

API Key

- Khóa API là một giá trị duy nhất được chỉ định cho người dùng sử dụng các dịch vụ. Bằng cách xem khóa được cung cấp theo yêu cầu, dịch vụ sẽ kiểm tra tính hợp lệ của khóa trước khi cấp hoặc không quyền cho người dùng truy cập sử dụng dịch vụ có liên quan.

API Key

- **Thí dụ 9:**

```
import requests
API_KEY = 'AIzaSyCfKDzIrmmLSyZLq_fHyKzifb_s_4p****'
video_id='7cmvABXyUC0'
url = "https://www.googleapis.com/youtube/v3/videos?id="+video_id+"&part=statistics&key="+API_KEY
response = requests.get(url).json()
print(response)
```

```
{'kind': 'youtube#videoListResponse', 'etag': 'Z_G2wUUuXYqbo4-YnyKPcc0a0do', 'items': [{'kind': 'youtube#vide
o', 'etag': 'LRJI245mh3Fjld2HuS0DjtjqjNA', 'id': '7cmvABXyUC0', 'statistics': {'viewCount': '784', 'likeCount':
'21', 'dislikeCount': '0', 'favoriteCount': '0', 'commentCount': '2'}}], 'pageInfo': {'totalResults': 1, 'resul
tsPerPage': 1}}
```

- **Lưu ý**:
- Sinh viên có thể tham khảo hướng dẫn tại đường link https://blog.hubspot.com/website/how-to-get-youtube-api-key để lấy API_KEY.
- Xem cách gọi API thông qua HTTPBasicAuth tại link https://realpython.com/python-requests/

JSON

- **JSON** (**J**ava**S**cript **O**bject **N**otation) là định dạng dữ liệu theo cặp key – value. Hầu hết các ngôn ngữ lập trình hiện nay đều dễ dàng đọc được loại định dạng này. JSON là một định dạng để gửi và nhận dữ liệu thông qua API. Định dạng này mã hóa các cấu trúc như danh sách (list) và từ điển (dictionary) thành chuỗi để đảm bảo các ứng dụng có thể dễ dàng đọc được.

JSON

- **Thí dụ 10**:
- {'kind': 'Response', 'items': [{'id': '7cmvABXyUC0', 'statistics': {'viewCount': '784', 'commentCount': '2'}}], 'pageInfo': {'totalResults': 1, 'resultsPerPage': 1}}
- Python hỗ trợ xử lý JSON thông qua thư viện json. Thư viện này có hai phương thức chính gồm:
  - dumps: chuyển đối tượng thành chuỗi.
  - loads: chuyển chuỗi JSON thành đối tượng

JSON

- **Thí dụ 11:** Sử dụng JSON để chuyển (dumps) list thành chuỗi và ngược lại (loads)

```
1  best_food_chains =["Taco Bell", "Shake Shack", "Chipotle"]
2  print(type(best_food_chains))
3  #Import the JSON library
4  import json
5  #Use jon.dumps to convert best_food_chains to a string
6  best_food_chains_string = json.dumps(best_food_chains)
7  print(type(best_food_chains_string))
8
9  #Convert best_food_chains_string back to a list
10 best_food_chains_list = json.loads(best_food_chains_string)
11 print (type(best_food_chains_list))
```

```
<class 'list'>
<class 'str'>
<class 'list'>
```

JSON

- **Thí dụ 12:** Sử dụng JSON để chuyển (dumps) từ điển thành chuỗi và ngược lại (loads)

```
#Make a dictionary
fast_food = {
"MacDonalds":14098,
 "Starbucks": 10821,
 "Pizza Hut": 7600
}
print (type(fast_food))
#We can also dump a dictionary to a string and load it
fast_food_string = json.dumps(fast_food)
print(type(fast_food_string))
fast_food_dic = json.loads(fast_food_string)
print (type(fast_food_dic))
```

```
<class 'dict'>
<class 'str'>
<class 'dict'>
```