|  |
| --- |
| BAN CƠ YẾU CHÍNH PHỦ  **HỌC VIỆN KỸ THUẬT MẬT MÃ**  ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯  Logo HvKTMM |
| BÀI TẬP LỚN  **TÌM HIỂU VỀ HTTP PROGRAMMING** |
| Môn: Kỹ thuật lập trình  Lớp: L03  *Nhóm sinh viên thực hiện*:  **Phạm Đức Mạnh**  **Trần Hữu Đức**  **Nguyễn Võ Xuân Hùng**  **Nguyễn Trường Giang Huy**  **Đặng Thu Thảo**  *Giảng viên hướng dẫn*:  **ThS. ĐẶNG XUÂN BẢO**  Khoa An toàn thông tin – Học viện Kỹ thuật mật mã |
| Hà Nội, 2022 |

Mục lục

[Lời cảm ơn](#_Toc106616940)

[Lời nói đầu](#_Toc106616941)

[Chương 1. HTTP PROGRAMMING](#_Toc106616949)

[1.1. **Giới thiệu giao thức HTTP**](#_Toc106616950) ………………………………………………….

[1.1.1. Khái niệm](#_Toc106616951)

[1.1.2. Mã trạng thái .](#_Toc106616952)

[1.2. **Xây dựng một ứng dụng HTTP với http.client** ……………………](#_Toc106616953)………

[1.3. **Xây dựng một ứng dụng HTTP với urllib.request ………………………...**](#_Toc106616954)

[1.3.1. Trích xuất email từ một URL với urllib.request ………………………](#_Toc106616955).

[1.3.2. Tải xuống tệp bằng urllib.request](#_Toc106616956) ………………………………………

[1.3.3. Xử lý các ngoại lệ với urllib.request](#_Toc106616957) ……………………………………

[1.4. **Xây dựng một ứng dụng HTTP với yêu cầu**](#_Toc106616959)………………………………..

[1.4.1. Lấy hình ảnh và liên kết từ một URL với yêu cầu……………………](#_Toc106616960) .

[1.4.2. Đưa ra yêu cầu GET với API REST](#_Toc106616960) ……………………………………

[1.4.3. Đưa ra yêu cầu POST với REST API](#_Toc106616960)…………………………………..

[1.4.4. Quản lý proxy với các yêu cầu](#_Toc106616960) …………………………………………

[1.4.5. Quản lý các trường hợp ngoại lệ với các yêu cầu………………………](#_Toc106616960)

[1.5.](#_Toc106616959) **[Xây dựng một ứng dụng HTTP với http……………………………………](#_Toc106616959)**

[1.6.](#_Toc106616959) **[Cơ chế xác thực với Python ………………………………………………….](#_Toc106616959)**

[Chương 2. Demo](#_Toc106616961)

[2.1. Chuẩn bị](#_Toc106616962)

[2.2. Quy trình thực hiện.](#_Toc106616963)

[Kết luận](#_Toc106616964)

[Tài liệu tham khảo](#_Toc106616965)

Lời cảm ơn

Trong quá trình thực hiện bài tập lớn này, nhóm đã nhận được sự giúp đỡ tận tình của giảng viên hướng dẫn là ThS. Đặng Xuân Bảo – Giảng viên Khoa An toàn thông tin Học viện Kỹ thuật Mật mã. Nhóm cảm ơn thầy và sự cố gắng của các thành viên trong nhóm để hoàn thành bài tập lớn này!

|  |  |
| --- | --- |
|  | **NHÓM SINH VIÊN THỰC HIỆN BÀI TẬP LỚN**  **Phạm Đức Mạnh**  **Trần Hữu Đức**  **Nguyễn Võ Xuân Hùng**  **Nguyễn Trường Giang Huy**  **Đặng Thu Thảo** |

Lời nói đầu

Trong bài tập nghiên cứu này, chúng em đã xem xét http.client, urllib.request, các yêu cầu và Mô-đun httpx để xây dựng máy khách HTTP. Mô-đun yêu cầu là một công cụ rất hữu ích nếu muốn sử dụng các điểm cuối API từ ứng dụng Python của mình. Chúng em đã xem xét các cơ chế xác thực chính và cách triển khai chúng với mô-đun yêu cầu. Điều này sẽ hữu ích cho các nhà phát triển như lúc có nhiều lựa chọn thay thế khác nhau khi cần sử dụng một mô-đun giúp đưa ra yêu cầu đối với dịch vụ web hoặc API REST dễ dàng hơn.

|  |  |
| --- | --- |
|  | **NHÓM SINH VIÊN THỰC HIỆN**  **Phạm Đức Mạnh**  **Trần Hữu Đức**  **Nguyễn Võ Xuân Hùng**  **Nguyễn Trường Giang Huy**  **Đặng Thu Thảo** |
|  |  |

**Chương 1. HTTP PROGRAMMING**

[2.1.](#_Toc106616950) **[Giới thiệu giao thức HTTP](#_Toc106616950)**

[2.1.1. Khái niệm](#_Toc106616951)

HTTP là một giao thức lớp ứng dụng xác định các quy tắc mà máy

khách, proxy và máy chủ cần theo dõi để trao đổi thông tin. Về cơ bản, nó bao gồm hai yếu tố:

• Một yêu cầu được thực hiện bởi máy khách, yêu cầu từ máy chủ một tài nguyên cụ thể được chỉ định bởi một URL.

• Một phản hồi, được gửi bởi máy chủ, cung cấp tài nguyên mà khách hàng yêu cầu.

Giao thức HTTP là một giao thức truyền dữ liệu siêu văn bản không trạng thái không lưu trữ trao đổi thông tin giữa máy khách và máy chủ. Là một giao thức không trạng thái để lưu trữ thông tin liên quan đến giao dịch HTTP, cần sử dụng các kỹ thuật khác để lưu trữ dữ liệu trao đổi, chẳng hạn như cookie (giá trị được lưu trữ ở phía máy khách) hoặc phiên (không gian bộ nhớ tạm thời dành riêng để lưu trữ thông tin về một hoặc nhiều HTTP giao dịch ở phía máy chủ). Các máy chủ trả về một mã HTTP cho biết kết quả của một hoạt động được yêu cầu bởi khách hàng. Ngoài ra, các yêu cầu có thể sử dụng tiêu đề để bao gồm thông tin bổ sung trong cả hai yêu cầu và phản hồi. Điều quan trọng cần lưu ý là giao thức HTTP sử dụng các socket ở mức thấp để thiết lập một kết nối máy khách-máy chủ. Trong Python, chúng ta có khả năng sử dụng mô-đun cấp cao hơn, mà tóm tắt chúng ta từ dịch vụ ổ cắm cấp thấp. Với sự hiểu biết cơ bản này về giao thức HTTP, bây giờ chúng ta sẽ tiến thêm một bước nữa và xây dựng các máy khách HTTP bằng cách sử dụng các thư viện Python khác nhau.

[2.1.2. Mã trạng thái](#_Toc106616952)

Xem lại các mã trạng thái

Mỗi khi một yêu cầu được gửi đến một máy chủ web, nó sẽ nhận và xử lý yêu cầu đó, để sau đó trả về các tài nguyên được yêu cầu cùng với các tiêu đề HTTP. Các mã trạng thái của một. Phản hồi HTTP cho biết liệu một yêu cầu HTTP cụ thể đã được hoàn tất thành công hay chưa.

Chúng ta có thể đọc mã trạng thái của một phản hồi bằng cách sử dụng thuộc tính trạng thái của nó. Giá trị 200 là mã trạng thái HTTP cho chúng ta biết rằng yêu cầu đã thành công:



Mã trạng thái được phân thành các nhóm sau:

• 100: Thông tin

• 200: Thành công

• 300: Chuyển hướng

• 400: Lỗi máy khách

• 500: Lỗi máy chủ

Trong mã loại 300, chúng ta có thể tìm thấy mã chuyển hướng 302, cho biết rằng

một URL nhất định được cung cấp bởi các tiêu đề vị trí đã tạm thời được di chuyển, hướng chúng đến thẳng địa điểm mới. Một mã khác mà chúng ta có thể tìm thấy là 307, được sử dụng làm chuyển hướng nội bộ trong trường hợp trình duyệt phát hiện thấy URL đang sử dụng HTTPS.

Trong phần tiếp theo, chúng ta sẽ xem xét mô-đun http.client, mô-đun này cho phép kiểm tra phản hồi của một trang web hoặc dịch vụ web và là một lựa chọn tốt để triển khai HTTP máy khách cho cả giao thức HTTP và HTTPS.

[2.2. **Xây dựng một ứng dụng HTTP với http.client**](#_Toc106616953)

Python cung cấp một loạt các mô-đun được thiết kế để tạo một máy khách HTTP. Chính của Python các mô-đun thư viện là http.client và urllib.request. Các mô-đun này có các khả năng khác nhau, nhưng chúng hữu ích cho hầu hết các thử nghiệm web. Chúng ta cũng có thể tìm thấy yêu cầu mô-đun cung cấp một số cải tiến so với thư viện tiêu chuẩn.

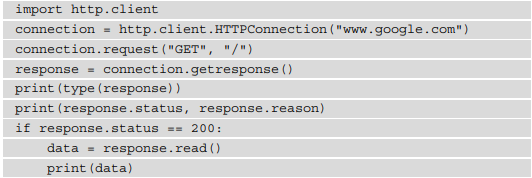
Để biết thêm về những yêu cầu này, hãy truy cập <https://docs.python.org/3/library/http.client.html>.

Vì vậy, chúng ta hãy hiểu mô-đun http.client trước. Mô-đun http.client xác định

một lớp triển khai lớp HTTPConnection. Lớp này chấp nhận một miền và một cổng dưới dạng tham số. Miền là bắt buộc và cổng là tùy chọn. Một ví dụ về điều này lớp đại diện cho một giao dịch với một máy chủ HTTP.

Hãy chứng minh điều này với sự trợ giúp của một ví dụ trong mã. Bạn có thể tìm thấy những điều sau đây

mã trong tệp request\_http\_client.py bên trong thư mục http.client:



Trong đoạn mã trước, chúng ta có thể thấy rằng phương thức getresponse () trả về một thể hiện của lớp http.client.HTTPResponse. Đối tượng phản hồi trả về thông tin

về dữ liệu tài nguyên được yêu cầu cũng như các thuộc tính và siêu dữ liệu phản hồi. Các phương thức read () cho phép đọc dữ liệu tài nguyên được yêu cầu và trả về số byte. Bây giờ chúng ta đã phân tích đối tượng phản hồi, chúng ta sẽ xem xét những gì có thể là giá trị mã trạng thái trong đối tượng đó.

Bây giờ bạn đã biết những điều cơ bản về giao thức HTTP và xây dựng máy khách HTTP với mô-đun http.client, hãy chuyển sang tìm hiểu về cách xây dựng một ứng dụng khách HTTP với mô-đun urllib.request.



Sử dụng chức năng urlopen, một đối tượng tương tự như một tệp được tạo ra để đọc URL. Đối tượng này có các phương thức như read, readline, readlines và close,

hoạt động giống hệt như trong các đối tượng tệp, mặc dù chúng ta thực sự đang làm việc với các phương thức trình bao bọc mà chúng ta sử dụng các ổ cắm cấp thấp để trừu tượng hóa chúng ta.

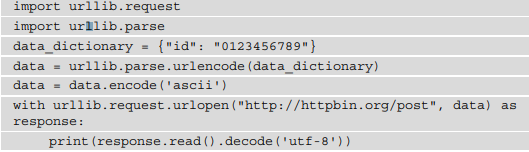
Hàm urlopen cung cấp một tham số dữ liệu tùy chọn để gửi thông tin đến

Địa chỉ HTTP sử dụng phương thức POST, nơi yêu cầu tự gửi các tham số. Đây

tham số là một chuỗi có mã hóa chính xác:



Trong tập lệnh sau, sử dụng phương thức urlopen để thực hiện yêu cầu POST bằng cách sử dụng tham số dữ liệu như một từ điển. Có thể tìm thấy mã sau trong urllib\_post\_request.py tệp bên trong thư mục urllib.request:



Trong đoạn mã trước, ta đang thực hiện yêu cầu POST bằng cách sử dụng từ điển dữ liệu. Sử dụng phương pháp mã hóa qua từ điển dữ liệu do dữ liệu POST cần phải ở trong định dạng byte.

Lấy nội dung của URL là một quá trình đơn giản khi được thực hiện bằng urllib.

Có thể mở trình thông dịch Python và thực hiện các hướng dẫn sau:



Ở đây chúng ta đang sử dụng phương thức urllib.request.urlopen () để gửi một yêu cầu và nhận được phản hồi cho tài nguyên tại http://www.packtpub.com, trong trường hợp này một trang HTML. Sau đó, sẽ in ra dòng đầu tiên của HTML nhận được, với phương thức readline () từ đối tượng phản hồi.

Phương thức urlopen () cũng hỗ trợ đặc tả thời gian chờ cho yêu cầu đại diện cho thời gian chờ đợi trong yêu cầu; nghĩa là, nếu trang chiếm nhiều hơn những gì chúng được chỉ ra, nó sẽ dẫn đến lỗi:



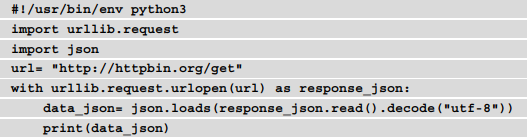
Trong ví dụ trước, chúng ta có thể thấy rằng phương thức urlopen () trả về một phiên bản của lớp http.client.HTTPResponse. Đối tượng phản hồi trả về thông tin cho chúng ta với dữ liệu được yêu cầu và phản hồi:



Nếu ta nhận được phản hồi ở định dạng JSON, có thể sử dụng mô-đun Python json để xử lý

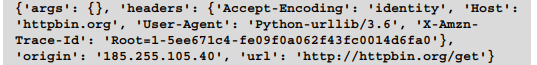


Trong tập lệnh sau, thực hiện một yêu cầu tới một dịch vụ trả về dữ liệu ở định dạng JSON. Bạn có thể tìm thấy mã sau trong tệp json\_response.py bên trong thư mục urllib.request:



Ở đây đang sử dụng một dịch vụ trả về một tài liệu JSON. Để đọc tài liệu này, chúng ta đang sử dụng mô-đun json cung cấp phương thức load (), phương thức này trả về một từ điển của phản hồi json.

Trong đầu ra của tập lệnh trước, chúng ta có thể thấy rằng phản hồi json trả về một từ điển với định dạng key: value cho mỗi tiêu đề:



[2.3. **Xây dựng một ứng dụng HTTP với urllib.request**](#_Toc106616954)

Có hai phần chính đối với các yêu cầu HTTP - tiêu đề và nội dung. Tiêu đề là

dòng thông tin chứa siêu dữ liệu cụ thể về phản hồi và cho khách hàng biết cách

để diễn giải phản hồi. Với mô-đun này, có thể kiểm tra xem tiêu đề có thể cung cấp

thông tin máy chủ web.

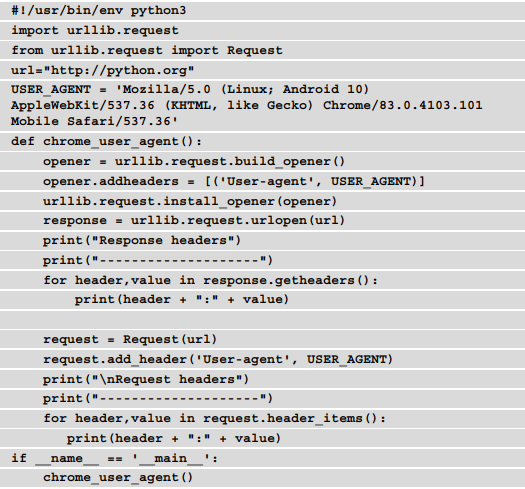
Tiêu đề HTTP chứa thông tin khác nhau về yêu cầu HTTP và ứng dụng khách

bạn đang sử dụng để thực hiện yêu cầu. Ví dụ: Tác nhân người dùng cung cấp thông tin về trình duyệt và hệ điều hành bạn đang sử dụng để thực hiện yêu cầu.

Tập lệnh sau sẽ lấy các tiêu đề của trang web thông qua các tiêu đề của đối tượng phản hồi.

Đối với tác vụ này, chúng ta có thể sử dụng thuộc tính headers hoặc phương thức getheaders (). Các Phương thức getheaders () trả về tiêu đề dưới dạng danh sách các bộ giá trị theo định dạng (tiêu đề tên, giá trị tiêu đề)

Có thể tìm thấy mã sau trong tệp get\_headers\_response\_request.pybên trong thư mục urllib.request:



Trong tập lệnh trước, chúng ta đang tùy chỉnh tiêu đề Tác nhân người dùng với một

phiên bản của trình duyệt Chrome. Để thay đổi Tác nhân người dùng, có hai lựa chọn thay thế. đầu tiên là sử dụng thuộc tính addheaders từ đối tượng opener. Cái thứ hai liên quan đến việc sử dụng phương thức add\_header () từ đối tượng Yêu cầu để thêm tiêu đề tạ đồng thời với việc tạo đối tượng yêu cầu.

Đây là đầu ra của tập lệnh trước đó:



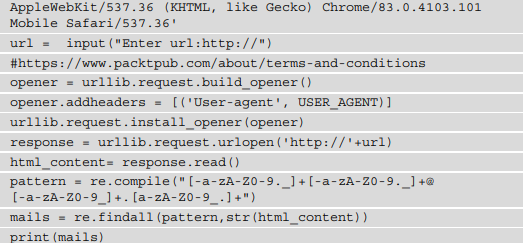
Tại đây, chúng ta có thể thấy việc thực thi tập lệnh trước đó bằng miền python.org, nơi chúng ta có thể thấy các tiêu đề phản hồi và yêu cầu. vừa học cách sử dụng tiêu đề trong gói urllib.request để lấy thông tin về máy chủ web. Tiếp theo, chúng ta sẽ học cách sử dụng gói này để giải nén

email từ các URL.

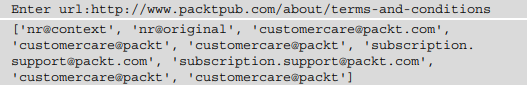
[2.3.1. Trích xuất email từ một URL với urllib.request](#_Toc106616955)

Trong đoạn mã sau, chúng ta có thể thấy cách trích xuất email bằng mô-đun biểu thức chính quy (re) để tìm các phần tử có chứa contain@ trong nội dung được yêu cầu trả về. Bạn có thể tìm thấy mã sau trong tệp get\_emails\_url\_request.py bên trong thư mục urllib.request:





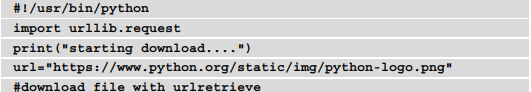
Trong tập lệnh trước, dụng phương thức urllib.request.build\_opener () để tùy chỉnh tiêu đề yêu cầu Tác nhân người dùng. Ta đang sử dụng nội dung HTML được trả về để tìm kiếm các email khớp với cụm từ thông dụng đã xác định:

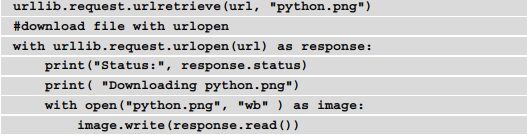


Trong đầu ra trước đó, chúng ta có thể thấy các thư thu được trong quá trình thực thi tập lệnh bằng cách sử dụng miền packtpub.com. Sử dụng phương pháp này, chúng ta có thể nhập URL để giải nén email và tập lệnh sẽ trả về các chuỗi xuất hiện trong mã HTML và khớp với các email trong biểu thức chính quy.

[2.3.2. Tải xuống tệp bằng urllib.request](#_Toc106616956)

Trong tập lệnh sau, chúng ta có thể thấy cách tải xuống tệp bằng cách sử dụng urlretrieve () và các phương thức urlopen (). Bạn có thể tìm thấy mã sau trong download\_file.py tệp bên trong thư mục urllib.request:





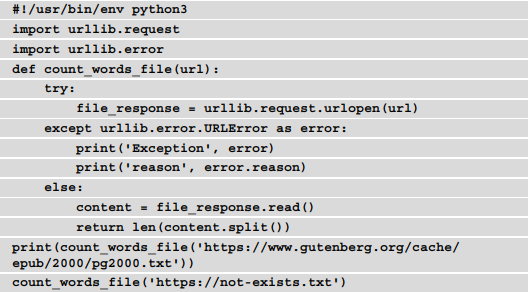
Với giải pháp thay thế đầu tiên, ta đang sử dụng trực tiếp phương thức urlretrieve () và với giải pháp thay thế thứ hai, ta đang sử dụng phản hồi trả về phương thức urlopen ().

[2.3.3. Xử lý các ngoại lệ với urllib.request](#_Toc106616957) v

Mã trạng thái phải luôn được xem xét để nếu có gì sai, hệ thống của chúng ta sẽ trả lời cho thích đáng. Gói urllib giúp kiểm tra mã trạng thái bằng cách nâng một ngoại lệ nếu nó gặp sự cố liên quan đến yêu cầu.

Bây giờ chúng ta hãy xem qua cách bắt những thứ này và xử lý chúng một cách hữu ích. Bạn có thể tìm mã sau trong tệp count\_words\_file.py bên trong urllib.

thư mục yêu cầu:



Ở đây, đang sử dụng mô-đun urllib.request để truy cập tệp internet thông qua URL. Nó cũng hiển thị số lượng từ mà nó chứa.

Phương thức count\_words\_file () nhận URL của tệp văn bản dưới dạng tham số

và trả về số từ mà nó chứa. Nếu URL không tồn tại, thì hãy nâng

ngoại lệ urllib.error.URLError. Đầu ra của tập lệnh trước như sau:



Trong tập lệnh trước đó, lệnh gọi đầu tiên trả về số dòng văn bản và trong lệnh gọi thứ hai, nó tạo ra một ngoại lệ vì URL không đúng.

Với điều này, chúng ta đã hoàn thành phần của trên mô-đun urllib.request. Nhớ lại rằng urllib.request cho phép kiểm tra phản hồi của một trang web hoặc một dịch vụ web và là một tùy chọn tốt để triển khai các máy khách HTTP yêu cầu tùy chỉnh yêu cầu.

Bây giờ bạn đã biết những điều cơ bản về việc xây dựng một ứng dụng khách HTTP với urllib.request, hãy chuyển sang tìm hiểu về cách xây dựng một ứng dụng khách HTTP với các yêu cầu

mô-đun.

[2.4. **Xây dựng một ứng dụng HTTP với yêu cầu**](#_Toc106616959)

Có thể tương tác với các API RESTful dựa trên HTTP ngày càng phổ biến

nhiệm vụ trong các dự án bằng bất kỳ ngôn ngữ lập trình nào. Trong Python, chúng ta cũng có tùy chọn tương tác với API REST theo cách đơn giản với mô-đun yêu cầu. Trong phần này, ta sẽ xem xét các cách khác nhau mà chúng ta có thể tương tác với một API dựa trên HTTP bằng cách sử dụng gói yêu cầu Python.

Một trong những tùy chọn tốt nhất trong hệ sinh thái Python để thực hiện các yêu cầu HTTP là mô-đun yêu cầu. Bạn có thể cài đặt thư viện yêu cầu trong hệ thống của mình trong một cách đơn giản với lệnh pip: yêu cầu cài đặt pip3 Mô-đun này có sẵn trên kho lưu trữ PyPi dưới dạng gói httpx. Nó có thể là được cài đặt thông qua pip hoặc tải xuống từ https://requests.readthedocs.io/vi / master, nơi lưu trữ tài liệu.

Để kiểm tra thư viện trong tập lệnh của chúng ta, chỉ cần nhập nó như ta

làm với các mô-đun khác. Về cơ bản, các yêu cầu là một trình bao bọc của urllib.request, cùng với các mô-đun Python khác để cung cấp cấu trúc REST với các phương thức đơn giản, vì vậy chúng ta có get, post, put, cập nhật, xóa, đầu và các phương pháp tùy chọn, đó là tất cả các phương pháp cần thiết để tương tác với một API RESTful. Mô-đun này có hình thức thực hiện rất đơn giản. Ví dụ, một truy vấn GET sử dụng yêu cầu sẽ như sau:



Như chúng ta có thể thấy, phương thức request.get () đang trả về một đối tượng phản hồi. Trong này

đối tượng, bạn sẽ tìm thấy tất cả các thông tin tương ứng với phản hồi của yêu cầu của chúng ta.

Đây là các thuộc tính chính của đối tượng phản hồi:

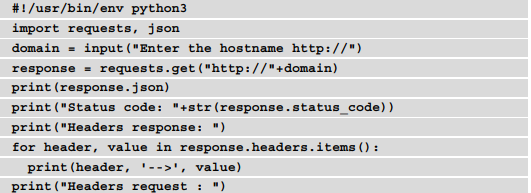
• response.status\_code: Đây là mã HTTP được máy chủ trả về.

• response.content: Tại đây chúng ta sẽ tìm thấy nội dung của phản hồi máy chủ.

• response.json (): Trong trường hợp câu trả lời là JSON, phương thức này tuần tự hóa chuỗi và trả về cấu trúc từ điển với JSON tương ứng kết cấu.

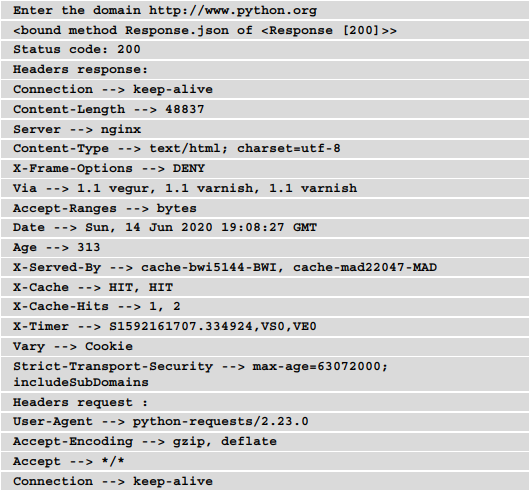
Trong trường hợp không nhận được JSON cho mỗi phản hồi, phương thức sẽ kích hoạt một ngoại lệ.Trong tập lệnh sau, chúng ta cũng có thể xem các thuộc tính thông qua đối tượng phản hồi trong miền python.org. Câu lệnh response.headers cung cấp các tiêu đề của phản hồi của máy chủ web. Về cơ bản, phản hồi là một từ điển đối tượng mà chúng ta có thể lặp lại với định dạng khóa-giá trị bằng phương thức items (). Bạn có thể tìm thấy mã sau trong tệp request\_headers.py bên trong

thư mục yêu cầu:



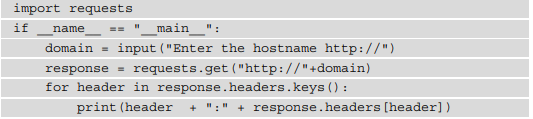


Trong đầu ra của tập lệnh trước, chúng ta có thể thấy tập lệnh đang được thực thi cho python miền tổ chức. Trong dòng cuối cùng của quá trình thực thi, chúng ta có thể đánh dấu sự hiện diện của các truy vấn python trong tiêu đề Tác nhân người dùng:



Theo cách tương tự, chúng ta chỉ có thể lấy các khóa () từ từ điển phản hồi đối tượng.

Bạn có thể tìm thấy mã sau trong tệp request\_headers\_keys.py bên trong thư mục yêu cầu:



Trong số những ưu điểm chính của mô-đun yêu cầu, chúng ta có thể quan sát những điều sau:

• Đây là một mô-đun tập trung vào việc tạo ra các máy khách HTTP đầy đủ chức năng.

• Nó hỗ trợ tất cả các phương pháp và tính năng được xác định trong giao thức HTTP.

• Nó là "Pythonic", tức là nó được viết hoàn toàn bằng Python và tất cả các thao tác đều được thực hiện một cách đơn giản và chỉ với một vài dòng mã.

• Các nhiệm vụ của nó bao gồm tích hợp với các dịch vụ web, tổng hợp các kết nối HTTP, mã hóa dữ liệu POST trong các biểu mẫu và xử lý cookie. Tất cả các tính năng này là được xử lý tự động bằng cách sử dụng các yêu cầu.

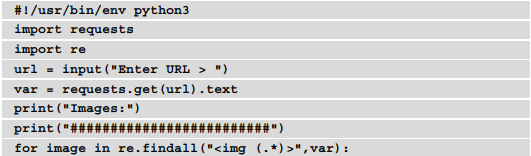
Bây giờ, chúng ta hãy xem với sự trợ giúp của một ví dụ làm thế nào chúng ta có thể lấy hình ảnh và liên kết từ một URL với mô-đun yêu cầu.

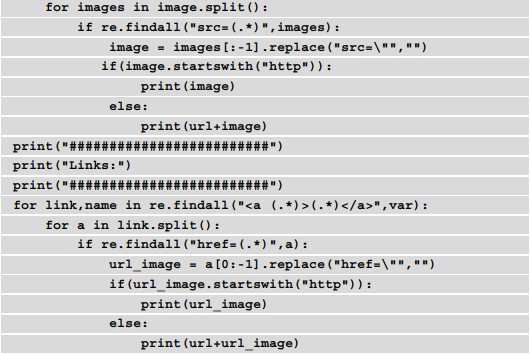
[2.4. **Xây dựng một ứng dụng HTTP với yêu cầu**](#_Toc106616959)

[2.4.1. Lấy hình ảnh và liên kết từ một URL với yêu cầu](#_Toc106616960)

Trong ví dụ sau, chúng ta sẽ trích xuất hình ảnh và liên kết bằng cách sử dụng các yêu cầu và mô-đun biểu thức chính quy. Cách dễ dàng để trích xuất hình ảnh từ một URL là sử dụng để tìm các phần tử HTML img và href trong URL mục tiêu.

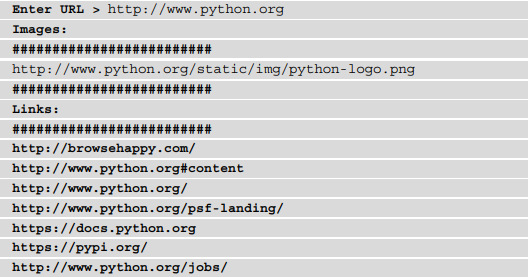
Bạn có thể tìm thấy mã sau trong tệp get\_images\_links\_url.py bên trong thư mục yêu cầu:

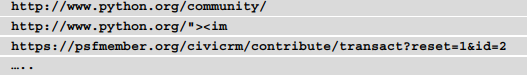




Trong tập lệnh trước, chúng ta đang sử dụng biểu thức chính quy để phát hiện hình ảnh và liên kết.

Trong cả hai trường hợp, sử dụng phương thức findall () từ mô-đun re. Đầu tiên, chúng tôi trích xuất hình ảnh bằng cách phát hiện các phần tử img và sau đó chúng ta trích xuất các liên kết bằng cách phát hiện các phần tử href:





Khi bạn thực thi tập lệnh trước đó, bạn sẽ thấy đầu ra với các hình ảnh và liên kết

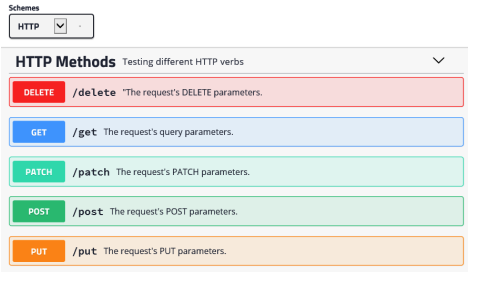
được trích xuất từ miền bạn đã nhập, như có thể thấy ở trên.

Cách này để trích xuất hình ảnh và liên kết từ một trang web có thể áp dụng cho

trích xuất bất kỳ phần tử HTML nào khác, bằng cách xác định biểu thức chính quy cho phần tử điều đó có thể khiến chúng tôi quan tâm.

[2.4.2. Đưa ra yêu cầu GET với API REST](#_Toc106616960)

Để kiểm tra các yêu cầu với mô-đun này, chúng ta có thể sử dụng dịch vụ <https://httpbin.org/> và thử các yêu cầu này, thực hiện từng loại riêng biệt. Trong mọi trường hợp, mã để thực thi để có được đầu ra mong muốn sẽ giống nhau; điều duy nhất sẽ thay đổi sẽ là loại yêu cầu và dữ liệu được gửi đến máy chủ:



Hinh 3.1 – REST API and HTTP methods in the httpbin service

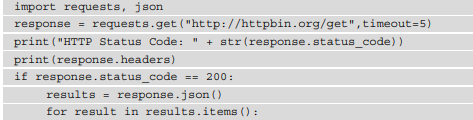
Nếu yêu cầu http://httpbin.org/get URL sẽ nhận được phản hồi trong Định dạng JSON:

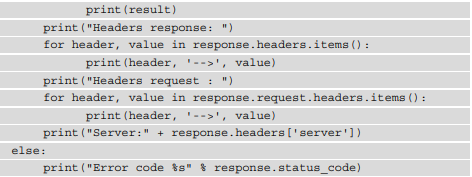


Trong đầu ra trước đó, chúng ta có thể thấy phản hồi ở định dạng JSON cho điểm cuối nhận có sẵn trong dịch vụ httpbin.org.

Bạn có thể tìm thấy mã sau trong tệp testing\_api\_rest\_get\_method.py

bên trong thư mục yêu cầu:





Khi thực thi mã trước đó, sẽ thấy đầu ra với các tiêu đề thu được

cho một yêu cầu và phản hồi. Phản hồi tiêu đề sẽ tương tự như kết quả thu được ở định dạng JSON. Với các yêu cầu GET, chúng ta có thể xác nhận một cách dễ dàng rằng dịch vụ đang chạy và trả về một phản hồi hợp lệ.

[2.4.3. Đưa ra yêu cầu POST với REST API](#_Toc106616960)

Không giống như phương thức GET gửi dữ liệu trong URL, phương thức POST cho phép gửi dữ liệu đến máy chủ trong phần thân yêu cầu.

Ví dụ: giả sử có dịch vụ đăng ký người dùng bằng biểu mẫu mà bạn phải chuyển một ID và email. Thông tin này sẽ được chuyển qua thuộc tính data thông qua một cấu trúc từ điển. Phương thức POST yêu cầu một trường bổ sung được gọi là dữ liệu, trong mà gửi một từ điển với tất cả các yếu tố mà chúng tôi sẽ gửi đến máy chủ thông qua phương pháp tương ứng.

Trong ví dụ này, chúng tôi sẽ mô phỏng việc gửi một biểu mẫu HTML thông qua một POST yêu cầu, giống như các trình duyệt làm khi chúng tôi gửi biểu mẫu đến một trang web. Dữ liệu biểu mẫu luôn được gửi ở định dạng từ điển khóa-giá trị.

Phương thức POST có sẵn trong https://httpbin.org/#/HTTP\_Methods/

dịch vụ post\_post:

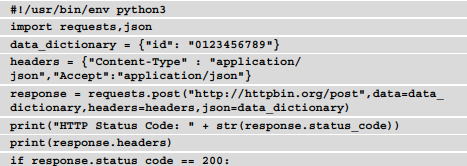


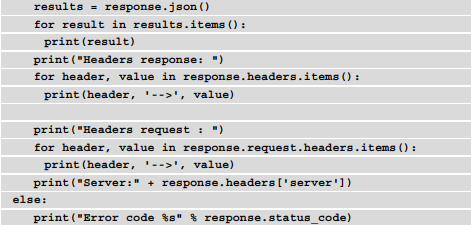
Ngoài ra, chúng tôi đang sử dụng một tiêu đề cụ thể để gửi thông tin đến máy chủ ở định dạng JSON.

Trong trường hợp này, chúng tôi có thể thêm tiêu đề của riêng mình hoặc sửa đổi những tiêu đề hiện có với tiêu đề tham số.

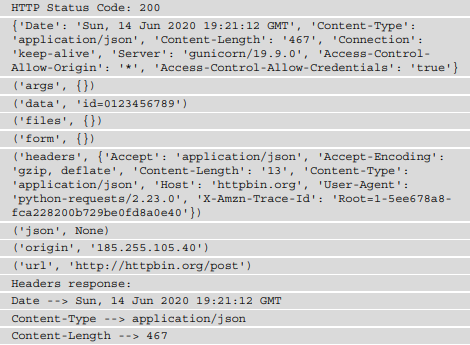
Bạn có thể tìm thấy mã sau trong tệp testing\_api\_rest\_post\_method.py

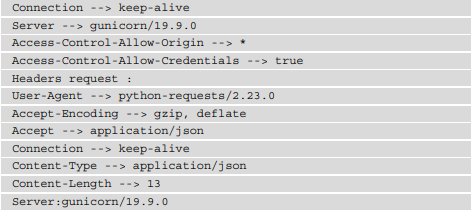
bên trong thư mục yêu cầu:





Trong đoạn mã trước, ngoài việc sử dụng phương thức POST, chúng tôi đang chuyển dữ liệu bạn muốn gửi đến máy chủ dưới dạng tham số trong thuộc tính dữ liệu. Khi bạn chạy tập lệnh trước, bạn sẽ nhận được kết quả sau:





Trong đầu ra của tập lệnh trước, chúng ta có thể thấy rằng đối tượng phản hồi có chứa

ID đang được gửi trong đối tượng từ điển dữ liệu. Ngoài ra, chúng ta có thể thấy các tiêu đề liên quan đến

loại nội dung ứng dụng / json và tiêu đề tác nhân người dùng nơi chúng tôi có thể xem tiêu đề được thiết lập trong giá trị python-request / 2.23 tương ứng với phiên bản của mô-đun yêu cầu mà chúng tôi đang sử dụng.

[2.4.4. Quản lý proxy với các yêu cầu](#_Toc106616960)

Một tính năng thú vị được cung cấp bởi mô-đun yêu cầu là tùy chọn để đưa ra yêu cầu thông qua proxy hoặc máy trung gian giữa mạng nội bộ của chúng tôi và mạng bên ngoài mạng. Một proxy được xác định theo cách sau:



Để thực hiện yêu cầu thông qua proxy, chúng tôi đang sử dụng thuộc tính proxy của get ()phương pháp:



Tham số proxy phải được chuyển dưới dạng từ điển, nghĩa là bạn phải

tạo một loại từ điển trong đó chúng tôi chỉ định giao thức với địa chỉ IP và cổng

nơi proxy đang nghe:



Mã trước có thể hữu ích trong trường hợp chúng tôi cần thực hiện yêu cầu từ nội bộ mạng thông qua một máy trung gian. Đối với điều này, nó là cần thiết để biết địa chỉ IP và cổng của máy này.

[2.4.5. Quản lý các trường hợp ngoại lệ với các yêu cầu](#_Toc106616960)

So với các mô-đun khác, mô-đun yêu cầu xử lý lỗi theo một cách khác.

Trong ví dụ sau, chúng ta thấy cách mô-đun yêu cầu tạo ra lỗi 404,

chỉ ra rằng nó không thể tìm thấy tài nguyên được yêu cầu:

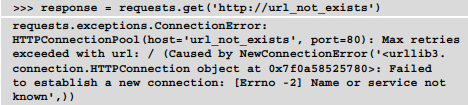


Để xem ngoại lệ được tạo bên trong, chúng ta có thể sử dụng raise\_for\_status ()

phương pháp:



Trong trường hợp đưa ra yêu cầu đối với máy chủ lưu trữ không tồn tại và khi hết thời gian có được tạo ra, chúng tôi nhận được một ngoại lệ ConnectionError:



Với điều này, chúng ta đã đến cuối phần của chúng ta về mô-đun yêu cầu. Như bạn có thể hiện đã nhận thấy, mô-đun yêu cầu giúp việc sử dụng các yêu cầu HTTP dễ dàng hơn trong Python so với urllib. Trừ khi bạn có yêu cầu sử dụng urllib, tôi sẽ

khuyên bạn nên sử dụng các yêu cầu cho các dự án của bạn bằng Python.

Bây giờ bạn đã biết những điều cơ bản về việc xây dựng một ứng dụng khách HTTP với mô-đun yêu cầu, hãy chuyển sang tìm hiểu về cách xây dựng một ứng dụng khách HTTP với mô-đun httpx cho quản lý các yêu cầu không đồng bộ.

[2.5.](#_Toc106616959) **[Xây dựng một ứng dụng HTTP với httpx](#_Toc106616959)**

Gói httpx là gói thư viện chuẩn Python được khuyến nghị cho HTTP và

các tác vụ không đồng bộ trong Python 3.7. Mô-đun này có giao diện đơn giản hơn và nó cũng có năng lực quản lý tất cả các nhiệm vụ liên quan đến yêu cầu không đồng bộ. Mô-đun này tương thích với các yêu cầu và với phiên bản của giao thức HTTP / 2, cung cấp một loạt các cải tiến ở cấp hiệu suất, chẳng hạn như nén

các tiêu đề được gửi trong các yêu cầu. Mô-đun này hỗ trợ cả hai phiên bản HTTP / 1.1 và HTTP / 2. Sự khác biệt chính giữa hai phiên bản này là phiên bản HTTP / 2 là một giao thức dựa trên dữ liệu nhị phân thay vì

dữ liệu dạng văn bản. HTTP / 2 là một phiên bản mới lớn của giao thức HTTP, mang lại hiệu quả hơn nhiềun vận chuyển với những lợi thế về hiệu suất có thể. HTTP / 2 không thay đổi cốt lõi ngữ nghĩa của yêu cầu hoặc phản hồi, nhưng không thay đổi cách dữ liệu được truyền từ và tới máy chủ. Bạn có thể cài đặt mô-đun httpx trên hệ thống của mình một cách dễ dàng bằng lệnh pip



Nếu bạn đang sử dụng Python 3.7, bạn có thể sử dụng lệnh sau:



Mô-đun này có sẵn trên kho lưu trữ PyPi dưới dạng gói yêu cầu. Nó có thể

được cài đặt thông qua Pip hoặc tải xuống từ https://www.python-httpx.org,

nơi bạn có thể tìm thấy tài liệu.

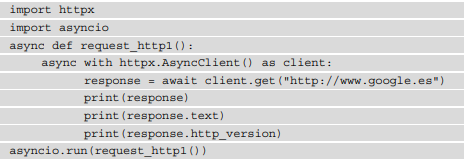
có thể tìm thấy mã sau trong tệp httpx\_basic.py bên trong thư mục httpx:



Để hỗ trợ lập trình không đồng bộ, chúng tôi có thể sử dụng mô-đun asyncio cho phép chúng tôi

để thực hiện nhiều yêu cầu song song mà không chặn các thao tác còn lại.

Bạn có thể tìm thấy mã sau trong tệp httpx\_asyncio.py bên trong thư mục httpx:



Hỗ trợ HTTP / 2 không được bật theo mặc định khi sử dụng máy khách httpx vì

HTTP / 1.1 là một lớp truyền tải trưởng thành, có khả năng chiến đấu và HTTP / 1.1 của chúng tai tại thời điểm này, việc triển khai có thể được coi là lựa chọn mạnh mẽ hơn.

Nếu chúng tôi muốn bật hỗ trợ HTTP / 2, chúng tôi có thể sử dụng tham số http2 = True đểbật hỗ trợ HTTP / 2 trên máy khách. Bạn có thể tìm thấy mã sau trong httpx\_

tệp asyncio\_http2.py bên trong thư mục httpx:



Để thực thi tập lệnh trước đó, chúng tôi cần cài đặt phần mở rộng http2 bằng cách sử dụng lệnh sau:



Khi thực thi tập lệnh trước đó, đầu ra sẽ cho biết rằng bạn đang sử dụng

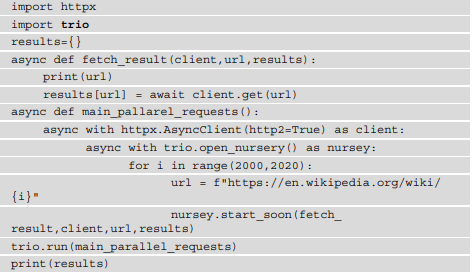
Phiên bản HTTP / 2, cho biết rằng bạn có thể xử lý nhiều yêu cầu đồng thời từ

một luồng TCP.Chúng tôi có các lựa chọn thay thế để quản lý các yêu cầu theo cách không đồng bộ. Trong những điều sau đây

ví dụ, chúng tôi đang sử dụng mô-đun bộ ba thay vì asyncio để thực hiện các tác vụ song song.

Bạn có thể tìm thấy mã sau trong tệp httpx\_http2\_trio.py bên trong httpx

thư mục



Ở đây chúng tôi đang sử dụng mô-đun bộ ba với mẫu async-await, nơi chúng tôi có thể đánh dấu sự hiện diện của phương thức open\_nursery (), phương thức này cung cấp một phương thức khác cách tiếp cận để lập trình đồng thời. Cách tiếp cận này dựa trên mỗi lần gọi đến nhà trẻ. start\_soon () thêm một tác vụ khác chạy song song.

Bạn có thể tham khảo thêm thông tin về mẫu này trong tài liệu về bộ ba: https: //

triple.readthedocs.io/en/stable/tutorial.html#warning-don-tforget-that-await.

Như chúng ta đã thấy trong phần này, mô-đun httpx giúp quản lý dễ dàng hơn

yêu cầu không đồng bộ kết hợp với asyncio và cả hai đều được khuyến nghị

mô-đun cho nhiệm vụ này.

Bây giờ bạn đã biết những điều cơ bản về việc xây dựng một ứng dụng khách HTTP với mô-đun httpx,chúng ta hãy chuyển sang tìm hiểu về các cơ chế xác thực HTTP và cách chúng được triển khai bằng Python.

[2.6.](#_Toc106616959) **[Cơ chế xác thựcvới Python ………………………………………………….](#_Toc106616959)**

Hầu hết các dịch vụ web mà chúng ta sử dụng ngày nay đều yêu cầu một số cơ chế xác thực trong

để đảm bảo rằng thông tin đăng nhập của người dùng hợp lệ để truy cập nó. Trong phần này, chúng ta sẽ tìm hiểu

cách triển khai xác thực trong Python.

Giao thức HTTP nguyên bản hỗ trợ ba cơ chế xác thực:

• Xác thực cơ bản HTTP: Base64 dựa trên xác thực cơ bản HTTP

cơ chế mã hóa người dùng được tạo bằng mật khẩu sử dụng người dùng định dạng:

mật khẩu mở khóa.

• Xác thực thông báo HTTP: Cơ chế này sử dụng MD5 để mã hóa người dùng, khóa,

và băm cảnh giới.

• Xác thực mang HTTP: Cơ chế này sử dụng xác thực dựa trên

trên access\_token. Một trong những giao thức phổ biến nhất sử dụng loại

xác thực là OAuth. Trong URL sau, chúng ta có thể tìm thấy các Python khác nhau

thư viện được hỗ trợ bởi giao thức này: https://oauth.net/code/python/

Python hỗ trợ cả hai cơ chế thông qua mô-đun yêu cầu. Tuy nhiên, chính

sự khác biệt giữa cả hai phương pháp là chỉ mã hóa cơ bản mà không thực sự mã hóa,

trong khi thông báo mã hóa thông tin của người dùng ở định dạng MD5.

Chúng ta hãy hiểu các cơ chế này chi tiết hơn trong các phần phụ sắp tới.

Xác thực cơ bản HTTP với mô-đun yêu cầu

HTTP cơ bản là một cơ chế đơn giản cho phép bạn triển khai xác thực cơ bản

qua tài nguyên HTTP. Ưu điểm chính là dễ dàng triển khai nó trong web Apache

máy chủ, sử dụng các chỉ thị Apache tiêu chuẩn và tiện ích httpasswd.

Vấn đề với phương pháp này là có thể dễ dàng trích xuất thông tin xác thực từ người dùng với

Trình đánh giá Wireshark vì thông tin được gửi ở dạng văn bản thuần túy. Đối với kẻ tấn công, nó có thể

dễ dàng giải mã thông tin ở định dạng Base64. Nếu khách hàng biết rằng một tài nguyên là

được bảo vệ bằng cơ chế này, thông tin đăng nhập và mật khẩu có thể được gửi bằng mã hóa cơ sở trong

tiêu đề Ủy quyền.

Xác thực quyền truy cập cơ bản giả định tên người dùng và mật khẩu sẽ xác định ứng dụng khách.

Khi máy khách trình duyệt truy cập lần đầu tiên vào một trang web bằng cách sử dụng xác thực này, máy chủ sẽ phản hồi

với phản hồi loại 401, chứa thẻ WWW-Authenticate, giá trị Cơ bản,

và tên miền được bảo vệ.

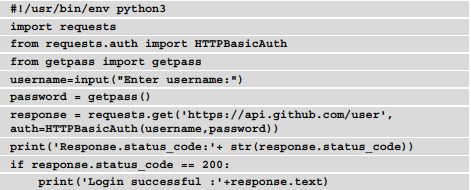
Giả sử rằng chúng tôi có một URL được bảo vệ bằng loại xác thực này, chúng tôi có thể sử dụng

Lớp HTTPBasicAuth từ mô-đun yêu cầu.

Trong tập lệnh sau, chúng tôi đang sử dụng lớp này để cung cấp thông tin đăng nhập người dùng dưới dạng một bộ tuple.

Bạn có thể tìm thấy mã sau trong tệp basic\_authentication.py bên trong

thư mục yêu cầu:



Ở đây, chúng tôi đang sử dụng HTTPBasicAuth để xác thực trong dịch vụ GitHub bằng cách sử dụng

tên người dùng và mật khẩu do người dùng nhập. Nếu đăng nhập thành công, nó sẽ trả về

thông tin về người dùng trong dịch vụ GitHub và các URL liên quan đến API GitHub

người dùng có thể truy cập.

Xác thực thông báo HTTP với mô-đun yêu cầu

Thông báo HTTP là một cơ chế được sử dụng trong giao thức HTTP để cải thiện

quá trình xác thực. MD5 thường được sử dụng để mã hóa thông tin người dùng, cũng như

khóa và miền, mặc dù các thuật toán khác, chẳng hạn như SHA, cũng có thể được sử dụng để cải thiện

bảo mật trong các biến thể khác nhau của nó.

Xác thực truy cập dựa trên số hóa mở rộng xác thực truy cập cơ bản bằng cách sử dụng một chiều

thuật toán mã hóa băm (MD5) để mã hóa thông tin xác thực đầu tiên và

sau đó thêm một giá trị kết nối duy nhất.

Trình duyệt máy khách sử dụng giá trị này khi tính toán phản hồi mật khẩu ở định dạng băm.

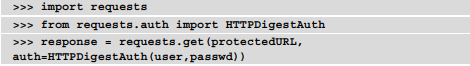
Mặc dù mật khẩu bị xáo trộn do sử dụng hàm băm mật mã và việc sử dụng

giá trị duy nhất ngăn không cho một cuộc tấn công phát lại bị đe dọa, tên đăng nhập được gửi đến

văn bản thuần túy đến máy chủ.

Giả sử chúng tôi có một URL được bảo vệ bằng loại xác thực này, chúng tôi có thể sử dụng

HTTPDigestAuth, có sẵn trong mô-đun con request.auth, như sau:

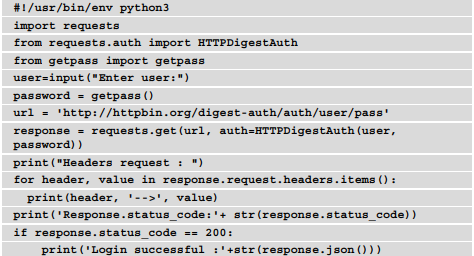


Trong tập lệnh sau, chúng tôi đang sử dụng dịch vụ xác thực, http://httpbin.org/

dig-auth / auth / user / pass, để kiểm tra xác thực thông báo để truy cập

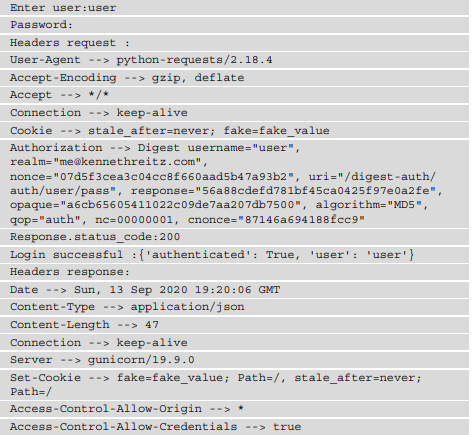
xác thực thông báo tài nguyên được bảo vệ. Tập lệnh tương tự như tập lệnh trước đó với xác thực cơ bản. Sự khác biệt chính là phần mà chúng tôi gửi tên người dùng và

mật khẩu trên URL được bảo vệ. Bạn có thể tìm thấy mã sau trong tệp dig\_authentication.py bên trong thư mục yêu cầu:



Trong tập lệnh trước,ta đang sử dụng dịch vụ httpbin để trình bày cách sử dụng Lớp HTTPDigestAuth để chuyển thông số người dùng và mật khẩu.

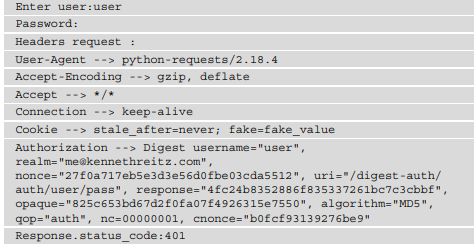
Nếu thực thi tập lệnh trước đó giới thiệu người dùng và mật khẩu chính xác, nhận được sau đầu ra với mã trạng thái 200, nơi chúng ta có thể thấy chuỗi JSON được liên kết với đăng nhập thành công:

****

Trong đầu ra trước, chúng ta có thể thấy cách, trong tiêu đề Ủy quyền, một yêu cầu

gửi thông tin liên quan đến thông báo và thuật toán đang được sử dụng.

Nếu chúng giới thiệu một người dùng hoặc mật khẩu không chính xác, sẽ nhận được kết quả sau với 401 mã trạng thái**:**

****

Trong phần này, chúng ta đã xem xét cách mô-đun yêu cầu hỗ trợ tốt cho cả hai

các cơ chế xác thực.

[Chương 3. Demo](#_Toc106616961)

[3.1. Chuẩn bị](#_Toc106616962)

[3.2. Quy trình thực hiện.](#_Toc106616963)

[Kết luận](#_Toc106616964)

[Tài liệu tham khảo](#_Toc106616965)