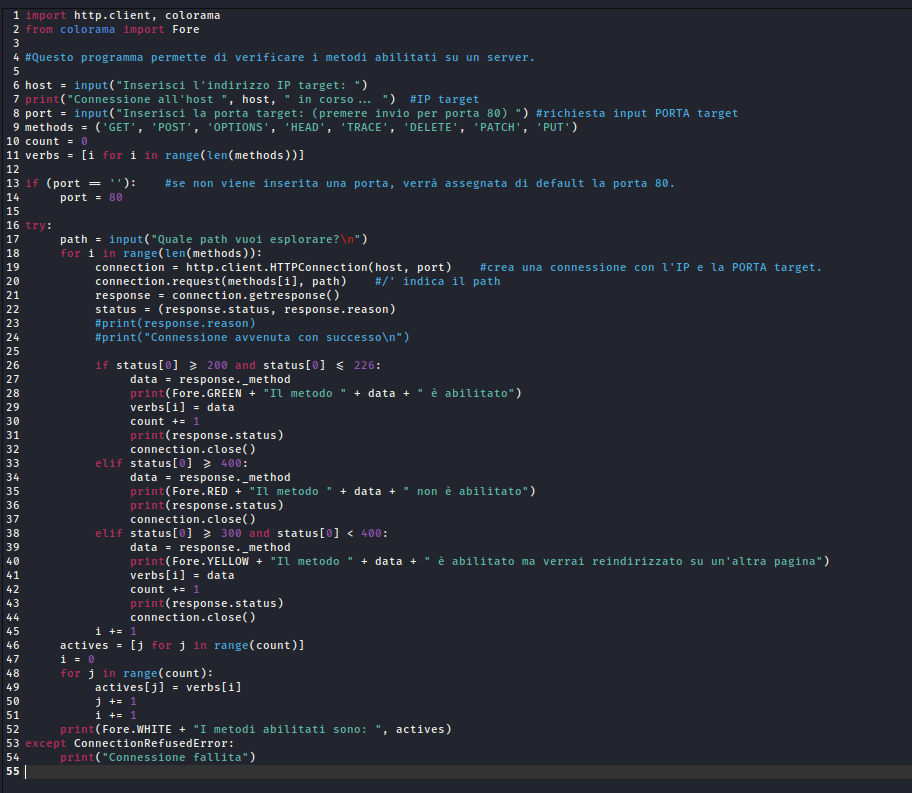
**RILEVATORE VERBI HTTP**

***Sul web server:***

*● Eventuale enumerazione dei metodi HTTP abilitati sul servizio HTTP in ascolto sulla porta 80.*

***Sull'application server:***

*● Enumerazione dei metodi HTTP abilitati.*



Il codice python scritto intercetta i **metodi HTTP** abilitati leggendo **indirizzo IP** e **PORTA** target della VM Metasploitable 2.

L’utente dovrà scegliere una porta aperta su cui è attivo un server HTTP in ascolto; se non effettuerà alcuna scelta il codice lo indirizzerà di default sulla porta 80.

Chiediamo all’utente il path interessato, permettendogli di verificarne lo stato dei metodi.

L’ host instaurerà la connessione col web server Metasploitable 2 ottenendo in risposta un codice di stato indicante l’esito della richiesta di connessione.

I codici di stato HTTP vanno letti secondo la seguente tabella:

* 2xx: **Successful** (la richiesta è stata soddisfatta)
* 3xx: **Redirection** (non c'è risposta immediata, ma la richiesta è sensata e viene detto come ottenere la risposta)
* 4xx: **Client error** (la richiesta non può essere soddisfatta perché sbagliata)
* 5xx: **Server error** (la richiesta non può essere soddisfatta per un problema interno del server)

Inoltre, per avere maggiori indicazioni sul codice di stato ricevuto, viene richiesta la specifica “reason” che lo descrive brevemente.

*Es.:*

*Codice : Reason*

* 200 : OK
* 302: MOVED TEMPORARILY
* 405: METHOD NOT ALLOWED

Abbiamo in precedenza creato una *tupla methods* contenente tutti i verbi HTTP disponibili.

*Metodo* *Descrizione*

**GET** Recupera una risorsa dal server (ad es. visitando una pagina)

**POST** Invia una risorsa al server (ad. es compilando un modulo)

**DELETE** Cancella una risorsa dal server (ad es. eliminando un file)

**PUT** Memorizza una risorsa sul server (ad es. caricando un file)

**HEAD** Recupera solo l'header della risposta senza la risorsa

**OPTIONS\*** Richiede quali metodi supporta il server per un path specifico

**PATCH** Modifica una risorsa nel server(ad es. aggiornando un file)

**TRACE** Permette di ricostruire il percorso di una richiesta HTTP

**CONNECT\*\*** Inizia una comunicazione *two-way* con la risorsa richiesta

***\* = nel nostro caso non utilizzato perché il server nasconde la risposta.***

***\*\*= nel nostro caso non utilizzato perché incompatibile con il codice.***

A questo punto andiamo a eseguire una richiesta per ogni singolo metodo, all’interno di un ciclo **for**, impostando il numero di tentativi in base alla lunghezza della tupla.

*(Si sceglie di ricorrere all’utilizzo della tupla perché il contenuto della stessa* ***non*** *deve essere modificabile.)*

Eseguiamo dei controlli sui **codici di stato** restituiti dal server:

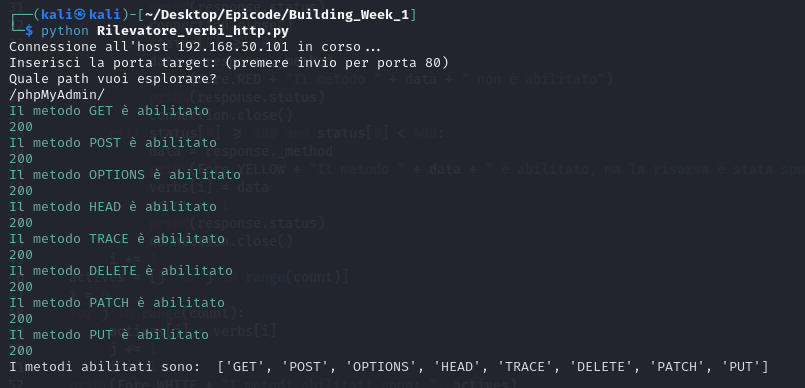
* in caso il codice sia di tipo **2xx** il metodo sarà abilitato;
* in caso il codice sia di di tipo **3xx** si verrà reindirizzati ad un nuovo URL. Si può comunque affermare che il metodo richiesto sia abilitato, anche se la risorsa non sarà raggiungibile all’indirizzo indicato poiché si verrà reindirizzati su un altro indirizzo.
* in caso il codice sia di tipo **4xx o superiore** il metodo HTTP risulterà non abilitato.

Effettuiamo il test all’interno del laboratorio virtuale, con **IP target** 192.168.50.101 e **PORTA target** 80 verificata grazie al **PORT SCANNING** effettuato in precedenza.

Inseriamo il path da seguire. **/phpMyAdmin/**

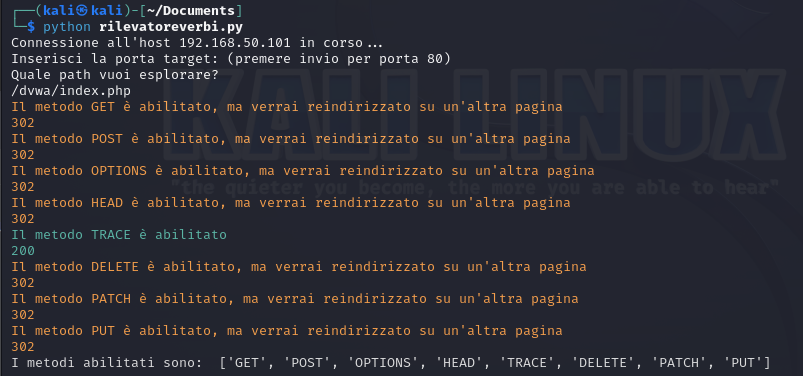
Otteniamo in risposta *lo stato dei metodi* su questo path.

*Esempio*:

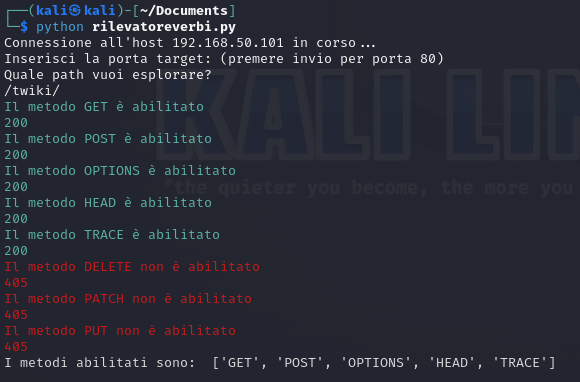


Possiamo osservare come tutti i metodi siano abilitati.

Completata l’enumerazione dei metodi HTTP sul Web Server, passiamo al controllo dei verbi HTTP abilitati sull’’**Application Server**.



Come abbiamo osservato, nonostante il *reindirizzamento* verso un’altra pagina, i verbi in giallo risultano abilitati.



Infine, utilizzando il path “/*twiki*/, osserviamo come il *verb scanner* riesca ad intercettare anche i verbi non abilitati.