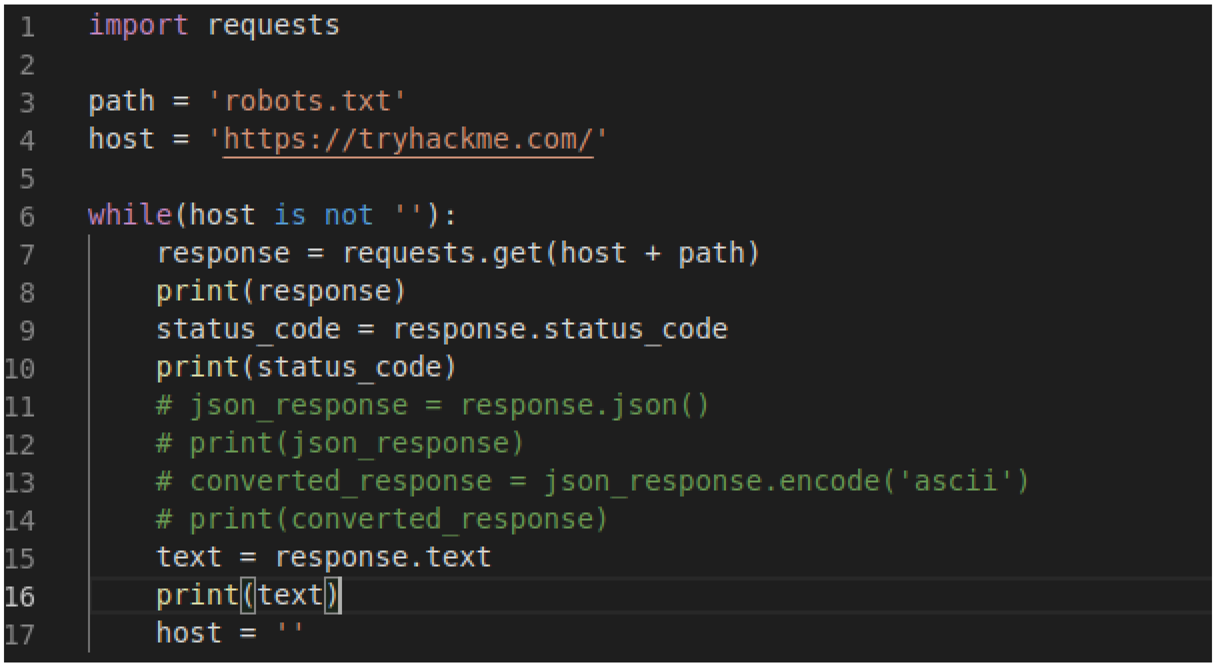
Python & Requests Library

Stuck? Join our Discord for help! https://discord.gg/wvfe3XJ

Today, we'll be looking at scripting with Python. Using scripts is an important part of practising security. The main reason to use scripts is for customisation and automation; you may want to perform actions several times that aren't supported by standard tools.

For now, let's look at using Python to make requests using a web page. For this, you'll need to download and install python3 and pip. Pip should already come installed with this version - if it isn't, you can download it from here.

Don't worry if you haven't done Python before - we'll walk through an example script and explain everything from scratch!



```python
import requests

path = 'robots.txt'
host = 'https://tryhackme.com/'

while(host is not ''):
    response = requests.get(host + path)
    print(response)
    status_code = response.status_code
    print(status_code)
    # json_response = response.json()
    # print(json_response)
    # converted_response = json_response.encode('ascii')
    # print(converted_response)
    text = response.text
    print(text)
    host = ''
```

The first few lines of Python scripts are usually import statements and they are used to import libraries. In this case, on line 1, we're importing the requests library. Software libraries/packages can be thought of as functions in a different file. We usually use libraries for a number of reasons:

- Efficiency - if someone has written code out there, then there's no point writing it from scratch(as long as it suits your purpose)
- Modularity - If we write code in terms of libraries, then we reduce the complexity of changing and maintaining it as we only have to do it in one place

In this case, we've imported the whole library. In other cases, we can import specific functions.

Line 3 and 4 are referred to as variables. Variables are just placeholders for different values. Python is what is known as a dynamically typed language, which means that the Python interpreter will try to assume what kind of data type you're trying to store in a variable. Variables are in the form of
Name = value

Here are some example declarations:

- number = 1
- decimal_number = 1.0
- string_value = 'hello'
- list = [1,2,3]
- dictionary_values = {"value_one" : 1, "value_two": 2}

Here note that a string value has to be enclosed in single quotes('') or double quotes(""). The list variable is used to store multiple values or variables. The dictionary_values variable is known as a dictionary that stores data in a key:value format where the keys are unique values. To access elements of a list use the syntax:

- List_name[position]. E.g. list[0] will access the value 1. Lists in Python(and in other programming languages) start at the position 0 instead of 1.
- Dictionary_variable[key_name] e.g. dictionary_values['value_one'] will access the value 1

On line 6, we have a while loop. As stated in the article earlier, we may want to repeat actions multiple times based on a condition and loops let us do this. The format of a python while loop is

```
while(condition is met):
   Execute_code_here
```

The condition can be various statements including:

- Arithmetic conditions(variable = value)
- String comparisons(variable is 'value')
- Chaining conditions using the and, or and not key words(if variable = value and variable is 'value')

The condition in this while loop is checking if the host variable is not an empty string(i.e. It's checking if the host variable actually contains something). On line 7 you can see we're actually using the library we imported. Since we used one import statement and didn't import functions, the syntax of this line is:

Library_used.function_from_library
Requests.get

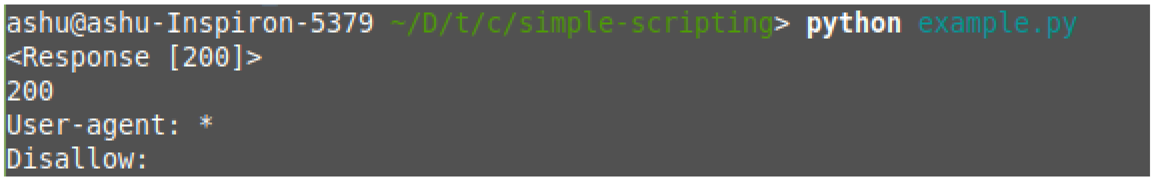
A function usually takes parameters or arguments. It needs parameters or arguments when it needs to use external values. This is why functions are so useful. You can have the same block of code and interchange different values. Here you can see the parameter is host + path. This syntax is used to concatenate two strings so host + path will actually be interpreted by python as https://tryhackme.com/ + robots.txt = https://tryhackme.com/robots.txt

Line 7 is essentially taking a URL, making a GET request to the page specified and storing the response in a variable. The requests library supports other actions and you can check it out here. Once we get the response, we can do a lot of different things with it. Line 9 accesses the status code - this is important to check if a resource exists. In most cases, we'll expect to get 200 response. Line 11-14 are commented out using the # character. This means that these lines will not be executed.

Line 11 shows that we can actually access the data in JSON format which is the same as that of a python dictionary(mentioned above). The JSON format is is commonly used to send and retrieve data. Line 13 is needed because the request library converts the data sent inside the JSON object to a different type of encoding(in this case unicode encoding). To make the data easier to interpret and read, this is converted to human readable ASCII. You can see from above that the print() syntax is used to print out the data specified.

Using the requests library, we can just access the raw text using the .text function. Finally, we set the host variable to ''(an empty string). We do this because we don't want the loop to run infinitely; leaving the host value the same would mean that the condition evaluated by the loop stays the same and the code in the loop is continually executed.

We would execute a python script using the command
python script_name.py


```
ashu@ashu-Inspiron-5379 ~/D/t/c/simple-scripting> python example.py
<Response [200]>
200
User-agent: *
Disallow:
```

When we run the script above, we see that it prints the response variable(the first line in the image above). It then prints the status code(the second line). It then prints the text received from the response(3rd and 4th line).

How would you actually use something like this in the real world:

- If you're trying to write a script to test login functionality of a web application where the login is multi step and uses data from different pages:
    - E.g. first page requires a username and password and second page requires data returned from this response
- Crawling web pages to build a site map
    - Send a request to a page, get other links and do this for every link on the domain