Task 3

Download Source Code Here

3.1 Classes and Objects in Py

01 - OOP in Python.py > ...

```python
# @author : HUSSAIN ASHIQ
# @Description: This code shows some comcepts of OOP in python


# class 1 : Student Class
class Student:
    name = "" #class attribute
    college = "MCS" #default value
    semester = ""
    def __init__(self, name, semester):
        self.name = name
        self.semester = semester
    def introduce(self):
        print("My name is {} and I a stduent of {}, \
         studying {} semester.".format(self.name, self.college, self.semester))

student1 = Student("HUSSAIN ASHIQ", "5TH")
student1.introduce()

## ============================================================================
# class 2: Software Engineering Student inherting from Student class
# demontrating inhertance : SE student is a Student

class SE_Student(Student):
    department = "CSE"
    def __init__(self, name, semester):
        super().__init__(name, semester)
    def whoami(self):
        print(f"I am {self.name} and I am an CSE student.")

se_std = SE_Student("KHATTAK", "5TH")
se_std.introduce()
se_std.whoami()
## ============================================================================
# class 3: Electrical Engineering Student inherting from Student class
# demontrating inhertance : SE student is a Student
class EE_Student(Student):
    department = "EE"
    def __init__(self, name, semester):
        super().__init__(name, semester)
    def whoami(self):
        print(f"I am {self.name} and I am an EE student.")

```

```python
ee_std = EE_Student("HAMZA", "5TH")
ee_std.introduce()
ee_std.whoami()
## ================================================================================
# class 4 : Computer class
# demo of Encapsuation
class Computer:

    def __init__(self):
        self.__maxprice = 900

    def sell(self):
        print("Selling Price: {}".format(self.__maxprice))

    def setMaxPrice(self, price):
        self.__maxprice = price

c = Computer()
c.sell()
# change the price
c.__maxprice = 1000 #no effect since the __maxprice is a private attribute.
c.sell()
# using setter function
c.setMaxPrice(1000)
c.sell()

## ================================================================================
# Class 5 - Polymorphism
# Polymorphism is an ability (in OOP) to use a common interface for multiple forms (data types).
# Suppose, we need to color a shape, there are multiple shape options (rectangle, square, circle).
# However we could use the same method to color any shape. This concept is called Polymorphism.

class Computer:

    def __init__(self):
        self.__maxprice = 900

    def sell(self):
        print("Selling Price: {}".format(self.__maxprice))

    def setMaxPrice(self, price):
        self.__maxprice = price

c = Computer()
c.sell()

# change the price
c.__maxprice = 1000
c.sell()

# using setter function
c.setMaxPrice(1000)
c.sell()

## ================================================================================
```

```
PS C:\Users\light-bringer\Desktop\NCSAEL Internship\TASKS_pYTHON\Task3> python -u "c:\Users\light-bring
er\Desktop\NCSAEL Internship\TASKS_pYTHON\Task3\01 - OOP in Python.py"
My name is HUSSAIN ASHIQ and I a stduent of MCS, studying 5TH semester.
My name is KHATTAK and I a stduent of MCS, studying 5TH semester.
I am KHATTAK and I am an CSE student.
My name is HAMZA and I a stduent of MCS, studying 5TH semester.
I am HAMZA and I am an EE student.
Selling Price: 900
Selling Price: 900
Selling Price: 1000
Selling Price: 900
Selling Price: 900
Selling Price: 1000
PS C:\Users\light-bringer\Desktop\NCSAEL Internship\TASKS_pYTHON\Task3>
```

3.2 Typecasting in Python

There may be times when you want to specify a type on to a variable. This can be done with casting. Python is an object-orientated language, and as such it uses classes to define data types, including its primitive types.
Casting in python is therefore done using constructor functions:

- int() - constructs an integer number from an integer literal, a float literal (by removing all decimals), or a string literal (providing the string represents a whole number)
- float() - constructs a float number from an integer literal, a float literal or a string literal (providing the string represents a float or an integer)
- str() - constructs a string from a wide variety of data types, including strings, integer literals and float literals

```python
#==========================================
# implicit conversion
# ==========================================
num_int = 123
num_flo = 1.23
num_new = num_int + num_flo
print("datatype of num_int:",type(num_int))
print("datatype of num_flo:",type(num_flo))
print("Value of num_new:",num_new)
print("datatype of num_new:",type(num_new))
#------------------------------------------
```

Output

```
PS C:\Users\light-bringer\Desktop\NCSAEL Internship\TASKS_pYTHON\Task3> python
esktop\NCSAEL Internship\TASKS_pYTHON\Task3\02 - Typecasting.py"
datatype of num_int: <class 'int'>
datatype of num_flo: <class 'float'>
Value of num_new: 124.23
datatype of num_new: <class 'float'>
PS C:\Users\light-bringer\Desktop\NCSAEL Internship\TASKS_pYTHON\Task3>
```

```python
#============================================
# explicit conversion
#============================================
num_int = 123
num_str = "456"
print("Data type of num_int:",type(num_int))
print("Data type of num_str before Type Casting:",type(num_str))
num_str = int(num_str)
print("Data type of num_str after Type Casting:",type(num_str))
num_sum = num_int + num_str
print("Sum of num_int and num_str:",num_sum)
print("Data type of the sum:",type(num_sum))
```

Output

```
PS C:\Users\light-bringer\Desktop\NCSAEL Internship\TASKS_pYTHON\Task3> python
\Desktop\NCSAEL Internship\TASKS_pYTHON\Task3\tempCodeRunnerFile.py"
Data type of num_int: <class 'int'>
Data type of num_str before Type Casting: <class 'str'>
Data type of num_str after Type Casting: <class 'int'>
Sum of num_int and num_str: 579
Data type of the sum: <class 'int'>
PS C:\Users\light-bringer\Desktop\NCSAEL Internship\TASKS_pYTHON\Task3>
```

3.3 Installing Django

Installation instructions are slightly different depending on whether you're installing a distribution-specific package, downloading the latest official release, or fetching the latest development version.

Installing an official release with `pip`

This is the recommended way to install Django.

1. Install pip. The easiest is to use the standalone pip installer. If your distribution already has **`pip`** installed, you might need to update it if it's outdated. If it's outdated, you'll know because installation won't work.

2. Enter the command in terminal: **pip install django**

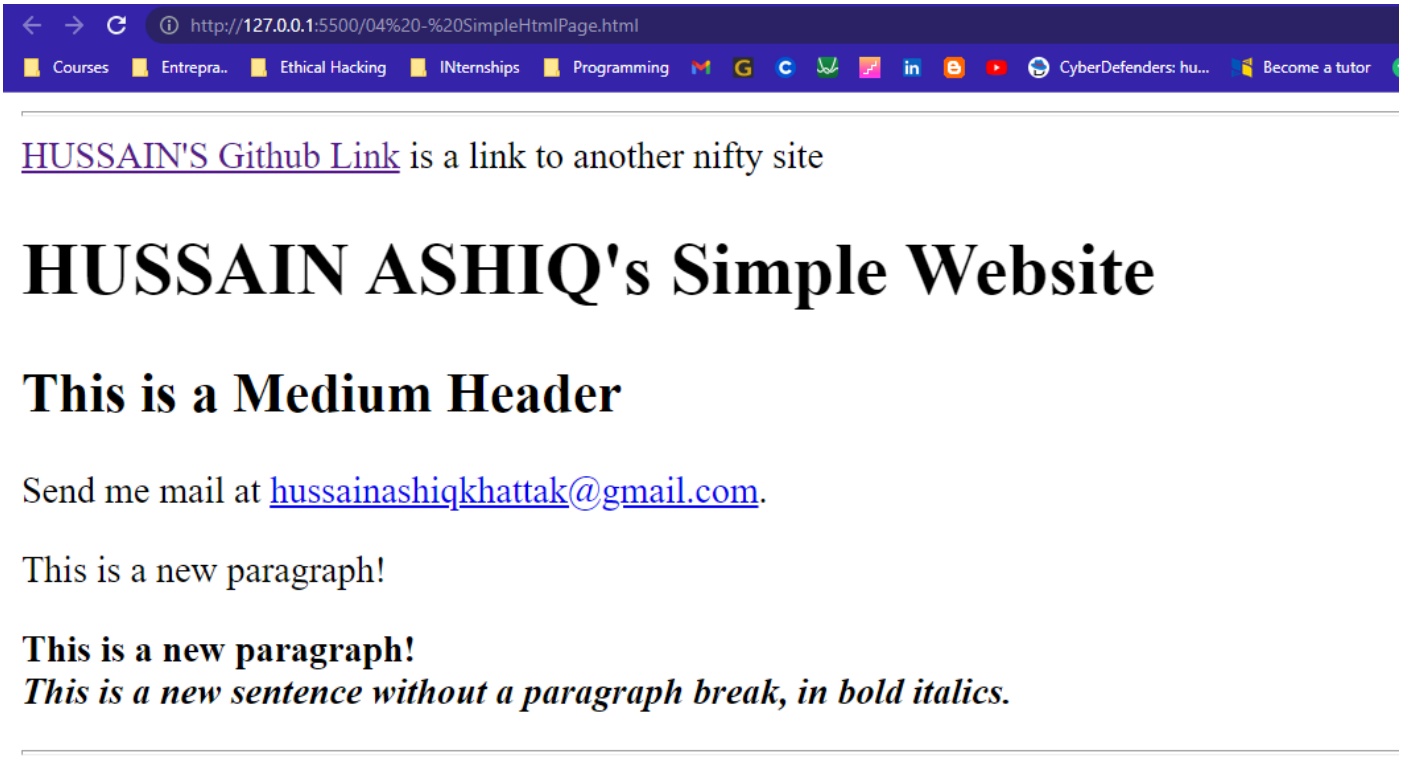
```
PS C:\Users\light-bringer\Desktop\NCSAEL Internship\TASKS_pYTHON\Task3> pip install django
```

3.4 Simple HTML Page

04 - SimpleHtmlPage.html

```
<html>
  <head>
    <title>Simple Page</title>
  </head>
  <body bgcolor="FFFFFF">
    <hr />
    <a href="https://github.com/hussainashiqktk">HUSSAIN'S Github Link</a> is a link to another nifty site
    <h1>HUSSAIN ASHIQ's Simple Website</h1>
    <h2>This is a Medium Header</h2>

    Send me mail at
    <a href="mailto:hussainashiqkhattak@gmail.com">
      hussainashiqkhattak@gmail.com</a
    >.

    <p>This is a new paragraph!</p>
    <p>
      <b>This is a new paragraph!</b>
      <br />
      <b
        ><i
          >This is a new sentence without a paragraph break, in bold italics.</i
        ></b
      >
    </p>
    <hr />
  </body>
</html>
```

When viewed in browser the page looks the following:



3.5 Python's Requests library

Requests is a simple, yet elegant, HTTP library.

Requests allows you to send HTTP/1.1 requests extremely easily. There's no need to manually add query strings to your URLs, or to form-encode your PUT & POST data — but nowadays, just use the json method!
Requests is one of the most downloaded Python packages today, pulling in around 30M downloads / week — according to GitHub, Requests is currently depended upon by 1,000,000+ repositories. You may certainly put your trust in this code.

Requests is a HTTP library for the Python programming language. The goal of the project is to make HTTP requests simpler and more human-friendly. The current version is 2.28.0. Requests is released under the Apache License 2.0. Requests is one of the most popular Python libraries that is not included with Python.

3.6 Getting google search with Python's Requests

05 - requestingGoogle.py > ...

```python
import requests
url = "https://www.google.com/search?q=cybersecurity"
response = requests.get(url)
file = open("06_response.html", "wb") #for viewing it as HTML in browser
file = open("06_response.text", "wb") #saving response in a text file
file.write(response.content)
print(response.content)
file.close()

```

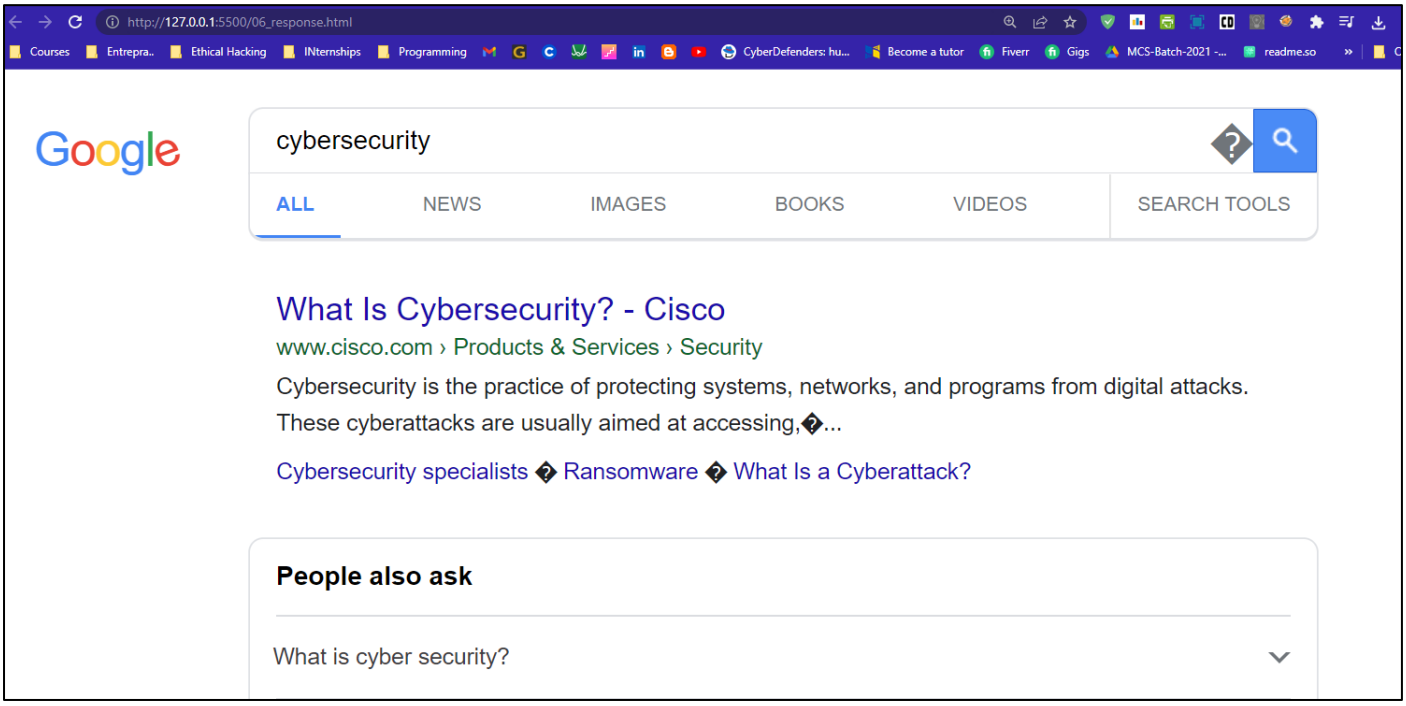
The output is saved in a text file named 06_response.txt and also in an html file named 06_response.html for viewing it in a browser.

05 - requestingGoogle.py | 06_response.txt

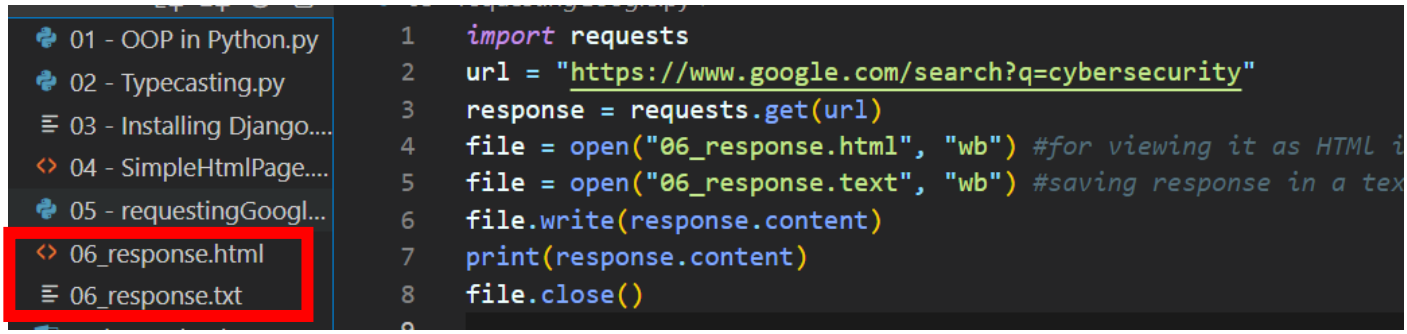
06_response.txt

```
<!doctype html><html lang="en-PK"><head><meta charset="UTF-8"><meta content="/images/branding/googleg/1x/
googleg_standard_color_128dp.png" itemprop="image"><title>cybersecurity - Google Search</title><script
nonce="XVGPOUsEdqrCidrln2JTqg">(function(){
document.documentElement.addEventListener("submit",function(b){var a;if(a=b.target){var c=a.getAttribute("data-submitfalse");
a="1"===c||"q"===c&&!a.elements.q.value?!0:!1}else a=!1;a&&(b.preventDefault(),b.stopPropagation())},!0);document.
documentElement.addEventListener("click",function(b){var a;a:{for(a=b.target;a&&a!==document.documentElement;a=a.parentElement)
if("A"===a.tagName){a="1"===a.getAttribute("data-nohref");break a}a=!1}a&&b.preventDefault()},!0);}).call(this);(function(){
var a=window.performance;window.start=Date.now();a:{var b=window;if(a){var c=a.timing;if(c){var d=c.navigationStart,f=c.
responseStart;if(f>d&&f<=window.start){window.start=f;b.wsrt=f-d;break a}}a.now&&(b.wsrt=Math.floor(a.now()))}window.
google=window.google||{};var h=function(g){g&&g.target.setAttribute("data-iml",Date.now())};document.documentElement.
addEventListener("load",h,!0);google.rglh=function(){document.documentElement.removeEventListener("load",h,!0)};}).call(this);
(function(){window._noJsad=1;})();(function(){window._skwEvts=[];})();(function(){window.google.erd={jsr:1,bv:1632,de:true};})
();(function(){var sdo=false;var mei=10;
var h=this||self;var k,l=null!=(k=h.mei)?k:1,n,p=null!=(n=h.sdo)?n:!0,q=0,r,t=google.erd,v=t.jsr;google.ml=function(a,b,d,m,e)
{e=void 0===e?2:e;b&&(r=a&&a.message);if(google.dl)return google.dl(a,e,d),null;if(0>v){window.console&&console.error(a,d);if
```

The html file when viewed in a browser looks as the following:



These two files are created by the above code in the current working directory:

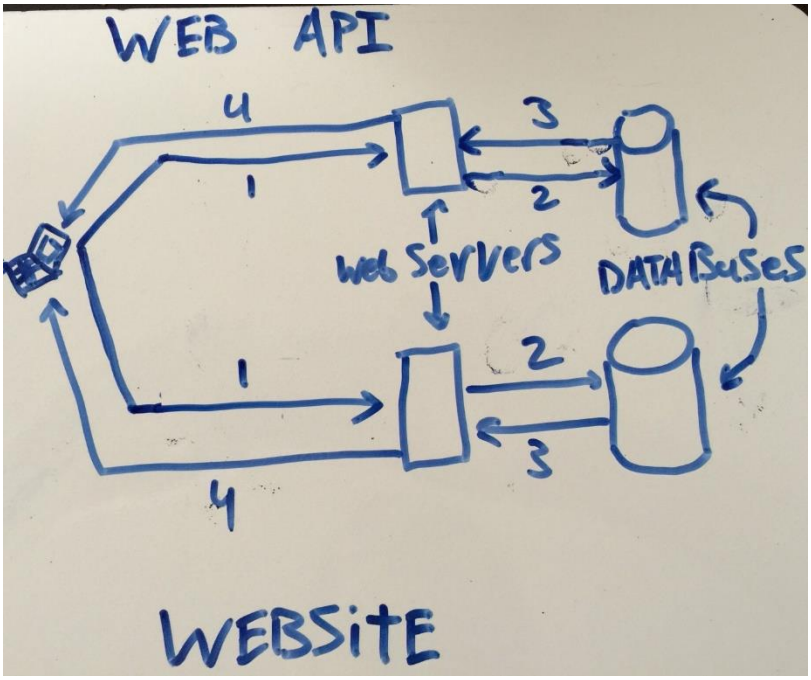


Task 4

4.1 APIs and their uses

“In computer programming, an application programming interface (API) is a set of subroutine definitions, protocols, and tools for building application software. In general terms, it is a set of clearly defined methods of communication between various software components”.

In general, APIs define the rules that programmers must follow to interact with a programming language, a software library, or any other software tool. Lately though, the term API is most often used to describe a particular kind of web interface. These Web APIs are a set of rules for interacting with a webserver (such as a Salesforce server), with the most common use case being data retrieval. API’s provide mechanisms for CRM customers to access and manipulate data stored by the API provider (Salesforce in this example). The user makes a “request” to a Salesforce webserver, that webserver accesses a Salesforce database (with the customers data) and returns it to the requester in a “response”.



Steps 1–3 are the same for both kinds of requests: 1. You send an HTTP request to a webserver. 2. That server queries its internal database. 3. The database gives the server the requested data. 4. The data is returned to you in an HTTP response as HTML/CSS/JS to display (website) or as JSON/XML (web API).

4.2 My Simple API

This is the code for the API I have made. This API has two endpoints. One is Homepage and the other is /user/.

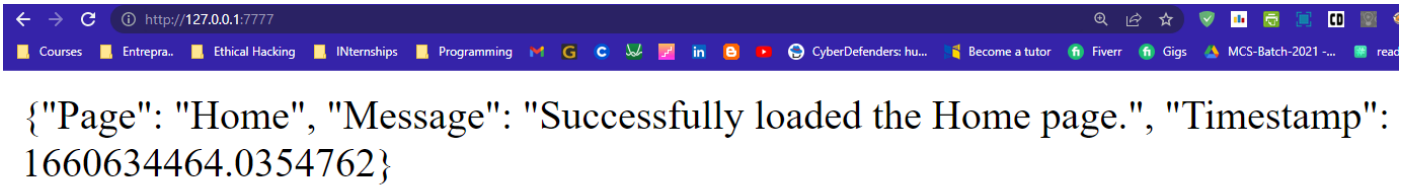
02_SimpleAPI.py > ...

```python

from flask import *
import json, time

app = Flask(__name__)

@app.route("/", methods=['GET'])
def home_page():
    data_set = {"Page":"Home",
    "Message":"Successfully loaded the Home page.",
    "Timestamp":time.time()
    }
    json_dump = json.dumps(data_set)
    return json_dump


@app.route("/user/", methods=['GET'])
def request_page():
    user_query = str(request.args.get('user')) #  /user/?user=jkdfksd
    data_set = {"Page":"Request",
    "Message":f"Successfully got the request for {user_query}.",
    "Timestamp":time.time()
    }
    json_dump = json.dumps(data_set)
    return json_dump

if __name__ == "__main__":
    app.run(port=7777)
```

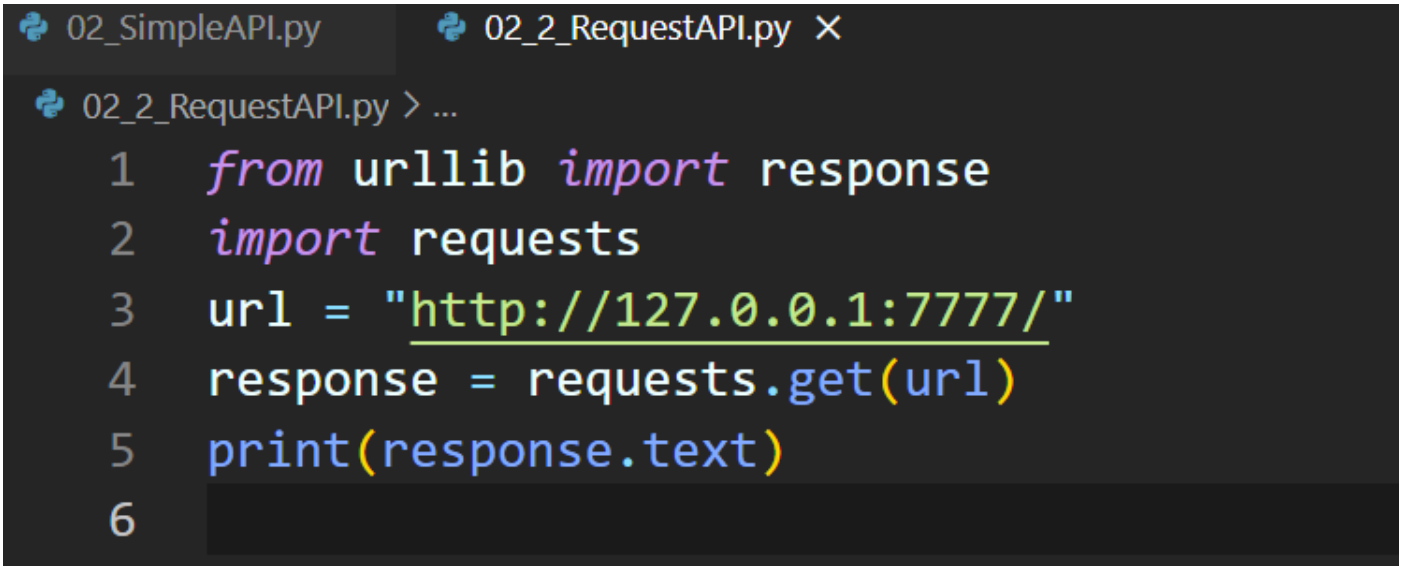
When this file is run, we get the following output saving that the app is running on localhost with the specified port i.e. 7777.

```
Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\light-bringer\Desktop\NCSAEL Internship\TASKS_pYTHON\Task4> python -u "c:\Users\light-bringer\Desktop\NCSAEL Internship\TASKS_pYTHON\Task4\02_SimpleAPI.py"
 * Serving Flask app "02_SimpleAPI" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:7777/ (Press CTRL+C to quit)
```

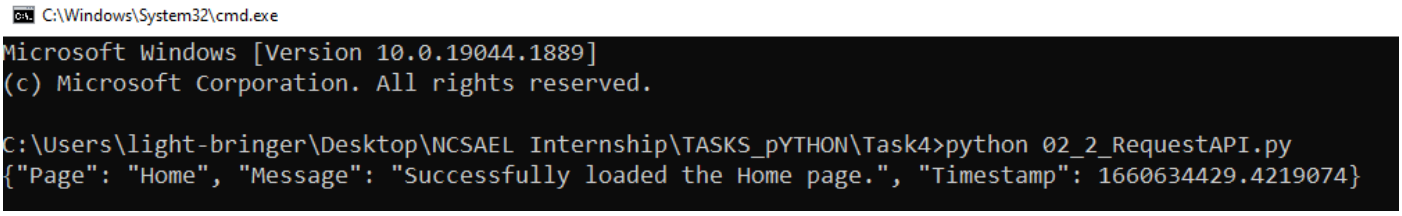
When we visit the link in a browser we see the following:



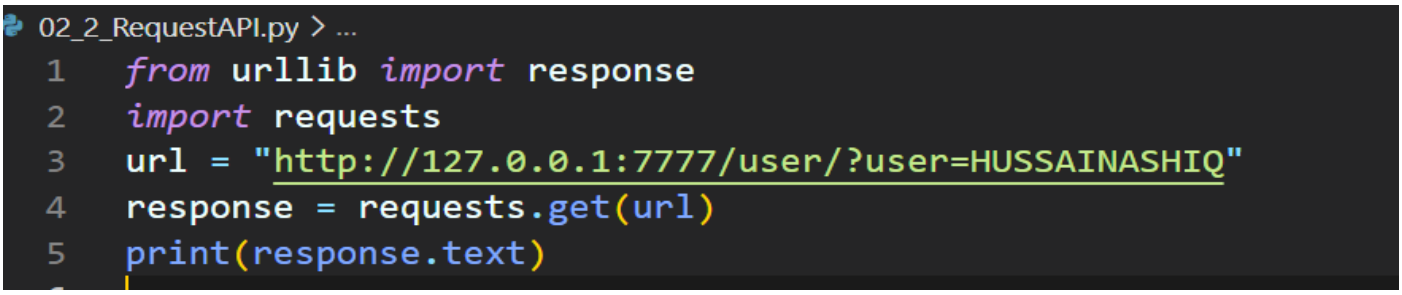
Now let's fetch the result using another python program. The following code fetches the result from the API which is already running on the localhost.

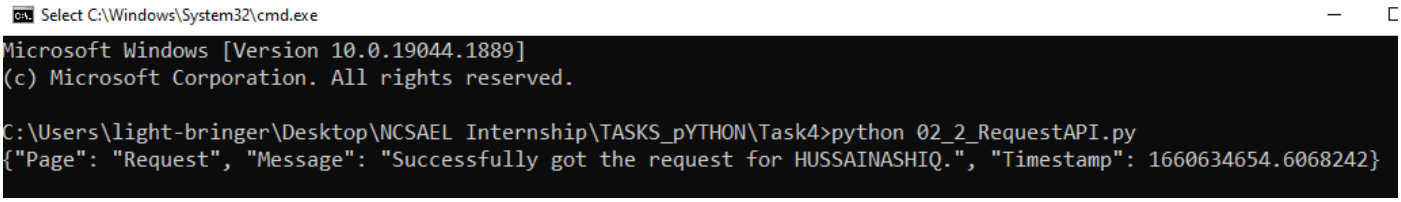


When we run this file, the result is printed on the terminal:



The following modified code fetched result from the second end point of the API.





4.3 Consuming other public APIs

02_03_ConsumingFreeAPIForTesting.py > ...

```
1
2  import requests
3  response = requests.get("https://api.thedogapi.com")
4  print(response.text)
```

Output

```
PS C:\Users\light-bringer\Desktop\NCSAEL Internship\TASKS_pYTHON\Task4> python -u "c:\Us
ers\light-bringer\Desktop\NCSAEL Internship\TASKS_pYTHON\Task4\02_03_ConsumingFreeAPIFor
Testing.py"
{"message":"The Dog API","version":"1.1.2"}
PS C:\Users\light-bringer\Desktop\NCSAEL Internship\TASKS_pYTHON\Task4>
```

Fetching result from another endpoint of thedogapi.

02_03_ConsumingFreeAPIForTesting.py > ...

```
1
2  import requests
3  response = requests.get("https://api.thedogapi.com/v1/breeds")
4  print(response.text)
```

```
PS C:\Users\light-bringer\Desktop\NCSAEL Internship\TASKS_pYTHON\Task4> python -u "c:\Users\light-bringer\Desktop\NCSAEL Internsh
ip\TASKS_pYTHON\Task4\02_03_ConsumingFreeAPIForTesting.py"
[{"weight":{"imperial":"6 - 13","metric":"3 - 6"},"height":{"imperial":"9 - 11.5","metric":"23 - 29"},"id":1,"name":"Affenpinsche
r","bred_for":"Small rodent hunting, lapdog","breed_group":"Toy","life_span":"10 - 12 years","temperament":"Stubborn, Curious, Pl
ayful, Adventurous, Active, Fun-loving","origin":"Germany, France","reference_image_id":"BJa4kxc4X","image":{"id":"BJa4kxc4X","wi
dth":1600,"height":1199,"url":"https://cdn2.thedogapi.com/images/BJa4kxc4X.jpg"}},{"weight":{"imperial":"50 - 60","metric":"23 -
27"},"height":{"imperial":"25 - 27","metric":"64 - 69"},"id":2,"name":"Afghan Hound","country_code":"AG","bred_for":"Coursing and
 hunting","breed_group":"Hound","life_span":"10 - 13 years","temperament":"Aloof, Clownish, Dignified, Independent, Happy","origi
n":"Afghanistan, Iran, Pakistan","reference_image_id":"hMyT4CDXR","image":{"id":"hMyT4CDXR","width":606,"height":380,"url":"https
://cdn2.thedogapi.com/images/hMyT4CDXR.jpg"}},{"weight":{"imperial":"44 - 66","metric":"20 - 30"},"height":{"imperial":"30","metr
ic":"76"},"id":3,"name":"African Hunting Dog","bred_for":"A wild pack animal","life_span":"11 years","temperament":"Wild, Hardwor
king, Dutiful","origin":"","reference_image_id":"rkiByec47","image":{"id":"rkiByec47","width":500,"height":335,"url":"https://cdn
2.thedogapi.com/images/rkiByec47.jpg"}},{"weight":{"imperial":"40 - 65","metric":"18 - 29"},"height":{"imperial":"21 - 23","metri
c":"53 - 58"},"id":4,"name":"Airedale Terrier","bred_for":"Badger, otter hunting","breed_group":"Terrier","life_span":"10 - 13 ye
ars","temperament":"Outgoing, Friendly, Alert, Confident, Intelligent, Courageous","origin":"United Kingdom, England","reference_
image_id":"1-7cgoZSh","image":{"id":"1-7cgoZSh","width":645,"height":430,"url":"https://cdn2.thedogapi.com/images/1-7cgoZSh.jpg"}
```

4.4 Getting result from VirusTotal using Python

Code that fetches the result from virus total for a sample malicious hash and stores the result in a text and a json file. Note that I have to signup up on VirusTotal.com to get the API key.

03_VirusTotal.py > ...

```python

#I signed up on VT and got my API key
from urllib import request

import requests

api_key = "d42b6925fe69ea9a46dfa8f46329a6b32a70c8cd1a2f5799bf960c59a36b54ab"
malacious_hash = "c0202cf6aeab8437c638533d14563d35"
url = "https://www.virustotal.com/api/v3/files/"+malacious_hash

headers = {
    "Accept": "application/json",
    "x-apikey": api_key
}

response = requests.get(url, headers=headers)
file = open("04_responseFromVirusTotal.json", "w") #saving in JSON file for syntax highlighting
file = open("04_responseFromVirusTotal.txt", "w") #saving in text file
file.write(response.text)
file.close()
print(response.text)

```

Following is a snip of the full output. The text file is shown in following pages.

```
PS C:\Users\light-bringer\Desktop\NCSAEL Internship\TASKS_pYTHON\Task4> python -u "c:\Users\light-bringer\Desktop\NCSAEL Internship\TASKS
_pYTHON\Task4\03_VirusTotal.py"
{
    "data": {
        "attributes": {
            "type_description": "Win32 EXE",
            "tlsh": "T1CF048D4772A532F8F173CA3585528452F7B6BC7507609B6F03A4827A1F176929F3AF20",
            "vhash": "015076655d155515555088z54hz3lz",
            "trid": [
                {
                    "file_type": "Win64 Executable (generic)",
                    "probability": 48.7
                },
                {
                    "file_type": "Win16 NE executable (generic)",
                    "probability": 23.3
                },
                {
                    "file_type": "OS/2 Executable (generic)",
                    "probability": 9.3
                },
                {
                    "file_type": "Generic Win/DOS Executable",
                    "probability": 9.2
                },
                {
                    "file_type": "DOS Executable Generic",
```

Snip from the text file:

04_responseFromVirusTotal.txt


```
{
    "data": {
        "attributes": {
            "type_description": "Win32 EXE",
            "tlsh": "T1CF048D4772A532F8F173CA3585528452F7B6BC7507609B6F03A4827A1F176929F3AF20",
            "vhash": "015076655d155515555088z54hz3lz",
            "trid": [
                {
                    "file_type": "Win64 Executable (generic)",
                    "probability": 48.7
                },
                {
                    "file_type": "Win16 NE executable (generic)",
                    "probability": 23.3
                },
                {
                    "file_type": "OS/2 Executable (generic)",
                    "probability": 9.3
                },
                {
                    "file_type": "Generic Win/DOS Executable",
                    "probability": 9.2
                },
                {
                    "file_type": "DOS Executable Generic",
                    "probability": 9.2
                }
            ],
            "crowdsourced_yara_results": [
                {
                    "description": "Detects strings known from Ryuk Ransomware",
                    "source": "https://github.com/Neo23x0/signature-base",
                    "author": "Florian Roth",
                    "ruleset_name": "crime_ryuk_ransomware",
                    "rule_name": "MAL_Ryuk_Ransomware",
                    "ruleset_id": "000a47a18d"
                },
                {
                    "description": "detects command variations typically used by ransomware",
                    "source": "https://github.com/ditekshen/detection",
```