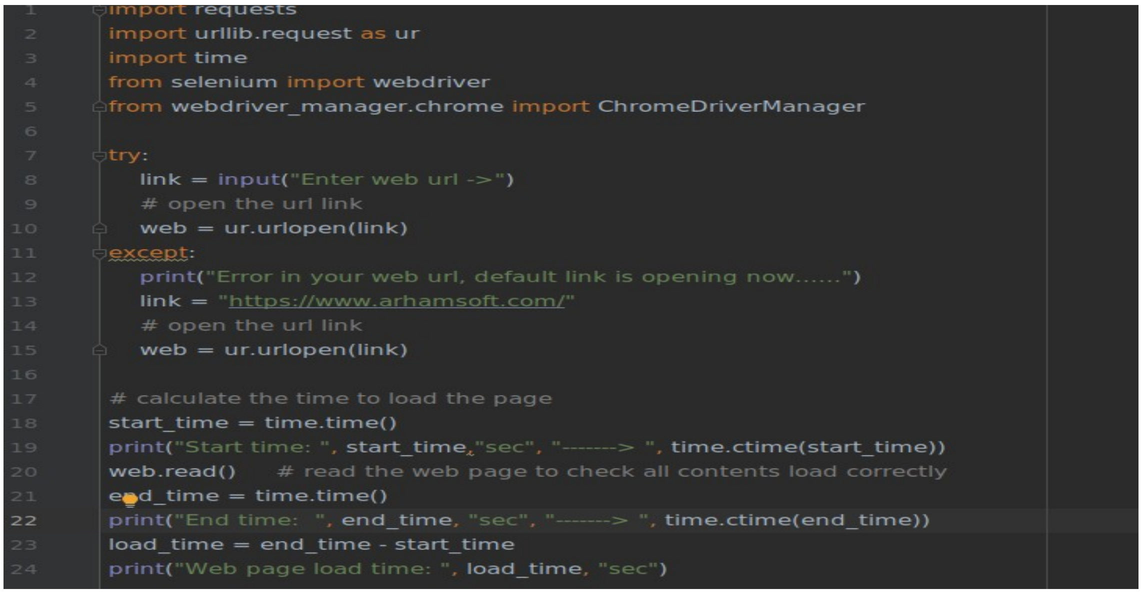
Python Assignment 01 – 28 Oct 2021

Task 01: Measure time to load page

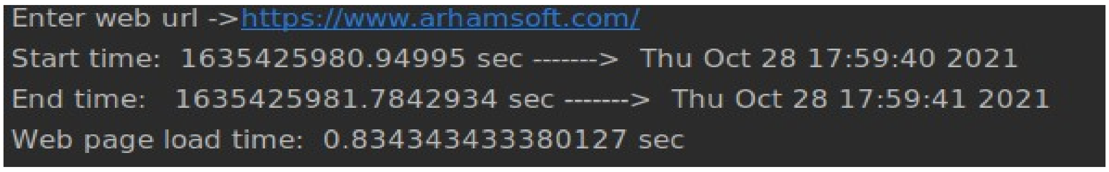
Method 01:

- **urllib:** urllib package is the URL handling module for python. It is used to fetch URLs. It uses the urlopen function and is able to fetch URLs using a variety of different protocols.
- **import urllib.request:** The urllib.request module defines functions and classes which help in opening URLs (mostly HTTP) in a complex world — basic and digest authentication, redirections, cookies and more.
- **urlib.request.urlopen(url):** Open the URL url, which can be either a string or a Request object.
- **time.time():** The time() function returns the number of seconds passed since epoch. For Unix system, January 1, 1970, 00:00:00 at UTC is epoch (the point where time begins).
- **time.ctime():** Convert a time in seconds since the Epoch to a string in local time. This is equivalent to asctime(localtime(seconds)). When the time tuple is not present, current time as returned by localtime() is used.
- **Input:**



```python
import requests
import urllib.request as ur
import time
from selenium import webdriver
from webdriver_manager.chrome import ChromeDriverManager

try:
    link = input("Enter web url ->")
    # open the url link
    web = ur.urlopen(link)
except:
    print("Error in your web url, default link is opening now......")
    link = "https://www.arhamsoft.com/"
    # open the url link
    web = ur.urlopen(link)

# calculate the time to load the page
start_time = time.time()
print("Start time: ", start_time,"sec", "-------> ", time.ctime(start_time))
web.read()     # read the web page to check all contents load correctly
end_time = time.time()
print("End time:  ", end_time, "sec", "-------> ", time.ctime(end_time))
load_time = end_time - start_time
print("Web page load time: ", load_time, "sec")
```

- **Expected output:**

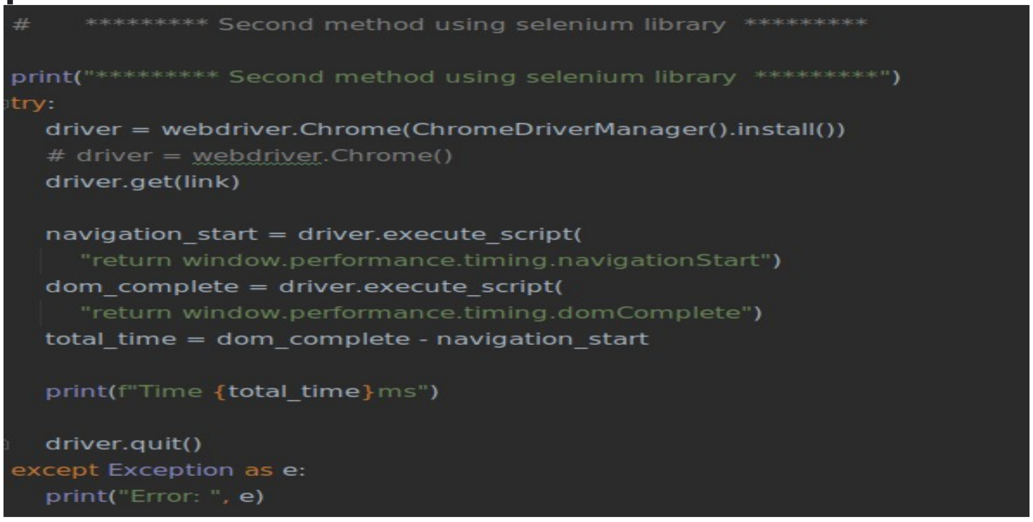


```
Enter web url ->https://www.arhamsoft.com/
Start time:  1635425980.94995 sec ------->  Thu Oct 28 17:59:40 2021
End time:   1635425981.7842934 sec ------->  Thu Oct 28 17:59:41 2021
Web page load time:  0.834343433380127 sec
```

Method 02:

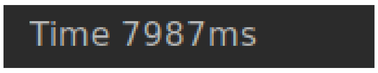
- **install selenium library:** Run this command on terminal “pip install selenium”. Selenium is a powerful tool for controlling web browsers through programs and performing browser automation. It is functional for all browsers, works on all major OS and its scripts are written in various languages i.e Python, Java, C#, etc, we will be working with Python.
- **From selenium import webdriver:** Selenium WebDriver is a web framework that permits you to execute cross-browser tests. This tool is used for automating web-based application testing to verify that it performs expectedly.
- **from webdriver_manager.chrome import ChromeDriverManager:** It is used to install the webdriver at runtime.

- **driver.get(link):** It get url link and open the url on chrome window.
- **driver.execute_script(“”):** Use the browser Navigation Timing API to get some numbers.
- **Input:**



```
#    ********* Second method using selenium library  *********

print("********* Second method using selenium library  *********")
try:
    driver = webdriver.Chrome(ChromeDriverManager().install())
    # driver = webdriver.Chrome()
    driver.get(link)

    navigation_start = driver.execute_script(
        "return window.performance.timing.navigationStart")
    dom_complete = driver.execute_script(
        "return window.performance.timing.domComplete")
    total_time = dom_complete - navigation_start

    print(f"Time {total_time}ms")

    driver.quit()
except Exception as e:
    print("Error: ", e)
```

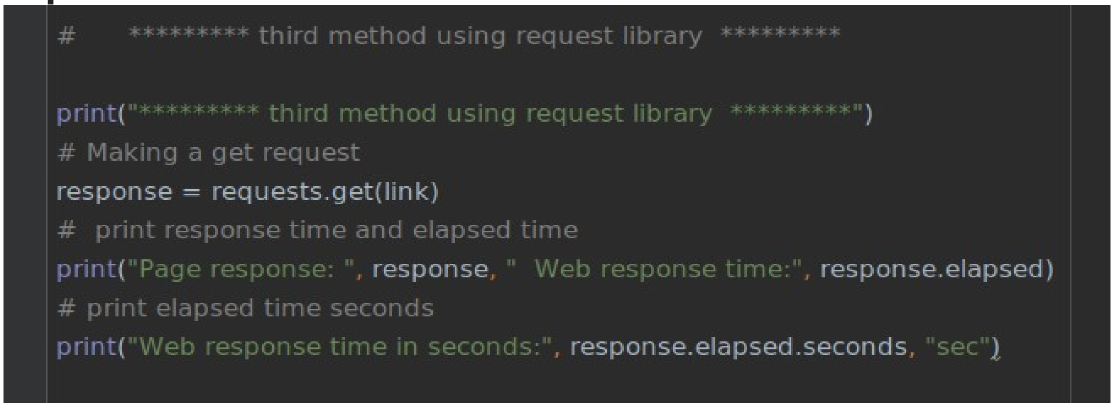
- **Expected output:**



```
Time 7987ms
```

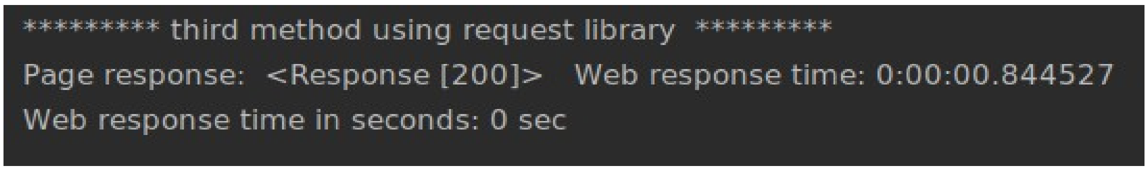
<u>Method 03:</u>

- **import requests:** requests will allow you to send HTTP/1.1 requests using Python. With it, you can add content like headers, form data, multipart files, and parameters via simple Python libraries. It also allows you to access the response data of Python in the same way.
- **requests.get(url):** It get the url link and load the data of url web response.
- **response.elapsed:** response.elapsed returns a timedelta object with the time elapsed from sending the request to the arrival of the response. It is often used to stop the connection after a certain point of time is “elapsed”.
- **Input:**



```
#    ********* third method using request library  *********

print("********* third method using request library  *********")
# Making a get request
response = requests.get(link)
#  print response time and elapsed time
print("Page response: ", response, "  Web response time:", response.elapsed)
# print elapsed time seconds
print("Web response time in seconds:", response.elapsed.seconds, "sec")
```

- **Expected output:**

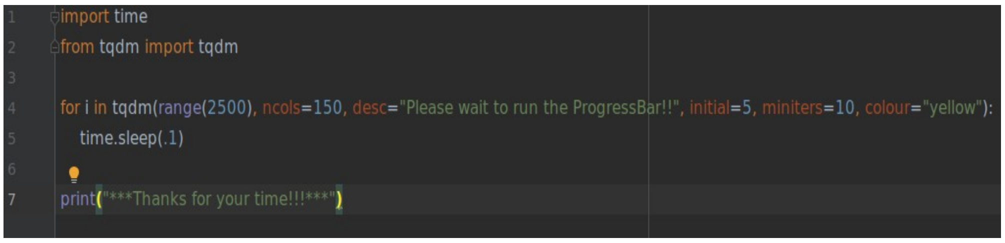


```
********* third method using request library  *********
Page response:  <Response [200]>   Web response time: 0:00:00.844527
Web response time in seconds: 0 sec
```

Python Assignment 01 – 28 Oct 2021

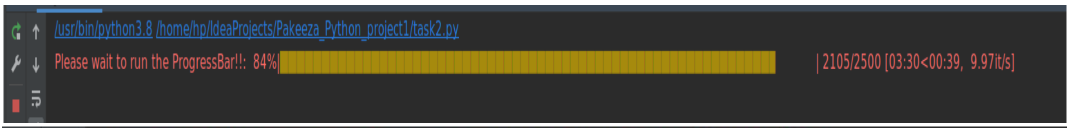
Task 02: Make progress bar using 'tqdm'

- **install tqdm library:** Run this command on terminal "pip install tqdm". "tqdm" is a Python library that allows you to output a smart progress bar by wrapping around any iterable. A tqdm progress bar not only shows you how much time has elapsed, but also shows the estimated time remaining for the iterable.
- **import time:** This module provides various time-related functions. For related functionality, see also the sleep, datetime and calendar modules.
- **tqdm syntax:** tqdm(iterable)
- **Parameters in tqdm:**
  - **Iterable:** It can be a range, a list whose progress we have to check.
  - **ncols:** (int, optional)
    The width of the entire output message. If specified, dynamically resizes the progressbar to stay within this bound. If unspecified, attempts to use environment width. The fallback is a meter width of 10 and no limit for the counter and statistics. If 0, will not print any meter (only stats).
  - **desc:** (str, optional)
    Prefix or text message for the progressbar.
  - **initial:** (int or float, optional)
    The initial counter value. Useful when restarting a progress bar [default: 0]. If using float, consider specifying {n:.3f} or similar in bar_format, or specifying unit_scale.
  - **miniters:** (int or float, optional)
    Minimum progress display update interval, in iterations. If 0 and dynamic_miniters, will automatically adjust to equal min interval. If > 0, will skip display of specified number of iterations.
  - **colour:** (str, optional)
    Bar colour (e.g: 'yellow', 'ffff00').
- **Example:**
  - **Input:**

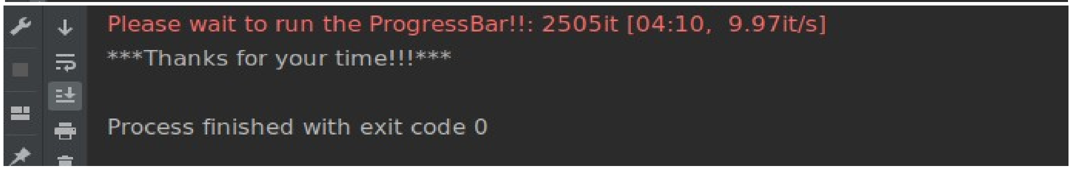


```python
import time
from tqdm import tqdm

for i in tqdm(range(2500), ncols=150, desc="Please wait to run the ProgressBar!!", initial=5, miniters=10, colour="yellow"):
    time.sleep(.1)

print("***Thanks for your time!!!***")
```

  - **Expected output:**



```
/usr/bin/python3.8 /home/hp/IdeaProjects/Pakeeza_Python_project1/task2.py
Please wait to run the ProgressBar!!:  84%|████████████████████████████████████████████          | 2105/2500 [03:30<00:39,  9.97it/s]
```



```
Please wait to run the ProgressBar!!: 2505it [04:10,  9.97it/s]
***Thanks for your time!!!***

Process finished with exit code 0
```