Week_03_assignment

File_name = timer.py

```
import timeit

def timer(number, repeat):
    def wrapper(func):
        runs = timeit.repeat(func, number=number, repeat=repeat)
        print(sum(runs) / len(runs))
    return wrapper
```

File_name = sunchTest.py

```
import requests

from timer import timer


url = 'https://httpbin.org/uuid'

def fetch(session, url):
    with session.get(url) as response:
        print(response.json()['uuid'])


@timer(1,1)
def main():
    with requests.Session() as session:
        for _ in range(100):
            print(fetch(session, url))
```

File_name = multiprocessing.py

```
import requests
from multiprocessing.pool import Pool
from timer import timer

url = 'https://httpbin.org/uuid'
def fetch(session, url):
    with session.get(url) as response:
        print(response.json()['uuid'])

@timer(1,1)
def main():
    with Pool() as pool:
        with requests.Session() as session:
            pool.starmap(fetch, [(session, url) for _ in range(100)])
```

File_name = multithreading.py

```
from concurrent.futures import ThreadPoolExecutor
import requests
from timer import timer


url = 'https://httpbin.org/uuid'
def fetch(session, url):
    with session.get(url) as response:
        print(response.json()['uuid'])

@timer(1,1)
def main():
    with ThreadPoolExecutor(max_workers=10) as executor:
        with requests.Session() as session:
```

```
        executor.map(fetch, [session]* 100, [url]*100)
        executor.shutdown(wait=True)
```

File_name = asyncioTest.py

```
import aiohttp
import asyncio

from timer import timer


url = 'https://httpbin.org/uuid'

async def fetch(session, url):
    async with session.get(url) as response:
        json_response = await response.json()
        print(json_response['uuid'])


async def main():
    async with aiohttp.ClientSession() as session:
        task = [fetch(session, url) for _ in range(100)]
        await asyncio.gather(*task)
@timer(1,1)
def func():
    asyncio.run(main())
```