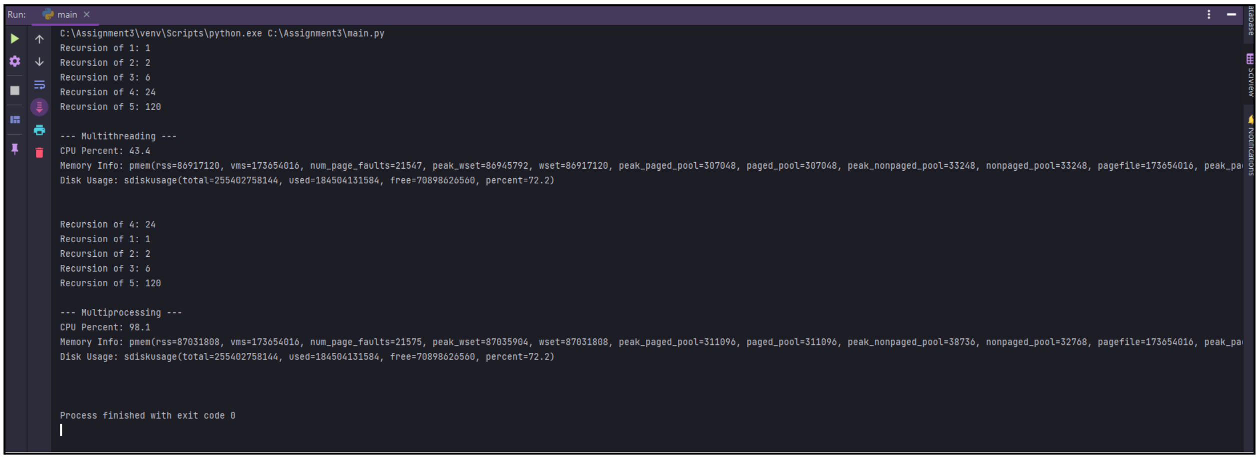
Name: Abhishek (Abi) Parekh
Assignment #3

Output:



```
C:\Assignment3\venv\Scripts\python.exe C:\Assignment3\main.py
Recursion of 1: 1
Recursion of 2: 2
Recursion of 3: 6
Recursion of 4: 24
Recursion of 5: 120

--- Multithreading ---
CPU Percent: 43.4
Memory Info: pmem(rss=86917120, vms=173654016, num_page_faults=21547, peak_wset=86945792, wset=86917120, peak_paged_pool=307048, paged_pool=307048, peak_nonpaged_pool=33248, nonpaged_pool=33248, pagefile=173654016, peak_pa
Disk Usage: sdiskusage(total=255402758144, used=184504131584, free=70898626560, percent=72.2)


Recursion of 4: 24
Recursion of 1: 1
Recursion of 2: 2
Recursion of 3: 6
Recursion of 5: 120

--- Multiprocessing ---
CPU Percent: 98.1
Memory Info: pmem(rss=87031808, vms=173654016, num_page_faults=21575, peak_wset=87035904, wset=87031808, peak_paged_pool=311096, paged_pool=311096, peak_nonpaged_pool=38736, nonpaged_pool=32768, pagefile=173654016, peak_pa
Disk Usage: sdiskusage(total=255402758144, used=184504131584, free=70898626560, percent=72.2)



Process finished with exit code 0
```

From this empirical study, it can be seen that the multiprocessed simulation takes up more computer resources compared to the multithreaded simulation.The greatest difference is seen in CPU usage, however other other computer resources do not have a large difference to show which type of simulation is more heavy on computer resources.

main.py

```python
from multithread import multiThreading
from multiprocess import multiProcess
from measure import graph_cpu_stats

if __name__ == '__main__':
   # Run both versions
   multiThreading(5)
   multiProcess(5)

   # Visualize measurements
   graph_cpu_stats()
```

multithread.py

```python
import threading
from measure import cpu_usage


def simulate(n):
   if n == 0:
       return 1
   else:
       return n * simulate(n - 1)
```

```
def compute(num):
   print(f"Recursion of {num}: {simulate(num)}")


def multiThreading(n):
   threads = []
   for i in range(1, n + 1):
       t = threading.Thread(target=compute, args=(i,))
       threads.append(t)
       t.start()

   for t in threads:
       t.join()

   cpu_usage("Multithreading")


if __name__ == '__main__':
   multiThreading(5)
```

multiprocess.py

```
import multiprocessing
from measure import cpu_usage


def simulate(n):
   if n == 0:
       return 1
   else:
       return n * simulate(n - 1)


def compute(num):
   print(f"Recursion of {num}: {simulate(num)}")


def multiProcess(n):
   processes = []
   for i in range(1, n + 1):
       p = multiprocessing.Process(target=compute, args=(i,))
       processes.append(p)
       p.start()

   for p in processes:
       p.join()

   cpu_usage("Multiprocessing")
```

```python


if __name__ == '__main__':
   multiProcess(5)
```

measure.py

```python
import psutil
import os

def cpu_usage(label):
   application = psutil.Process(os.getpid())
   print("")
   print(f"--- {label} ---")
   print(f"CPU Percent: {psutil.cpu_percent(interval=None)}")
   print(f"Memory Info: {application.memory_info()}")
   print(f"Disk Usage: {psutil.disk_usage('/')}")
   print("\n")
```

Bibliography

[1] “Difference between multithreading vs multiprocessing in python,” GeeksforGeeks, https://www.geeksforgeeks.org/difference-between-multithreading-vs-multiprocessing-in-python/ (accessed Sep. 4, 2023).

[2] S. Kumar, “Python : Threading vs Multiprocessing,” Medium, https://medium.com/@ksarthak4ever/python-threading-vs-multiprocessing-338724634bb6 (accessed Sep. 4, 2023).

[3] M. McCurdy, “Python multithreading and Multiprocessing Tutorial: Toptal®,” Toptal Engineering Blog, https://www.toptal.com/python/beginners-guide-to-concurrency-and-parallelism-in-python (accessed Sep. 4, 2023).

[4] “Multiprocessing vs Threading python,” Stack Overflow, https://stackoverflow.com/questions/3044580/multiprocessing-vs-threading-python (accessed Sep. 4, 2023).