Problem 1:

Writing 'time consuming function'

```
In [ ]: import time

        from function import time_consuming_function
```

```
In [ ]: start_time = time.time()

        time_consuming_function()

        print("Executing function 1 time: {0}".format(time.time() - start_time))
```

```
Executing function 1 time: 5.004389762878418
```

```
In [ ]: N = 8

        t1 = time.time()

        for i in range(N):
            time_consuming_function()

        t1 = time.time() - t1

        print("Executing function using for loop for N = {0} time t1: {1}".format(N,
```

```
Executing function using for loop for N = 8 time t1: 40.025580167770386
```

Parallelizeing

```
In [ ]: import multiprocessing as mp
```

```
In [ ]: def main(n, sleep_time=5):
            start_time = time.time()
            _n = [1]*n
            pool = mp.Pool(mp.cpu_count())
            _ = pool.starmap(time_consuming_function, [(i, sleep_time) for i in _n])
            pool.close()

            return time.time() - start_time
```

```
In [ ]: p = mp.cpu_count()
        tp = main(n=8)
        print("p = {}, \ntp = {}".format(p, tp))
```

```
p = 8,
tp = 5.288182020187378
```

In [ ]:
```
def get_speed_up(t, tp):
    return t/tp

def get_avg_efficiency(sp, p):
    return (sp/p)*100
```

In [ ]:
```
def _print(p, for_n, speed_up, avg_efficiency):
    print("For n = {0}, speedup is = {1}, average_efficiency is = {2}%".form
```

In [ ]:
```
def get_speedups(sleep_time=5, n1=16, n2=54, n3=400):
    tp1 = main(n=n1, sleep_time=sleep_time)
    tp2 = main(n=n2, sleep_time=sleep_time)
    tp3 = main(n=n3, sleep_time=sleep_time)

    speed_up_1 = get_speed_up(n1*sleep_time, tp1)
    speed_up_2 = get_speed_up(n2*sleep_time, tp2)
    speed_up_3 = get_speed_up(n3*sleep_time, tp3)
    return speed_up_1, speed_up_2, speed_up_3
```

When no major task is running in background

In [ ]:
```
N1, N2, N3 = 16, 54, 400
speed_up_1, speed_up_2, speed_up_3 = get_speedups(sleep_time=5, n1=N1, n2=N2
_print(p, N1, speed_up_1, get_avg_efficiency(speed_up_1, p))
_print(p, N2, speed_up_2, get_avg_efficiency(speed_up_2, p))
_print(p, N3, speed_up_3, get_avg_efficiency(speed_up_3, p))
```

```
For n = 16, speedup is = 7.79734162733867, average_efficiency is = 97.46677
034173338%
For n = 54, speedup is = 6.718746875974733, average_efficiency is = 83.9843
3594968417%
For n = 400, speedup is = 7.681804985052802, average_efficiency is = 96.022
56231316002%
```

Problem 2: Background processes case.

When several 4K videos on youtube on several tabs in the browser is running is background.

In [ ]:
```
speed_up_1, speed_up_2, speed_up_3 = get_speedups(sleep_time=5, n1=16, n2=54
_print(p, N1, speed_up_1, get_avg_efficiency(speed_up_1, p))
_print(p, N2, speed_up_2, get_avg_efficiency(speed_up_2, p))
_print(p, N3, speed_up_3, get_avg_efficiency(speed_up_3, p))
```

```
For n = 16, speedup is = 7.335206832178229, average_efficiency is = 91.6900
8540222787%
For n = 54, speedup is = 6.704647720329827, average_efficiency is = 83.8080
9650412284%
For n = 400, speedup is = 7.6722128467843005, average_efficiency is = 95.90
266058480375%
```

Conclusion: when runniging 4K videos on 4 tabs, it speedup and average efficiency slightly changed (negligible). I ran the code in MacOS. May be it optimized better.

Problem 3: Random running time scenario.

In [ ]:

```
import random
sleep_time = random.randint(1, 5)
sleep_time
```

Out[ ]:

```
1
```

In [ ]:

```
speed_up_1, speed_up_2, speed_up_3 = get_speedups(sleep_time=sleep_time, n1=
_print(p, N1, speed_up_1, get_avg_efficiency(speed_up_1, p))
_print(p, N2, speed_up_2, get_avg_efficiency(speed_up_2, p))
_print(p, N3, speed_up_3, get_avg_efficiency(speed_up_3, p))
```

```
For n = 16, speedup is = 7.135459816452579, average_efficiency is = 89.1932
4770565724%
For n = 54, speedup is = 6.550397204514992, average_efficiency is = 81.8799
650564374%
For n = 400, speedup is = 7.638220822541703, average_efficiency is = 95.477
76028177128%
```

In [ ]: