NAMA: YUNITA NUR

NPM: 21083010107

SISTEM OPERASI A

LAPORAN TUGAS 8

Soal Latihan 8:

Dengan menggunakan pemrosesan paralel buatlah program yang dapat menentukan sebuah bilangan itu ganjil atau genap! Batasan: Nilai yang dijadikan argument pada fungsi sleep () adalah satu detik, Masukkan jumlahnya satu dan berupa bilangan bulat, Masukkan adalah batas dari perulangan tersebut, Setelah perulangan selesai program menampilkan waktu eksekusi pemrosesan sekuensial dan parallel.

```
yunitanur@yunitanur:~$ nano Tugas_8.py
```

- Membuat file baru dengan nama file Tugas_8 menggunakan perintah nano

```
                    yunitanur@yunitanur: ~
Berkas  Sunting  Tampilan  Cari  Terminal  Bantuan
 GNU nano 6.2                        Tugas_8.py
from time import time, sleep
from os import getpid
from multiprocessing import cpu_count, Pool, Process
```

- Import library python yaitu dengan fungsi sebagai berikut: **Time** berfungsi untuk mengambil waktu(detik). **Sleep** berfungsi untuk memberi jeda pada waktu(detik). **Getpid** berfungsi untuk mengambil ID proses**. Cpu_count** untuk melihat jumlah cpu. Pool merupakan sebuah class pada library multiprocessing yang berfungsi untuk melakukan sebuah proses pada paralel dengan menggunakan proses sebanyak jumlah cpu pada computer. **Process** merupakan sebuah class pada library multiprocessing untuk melakukan proses pada paralel dengan menggunakan proses secara beruntun pada computer kita sendiri.

```
print("input nilai :")
nilai = int(input())

def cetak(i):
        for i in range(nilai):
                if i % 2 == 0:
                   print(f"{i+1} Bilangan Ganjil","- ID proses", getpid())
                else:
                  print(f"{i+1} Bilangan Genap","- ID proses", getpid())
        sleep(1)
print(" ")
```

- Mendeklarasikan fungsi print dengan nama input nilai yang akan digunakan agar dapat memunculkan berapa jumlah nilai yang ingin dimasukkan. Kegunaan **Def cetak** untuk mendeklarasikan mana angka yang termasuk bilangan ganjil atau bilangan genap beserta ID proses sejumlah parameter yang digunakan dan disini menggunakan metode looping for dan fungsi if else. Kemudian saya menambahkan fungsi **sleep(1)** untuk menjeda setiap satu detik nya untuk ke argumen berikutnya.

```
# multiprocessing sekuensial
print("Multiprocessing _Sekuensial")
sekuensial_awal = time()

for i in range(1):
        cetak(i)
sekuensial_akhir = time()
print(" ")
```

- Multiprocessing sekuensial ini dibuat untuk sebuah pendeklarasian dan disini metode menggunakan looping for. Syntax **"sekuensial_awal = time()"** berfungsi untuk mendapatkan waktu sebelum di eksekusi kan lalu syntax for adalah berlangsungnya proses sekuensial. Dan **range(1)** artinya output yang akan dikeluarkan nantinya sebanyak satu kali. Kemudian pada syntax **"sekuensial_akhir = time()"** untuk mendapatkan waktu setelah di eksekusi. Lalu yang terakhir adalah memanggil syntax dengan perintah **print(" ")**

```
# multiprocessing process
print("Multiprocessing_Process")
kumpulan_proses = []
process_awal = time()

for i in range(1):
       p = Process(target = cetak, args = (i,))
       kumpulan_proses.append(p)
       p.start()

for i in kumpulan_proses:
        p.join()
process_akhir = time()
print(" ")
```

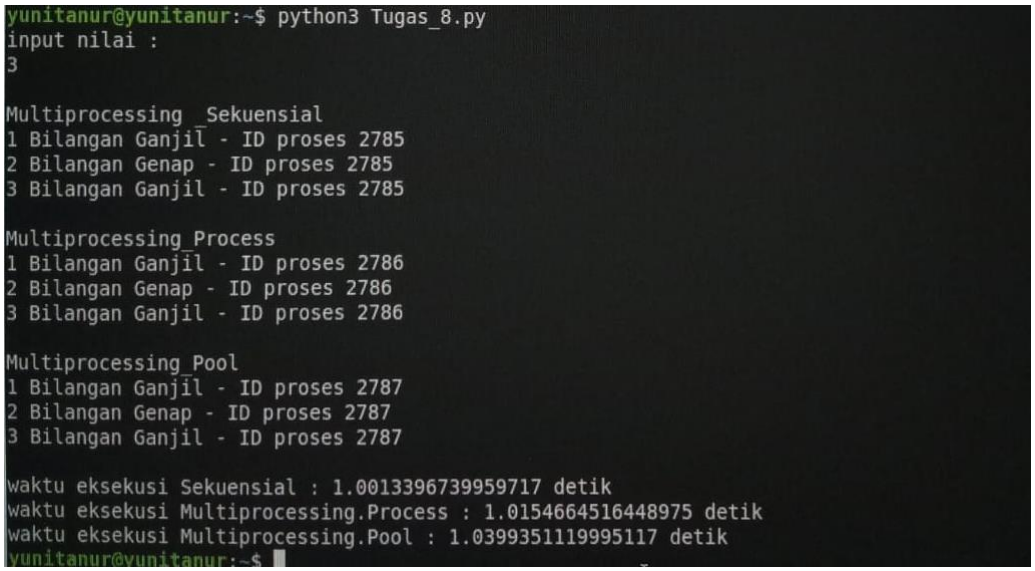
- Multiprocessing sekuensial ini dibuat untuk sebuah pendeklarasian dan disini menggunakan looping for. Dengan syntax **"proses_awal = time()"** yang berfungsi untuk mendapatkan waktu sebelum di eksekusikan lalu syntax for adalah berlangsungnya proses sekuensial. Dan **range(1**) artinya output yang akan dikeluarkan nanti sebanyak satu kali**. "for i in kumpulan_proses"** digunakan untuk menganggabungkan suatu proses-proses agar tidak loncat ke proses yang sebelumnya. Kemudian pada syntax **"proses_akhir = time()"** untuk mendapatkan waktu setelah di eksekusi. Lalu yang terakhir adalah memanggil syntax dengan perintah **print(" ")**

```
# multiprocessing pool
print("Multiprocessing_Pool")
pool_awal = time()
pool = Pool()
pool.map(cetak, range(0,1))
pool.close()
pool_akhir = time()
print(" ")
```

- Multiprocessing pool ini menggunakan fungsi .map() agar bisa memetakan panggilan fungsi cetak nya kedalam sebanyak 1 kali

```
# membandingkan waktu eksekusi
print("waktu eksekusi Sekuensial :", sekuensial_akhir - sekuensial_awal, "detik")
print("waktu eksekusi Multiprocessing.Process :", process_akhir - process_awal, "detik")
print("waktu eksekusi Multiprocessing.Pool :", pool_akhir - pool_awal, "detik")
```

- Step yang terakhir yaitu membuat sebuah lama waktu pengeksekusian itu berlangsung, caranya adalah dengan waktu akhir – waktu awal, seperti yang telah kita deklarasikan pada setiap awal dan akhir dari sebuah program multiprocessing



- Diatas merupakan output yang dihasilkan dari syntax sebelumnya