Nama : Alyssa Amorita Azzah
NPM : 21083010057
Kelas : Sistem Operasi A

LAPORAN TUGAS 8 MULTIPROCESSING
SISTEM OPERASI A

Dengan menggunakan pemrosesan paralel buatlah program yang dapat menentukan sebuah bilangan itu ganjil atau genap!

Batasan :

1. Nilai yang dijadikan argumen pada fungsi sleep() adalah satu detik.
2. Masukkan jumlahnya satu dan berupa bilangan bulat.
3. Masukkan adalah batas dari perulangan tersebut.
4. Setelah perulangan selesai program menampilkan waktu eksekusi pemrosesan sekuensial dan paralel.

Jawaban :

1. Membuat file baru

```
alyssa@alyssa-VirtualBox:~/uploadtogit/Tugas8SistemOperasi$ nano tugas_8.py
```

Buat file baru dengan command [nano tugas_8.py]. Nano merupakan command yang digunakan untuk membuat file baru sekaligus menyimpannya.

2. Menuliskan Script

```
 GNU nano 6.2                                                tugas_8.py *
#import library
from os import getpid
from time import time, sleep
from multiprocessing import Pool, Process
```

Ketikkan script untuk meng-import library yang akan digunakan, yaitu [getpid], [time], [sleep], [Pool], dan [Process].

a. [getpid] = untuk mengambil ID proses.
b. [time] = untuk mengambil waktu(detik).
c. [sleep] = untuk memberi jeda waktu(detik).
d. [Pool] = sebuah class pada library multiprocessing yang digunakan untuk melakukan pemrosesan paralel dengan menggunakan proses sebanyak jumlah CPU pada komputer.
e. [Process] = sebuah class pada library multiprocessing yang digunakan untuk melakukan pemrosesan paralel dengan menggunakan proses secara beruntun pada komputer.

```
#inisialisasi fungsi yang akan digunakan
def cetak(i):
    if (i+1)%2 == 0:
        print(i+1, "Genap - ID proses", getpid())
        sleep(1)
    else:
        print(i+1, "Ganjil - ID proses", getpid())
        sleep(1)
```

Selanjutnya, lakukan inisialisasi fungsi yang akan digunakan. Fungsi tersebut digunakan untuk mencetak angka dari variabel i beserta ID proses sejumlah parameter yang diberikan. Fungsi sleep digunakan untuk memberi jeda waktu dalam bentuk detik sebanyak parameter yang diberikan. Pada gambar di atas dituliskan [sleep(1)] yang artinya jeda waktunya adalah 1 detik.

```
#input bilangan
a = int(input("Masukkan Angka : "))
```

Ketikkan script untuk meng-input bilangan sesuai dengan permintaan soal. Inisialisasikan a sebagai inputan bilangan.

```
#sekuensial
print("\nSekuensial")
#untuk mendapatkan waktu sebelum eksekusi
sekuensial_awal = time()

#proses berlangsung
for i in range(a):
    cetak(i)

#untuk mendapatkan waktu setelah eksekusi
sekuensial_akhir = time()
```

Selanjutnya lakukan pemrosesan sekuensial yang diawali dengan [print("\nSekuensial")]. Kemudian untuk mendapatkan waktu sebelum eksekusi ketikkan script [sekuensial_awal = time()]. Seperti yang sudah dijabarkan sebelumnya, fungsi [time] digunakan untuk mengambil waktu. Selanjutnya ketikkan script loop [for … in ...] untuk memproses sekuensial sebanyak range dari a dan lanjutkan dengan fungsi [cetak]. Dan kemudian untuk mendapatkan waktu setelah eksekusi ketikkan script [sekuensial_akhir = time()].

```
#multiprocessing dengan kelas process
print("\nmultiprocessing.Process")
#untuk menampung proses-proses
kumpulan_process = []

#untuk mendapatkan waktu sebelum eksekusi
process_awal = time()

#proses berlangsung
for i in range(a):
    p = Process(target=cetak, args=(i,))
    kumpulan_process.append(p)
    p.start()

#untuk menggabungkan proses-proses agar tidak loncat ke proses sebelumnya
for i in  kumpulan_process:
    p.join()

#untuk mendapatkan waktu setelah eksekusi
process_akhir = time()
```

Selanjutnya lakukan multiprocessing dengan kelas process yang diawali dengan [print("\nmultiprocessing.Process")]. Kemudian untuk menampung proses-proses ketikkan script [kumpulan_process = []]. Selanjutnya untuk mendapatkan waktu sebelum eksekusi ketikkan script [process_awal = time()]. Seperti yang sudah dijabarkan sebelumnya, fungsi [time] digunakan untuk mengambil waktu. Selanjutnya ketikkan script loop [for … in ...] untuk memproses multiprocessing

dengan kelas process dengan ID yang berbeda-beda dan lanjutkan dengan fungsi [p.start()]. Dapat diperhatikan dengan seksama bahwa ID proses tiap memanggil fungsi cetak adalah berbeda-beda. Ini menandakan bahwa tiap pemanggilan fungsi cetak ditangani oleh satu proses saja. Kemudian untuk pemanggilan selanjutnya ditangani oleh proses yang lain. Selanjutnya ketikkan script loop [for … in …] untuk menggabungkan proses-proses. Kumpulan proses harus ditampung dan digabung menjadi satu (p.join()) agar tidak merambah ke proses selanjutnya. Dan kemudian untuk mendapatkan waktu setelah eksekusi ketikkan script [process_akhir = time()].

```
#multiprocessing dengan kelas pool
print("\nmultiprocessing.Pool")
#untuk mendapatkan waktu sebelum eksekusi
pool_awal = time()

#proses berlangsung
pool = Pool()
pool.map(cetak, range(0,a))
pool.close

#untuk mendapatkan waktu sebelum eksekusi
pool_akhir = time()
```

Kemudian lakukan multiprocessing dengan kelas Pool yang diawali dengan script [print("\nmultiprocessing.Pool")]. Selanjutnya untuk mendapatkan waktu sebelum eksekusi ketikkan script [pool_awal = time()]. Dilanjutkan dengan proses yang lain dan diakhiri dengan script [pool_akhir = time()] untuk mendapatkan waktu sebelum eksekusi.

```
#bandingkan waktu eksekusi
print("\nWaktu 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")
```

Terakhir bandingkan waktu eksekusi antara proses sekuensial, multiprocessing dengan kelas process, dan multiprocessing dengan kelas pool dalam bentuk detik. Dari hasil membandingkan ketiga proses tersebut, sudah sewajarnya proses sekuensial lebih lambat dibanding multiprocessing. Namun bukan berarti harus melakukan multiprocessing terus menerus. Cukup gunakan metode sesuai kebutuhan. Setelah selesai mengetikkan semua script, klik Ctrl + X + Y + Enter untuk menyimpan file script yang baru saja dibuat.

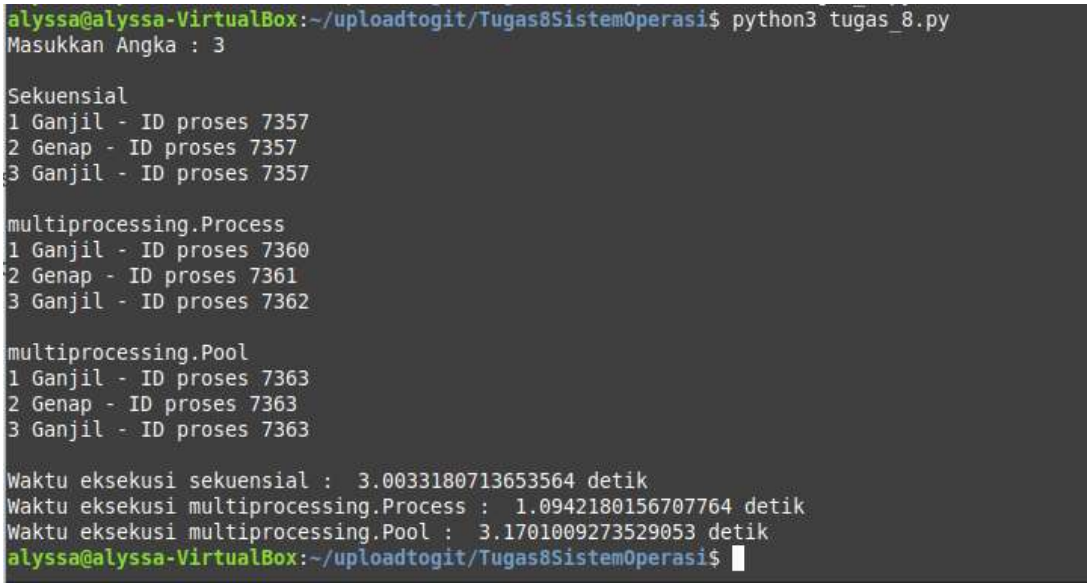
3. Output

```
alyssa@alyssa-VirtualBox:~/uploadtogit/Tugas8SistemOperasi$ python3 tugas_8.py
```

Untuk melihat output dari file script multiprocessing [tugas_8.py], ketikkan command [python3 tugas_8.py].

```
alyssa@alyssa-VirtualBox:~/uploadtogit/Tugas8SistemOperasi$ python3 tugas_8.py
Masukkan Angka : 3
```

Input angka sesuai permintaan soal yaitu 3.



Kemudian akan muncul output seperti gambar di atas, sesuai dengan permintaan soal.