Nama : Naomi Dwi Anggraini
NPM : 21083010010
Kelas : B

Tugas 8
Multiprocessing

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 paralel.

Pertama yaitu gunakan command nano untuk membuat file Tugas_8.py

```
naomi@naomi-VirtualBox:~$ nano Tugas 8.py
```

Lalu ketikan import library yang akan digunakan, yaitu **getpid** digunakan untuk mengambil ID proses, **time** digunakan untuk mengambil waktu(detik), **sleep** digunakan untuk memberi jeda waktu(detik), **cpu_count** digunakan untuk melihat jumlah CPU, **Pool** adalah sebuah class pada library multiprocessing yang digunakan untuk melakukan pemrosesan paralel dengan menggunakan proses sebanyak jumlah CPU pada komputer, **Process** adalah sebuah class pada library multiprocessing yang digunakan untuk melakukan pemrosesan paralel dengan menggunakan proses secara beruntun pada komputer.

```
File   Edit   View   Search   Terminal   Help
  GNU nano 6.2                          Tugas_8.py
# import library
from os import getpid
from time import time, sleep
from multiprocessing import cpu_count, Pool, Process
```

Selanjutnya melakukan inisialisasi fungsi yang akan digunakan. Fungsi di atas digunakan untuk mencetak angka dari variabel i beserta ID proses sejumlah parameter yang diberikan. Fungsi sleep digunakan untuk memberi jeda waktu(detik) sebanyak parameter yang diberikan. Sleep(1) artinya memberi jeda waktu selama 1 detik.

```
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)
```

Inisialisasikan n sebagai inputan range angka dengan tipe integer.

```
n = int(input("Masukkan Range Angka : "))
```

Selanjutnya, melakukan pemrosesan sekuensial. Gunakan fungsi time untuk mendapatkan waktu sebelum dan sesudah eksekusi. Lalu loop for...in digunakan untuk memproses sekuensial sebanyak range(n). Selanjutnya, pada loop masukan fungsi cetak.

```
print("Sekuensial")
# UNTUK MENDAPATKAN WAKTU SEBELUM EKSEKUSI
sekuensial_awal = time()

# PROSES BERLANGSUNG
for i in range(n):
    cetak(i)

# UNTUK MENDAPATKAN WAKTU SETELAH EKSEKUSI
sekuensial_akhir = time()
```

Melakukan multiprocessing dengan kelas Process. Menggunakan fungsi time untuk mendapatkan waktu sebelum dan sesudah eksekusi. Pada loop for..in masukkan fungsi cetak, tetapi dengan ID proses yang berbeda-beda. Ini menandakan bahwa tiap pemanggilan fungsi cetak ditangani oleh satu proses saja. Kemudian untuk pemanggilan selanjut'nya ditangani oleh proses yang lain. Kumpulan proses harus ditampung dan digabung menjadi satu(p.join())agar tidak merambah ke proses selanjutnya. Silahkan eksekusi file berikut pada terminal anda, maka anda akan paham apa yang saya maksudkan.

```
print("multiprocessing.Process")
# UNTUK MENAMPUNG PROSES-PROSES
kumpulan_proses = []

# UNTUK MENDAPATKAN WAKTU SEBELUM EKSEKUSI
process_awal = time()
# PROSES BERLANGSUNG
for i in range(n):
    p = Process(target=cetak, args=(i,))
    kumpulan_proses.append(p)
    p.start()

# UNTUK MENGGABUNGKAN PROSES-PROSES AGAR TIDAK LONCAT KE PROSES SEBELUMNYA
for i in kumpulan_proses:
    p.join()

# UNTUK MENDAPATKAN WAKTU SETELAH EKSEKUSI
process_akhir = time()
```

Melakukan multiprocessing dengan kelas pool. Gunakan fungsi time untuk mendapatkan waktu sebelum dan sesudah eksekusi. Fungsi map() digunakan untuk memetakan pemanggilan fungsi cetak ke dalam CPU yang memiliki komputer sebanyak n kali.

```
print("multiprocessing.Pool")
# UNTUK MENDAPATKAN WAKTU SEBELUM EKSEKUSI
pool_awal = time()

# PROSES BERLANGSUNG
pool = Pool()
```

```
# PROSES BERLANGSUNG
pool = Pool()
pool.map(cetak, range(0,n))
pool.close

# UNTUK MENDAPATKAN WAKTU SEBELUM EKSEKUSI
pool_akhir = time()
```

Tampilkan waktu eksekusi pemrosesan sekuensial dan paralel dengan mengurangi waktu sesudah dengan waktu sebelum eksekusi.

```
print("Waktu eksekusi sekuensial :", sekuensial_akhir - sekuensial_awal, "detik")
print("Waktu eksekusi multiprocessing.Process :", process_akhir - process_awal, "
print("Waktu eksekusi multiprocessing.Pool :", pool_akhir - pool_awal, "detik")
```

Selanjutnya baca script yang sudah dituliskan dengan python3 Tugas_8.py Lalu masukan range angka 3. Sehingga batas dari perulangan pemrosesan sekuensial dan paralel adalah sebanyak 3. Lalu juga ditunjukan waktu dari setiap eksekusinya.