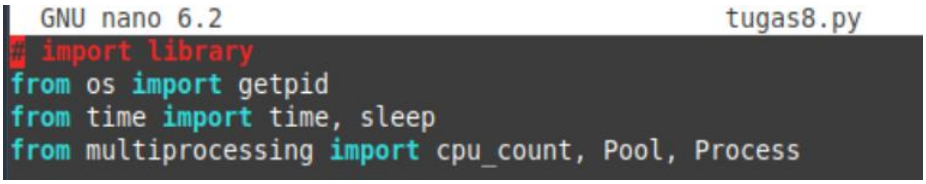
Nama : Enzelica Vica Christina

NPM : 21083010114

Kelas : Sistem Operasi B

TUGAS MULTIPROCESSING

Langkah pertama adalah import semua library yang dibutuhkan.



- **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.

Selanjutnya adalah membuat perintah untuk menerima input dan menampilkan output. Variabel batas adalah batas perulangan.

```
# input
batas = int(input("Input:\n"))

print("\nOutput:")
```

Inisialisasi fungsi. Disini saya membuat fungsi untuk menentukan bilangan ganjil atau genap.

- Jika i habis dibagi 2, cetak sesuai dengan syntax di bawah. Jika tidak, cetak sesuai syntax di bawah ini.
- sleep digunakan untuk membuat jeda selama 1 detik.

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

Fungsi sekuensial.

- Membuat perintah untuk mendapatkan waktu sebelum eksekusi.
- Perulangan proses. Untuk i di dalam batas (input user), eksekusi fungsi tentukan(i).
- Membuat perintah untuk mendapatkan waktu setelah eksekusi.

```
# sekuensial
print("\nSekuensial")
sekuensial_awal = time()

for i in range(batas):
   tentukan(i)

sekuensial_akhir = time()
```

Fungsi multiprocessing.Process

- Membuat variabel untuk menyimpan sekumpulan proses.
- Membuat perintah untuk mendapatkan waktu sebelum proses.
- Proses berlangsung. Di dalam proses ini terjadi penambahan nilai di dalam variabel yang menyimpan sekumpulan proses.
- Penggabungan sekumpulan proses agar tidak loncat ke proses sebelumnya.
- Membuat perintah untuk mendapatkan waktu setelah proses.

```
# multiprocessing.Process
print("\nmultiprocessing.Process")
kumpulan_proses = []

process_awal = time()

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

for i in kumpulan_proses:
   p.join()

process_akhir = time()
```

Fungsi multiprocessing.Pool

- Membuat perintah untuk mendapatkan waktu sebelum proses.
- Proses berlangsung.
- Membuat perintah untuk mendapatkan waktu setelah proses.

```
# multiprocessing.Pool
print("\nmultiprocessing.Pool")
pool_awal = time()

pool = Pool()
pool.map(tentukan, range(batas))
pool.close

pool_akhir = time()
```

Ini adalah syntax untuk menampilkan perbandingan waktu antara ketiga metode di atas. Untuk mengetahui wakttu yang digunakan adalah dengan menghitung selisih waktu setelah proses dengan sebelum proses.

```
# perbandingan waktu
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")
```

Ini adalah output nya.

```
enzelica@enzelica-VirtualBox:~/pertemuan_8$ python3 tugas8.py
Input:
3

Output:

Sekuensial
1 Genap - ID proses 3181
2 Ganjil - ID proses 3181
3 Genap - ID proses 3181

multiprocessing.Process
1 Genap - ID proses 3184
2 Ganjil - ID proses 3185
3 Genap - ID proses 3186

multiprocessing.Pool
1 Genap - ID proses 3187
2 Ganjil - ID proses 3187
3 Genap - ID proses 3187

Waktu eksekusi sekuensial : 3.0034048557281494 detik
Waktu eksekusi multiprocessing.Process : 1.0245170593261719 detik
Waktu eksekusi multiprocessing.Pool : 3.0452423095703125 detik
```