

Nama : Ahmad Wafi Fathurrahman

NIM : 21083010011

Kelas : Sistem Operasi (A)

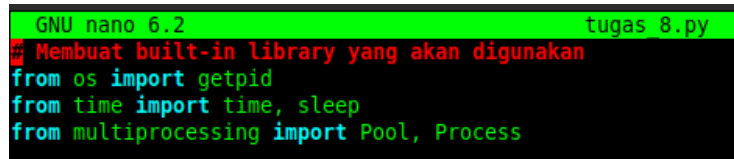
## Tugas 8

### Latihan Soal

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

### Penyelesaian

1. Membuat file dengan melakukan perintah nano
2. Setelah itu, menuliskan sebuah script dengan rincian sebagai berikut.
  - a. Membuat built-in library



```
GNU nano 6.2      tugas 8.py
# Membuat built-in library yang akan digunakan
from os import getpid
from time import time, sleep
from multiprocessing import Pool, Process
```

Langkah pertama dalam menuliskan sebuah script adalah mengimport beberapa built-in library yang akan digunakan dalam program.

- 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.

b. Inisialisasi function

```
# Menginisiasi Function yang nantinya akan digunakan
def cetak(i):
    a = i % 2
    if a == 0:
        print(i, "Angka Genap", "- ID Proses", getpid())
    else:
        print(i, "Angka Ganjil", "- ID Proses", getpid())
        sleep(1)
```

Disini, membuat function cetak dengan parameter i (cetak(i)). Setelahnya, mendeklarasikan variabel  $a = i \% 2$  yang berfungsi untuk menghitung modulus dari variabel i dan menentukan apakah termasuk bilangan genap atau ganji dengan melakukan kondisi if dan else sebagai berikut.

- Untuk kondisi if  $i \% 2 == 0$ , maksudnya adalah apabila hasil perhitungan modulus dari variabel i adalah 0, maka bilangan tersebut termasuk ke dalam bilangan atau angka genap.
- Untuk kondisi else, maksudnya adalah apabila hasil perhitungan modulus dari variabel i negasi atau tidak sama 0, maka bilangan tersebut termasuk ke dalam bilangan atau angka ganjil.
- Terakhir, terdapat function sleep(1) yang digunakan untuk memberikan jeda waktu selama satu detik sebelum mengeksekusi perintah selanjutnya.

c. Input bilangan

```
# Membuat inputan untuk batasan
angka = int(input("Masukkan batasan perulangan: "))
```

Membuat field inputan dengan variabel bernama angka dan data yang bertipe integer, yang berguna untuk sebagai batasan pada program ketika nanti melakukan perulangan.

d. Sekuensial

```
# Sekuensial
print("\nPemrosesan Sekuensial")
sekuensial_awal = time()

for i in range(1, angka + 1):
    cetak(i)

sekuensial_akhir = time()
```

Diawal, dilakukan print (pemrosesan sekuensial) yang bertujuan untuk menandakan bahwa program berikut merupakan bagian dari pemrosesan sekuensial. Selanjutnya mendeklarasikan variabel sekuensial\_awal yang di dalamnya terdapat function time dengan tujuan untuk mengambil waktu (detik) pada awal pemrosesan.

Lalu, terdapat perulangan looping for yang di dalamnya memanggil function cetak pada argument variabel i. Untuk looping sendiri, terdapat function range yang di dalamnya terdapat angka 1, berfungsi sebagai landasan yang dimana angka satu merupakan awal dari bilangan bulat positif dan diakhiri dengan angka + 1 untuk mendapatkan batasan dari variabel angka yang telah di inputkan oleh user sebelumnya.

Terakhir, mendeklarasikan variabel `sekuensial_akhir` yang di dalamnya terdapat function `time`, berguna untuk mengambil waktu(detik) pada saat pemrosesan telah berakhir.

e. Kelas Process

```
# Kelas Process
print("\nMultiprocessing dengan kelas Process")

kumpulan_proses = []
process_awal = time()

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

for i in kumpulan_proses:
    p.join()

process_akhir = time()
```

Pertama, dilakukan `print` (`Multiprocessing dengan kelas Process`) yang bertujuan untuk memberi tahu bahwa program berikut merupakan bagian dari `Multiprocessing` dengan kelas `Process`. Lalu, menginisialisasikan sebuah list kosong yang terdapat dalam variabel `kumpulan_proses`. Selanjutnya, mendeklarasikan variabel `process_awal` yang di dalamnya terdapat function `time` dengan tujuan untuk mengambil waktu (detik) pada awal pemrosesan.

Lalu, melakukan perulangan looping `for` yang pertama untuk variabel `i` dalam `range(1, angka + 1)` yang didalamnya memanggil dan menggunakan kelas `Process` dengan argument `target=cetak, args=(i,)`. Argument tersebut akan disimpan di dalam variabel `p`. Kemudian, isi dari variabel `p` tersebut akan dimasukkan ke dalam list bernama `kumpulan_proses` yang telah dibuat sebelumnya. Setelah nilai dari variabel `p` telah dimasukkan ke dalam list, digunakan lah method `start` untuk memulai pemrosesan looping.

Untuk looping `for` yang kedua, digunakan method `join` untuk menggabungkan kumpulan proses yang telah ditampung dalam variabel `p` agar nantinya tidak berlanjut ke proses selanjutnya. Terakhir, mendeklarasikan variabel `process_akhir` yang di dalamnya terdapat function `time`, berguna untuk mengambil waktu(detik) pada saat pemrosesan telah berakhir.

f. Kelas Pool

```
# Kelas Pool
print("\nMultiprocessing dengan Kelas Pool")
pool_awal = time()

pool = Pool()
pool.map(cetak, range(1, angka + 1))
pool.close()

pool_akhir = time()
```

Pertama, dilakukan print (Multiprocessing dengan kelas Pool) yang bertujuan untuk memberi tahu bahwa program berikut merupakan bagian dari Multiprocessing dengan kelas Pool. Selanjutnya, mendeklarasikan variabel pool\_awal yang di dalamnya terdapat function time dengan tujuan untuk mengambil waktu (detik) pada awal pemrosesan.

Kemudian, menginisialisasikan kelas Pool ke dalam variabel pool. Lalu, digunakan method map terhadap variabel pool yang bertujuan untuk memetakan pemanggilan function cetak ke dalam 4 CPU sebanyak yang terdapat di dalam function range. Setelahnya, digunakan method close untuk memberhentikan pemberian tugas pada variabel atau kelas Pool. Terakhir, mendeklarasikan variabel pool\_akhir yang di dalamnya terdapat function time, berguna untuk mengambil waktu(detik) pada saat pemrosesan telah berakhir.

g. Perbandingan waktu pada saat eksekusi

```
# Perbandingan Waktu pada saat Eksekusi
print("\nSekuensial      :", sekuensial_akhir - sekuensial_awal, "detik")
print("Kelas Process    :", process_akhir - process_awal, "detik")
print("Kelas Pool       :", pool_akhir - pool_awal, "detik")
```

Pada proses ini, dilakukan perbandingan waktu pada saat eksekusi dengan cara waktu proses akhir – waktu proses dimulai.

### 3. Output dari script

```
awf01@awf01-VirtualBox:~/Tugas_8$ python3 tugas_8.py
Masukkan batasan perulangan: 3

Pemrosesan Sekuensial
1 Angka Ganjil - ID Proses 6052
2 Angka Genap - ID Proses 6052
3 Angka Ganjil - ID Proses 6052

Multiprocessing dengan kelas Process
1 Angka Ganjil - ID Proses 6053
3 Angka Ganjil - ID Proses 6055
2 Angka Genap - ID Proses 6054

Multiprocessing dengan Kelas Pool
1 Angka Ganjil - ID Proses 6056
2 Angka Genap - ID Proses 6056
3 Angka Ganjil - ID Proses 6056

Sekuensial      : 2.002286195755005 detik
Kelas Process   : 1.0458900928497314 detik
Kelas Pool      : 2.0828537940979004 detik
awf01@awf01-VirtualBox:~/Tugas_8$
```

- Di inputkan bilangan 3 sebagai batasan perulangan
- Pada pemrosesan sekuensial dan multiprocessing dengan pool dapat dilihat bahwa setiap bilangan yang dicetak melalui function cetak memiliki ID proses yang sama, yakni 6052 dan 6056. Hal tersebut berarti bahwa pemrosesan function cetak tersebut dilakukan oleh satu proses yang sama.
- Namun, pada Multiprocessing dengan kelas Process dapat dilihat bahwa setiap bilangan yang dicetak melalui function cetak ini memiliki ID proses yang berbeda, tetapi secara berurutan. Hal tersebut berarti bahwa setiap pemanggilan function cetak ditangani oleh satu proses saja.
- Terakhir, terdapat perbandingan waktu yang dimana multiprocessing dengan kelas Process memiliki durasi waktu yang singkat jika dibandingkan dengan yang lainnya.