

# Tugas 8

## Multiprocessing

Nama : Muhamad Haris Hartanto

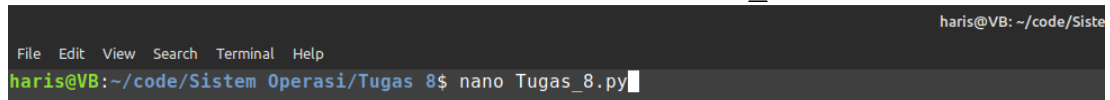
NPM : 21083010045

Kelas : Sistem Operasi – A

### A. Latihan Soal

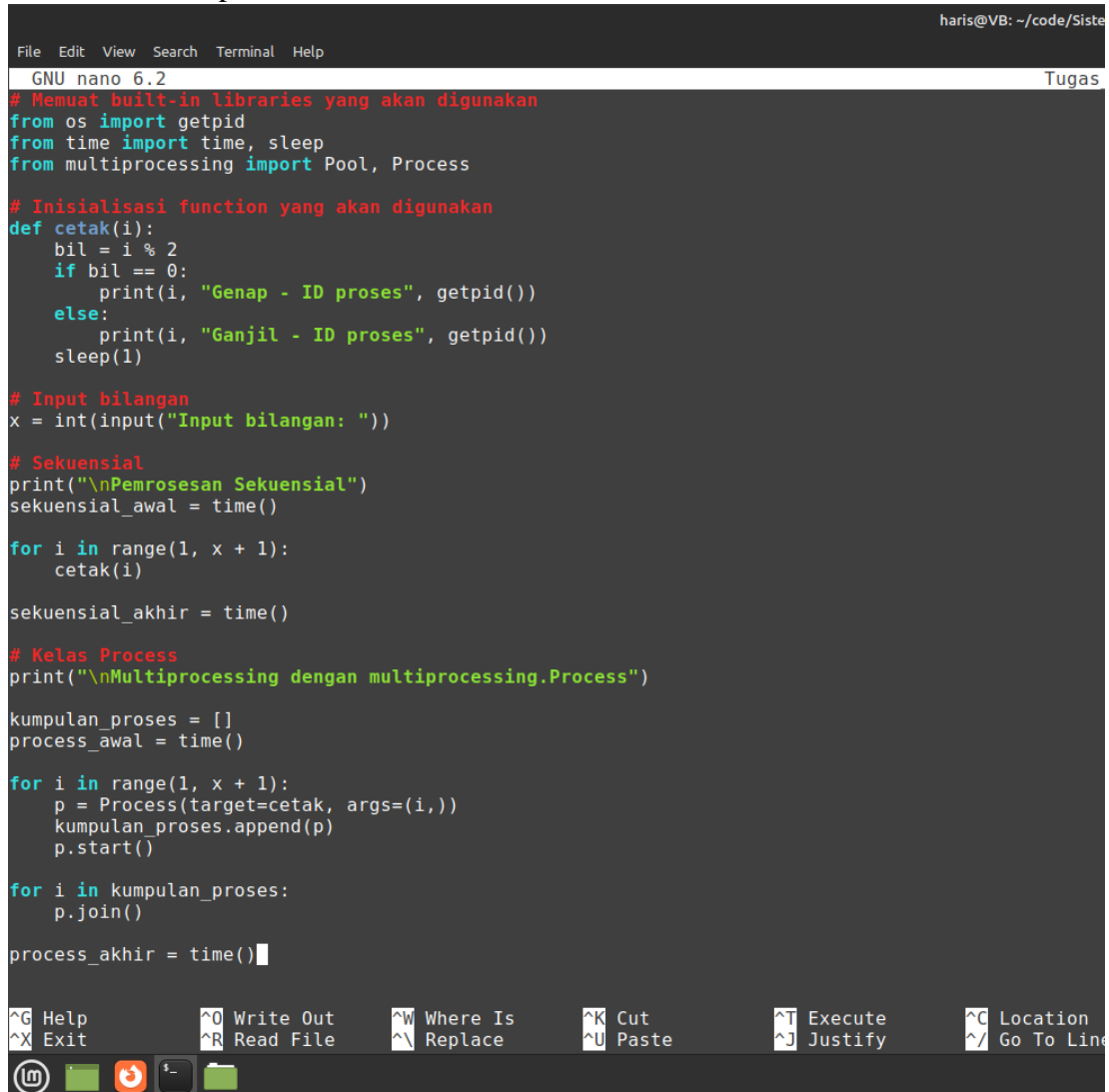
#### 1. Membuat File

Dilakukan dengan menjalankan command `nano Tugas_8.py`.



```
haris@VB: ~/code/Sistem Operasi/Tugas 8$ nano Tugas_8.py
```

#### 2. Menuliskan Script



```
haris@VB: ~/code/Siste
Tugas
GNU nano 6.2
# Memuat built-in libraries yang akan digunakan
from os import getpid
from time import time, sleep
from multiprocessing import Pool, Process

# Inisialisasi function yang akan digunakan
def cetak(i):
    bil = i % 2
    if bil == 0:
        print(i, "Genap - ID proses", getpid())
    else:
        print(i, "Ganjil - ID proses", getpid())
        sleep(1)

# Input bilangan
x = int(input("Input bilangan: "))

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

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

sekuensial_akhir = time()

# Kelas Process
print("\nMultiprocessing dengan multiprocessing.Process")

kumpulan_proses = []
process_awal = time()

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

for i in kumpulan_proses:
    p.join()

process_akhir = time()
```

```
haris@VB: ~/code/Siste
File Edit View Search Terminal Help
GNU nano 6.2 Tugas
x = int(input("Input bilangan: "))

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

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

sekuensial_akhir = time()

# Kelas Process
print("\nMultiprocessing dengan multiprocessing.Process")

kumpulan_proses = []
process_awal = time()

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

for i in kumpulan_proses:
    p.join()

process_akhir = time()

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

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

pool_akhir = time()

# Perbandingan 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")

^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^I Justify
              ^_           ^_           ^_           ^_           ^_ Location
              ^_           ^_           ^_           ^_           ^_ Go To Line
```

### Penjelasan:

Pada script ini berisi program yang berfungsi untuk mencetak bilangan bulat positif beserta tipe (ganjil/genap) berdasarkan batasan bilangan yang ditentukan oleh user dengan pemrosesan sekuensial, multiprocessing dengan kelas Process, dan multiprocessing dengan kelas Pool.

#### ▪ Line 2-4 (Memuat built-in libraries yang akan digunakan)

Dilakukan import beberapa built-in libraries yang akan digunakan dalam program, yaitu:

1. `getpid`  
Merupakan sebuah function yang mengembalikan (return) ID proses yang sedang berjalan.
2. `time`  
Merupakan sebuah function yang berfungsi untuk mengambil waktu (detik).
3. `sleep`  
Merupakan sebuah function yang berfungsi untuk menangguhkan eksekusi perintah dalam jumlah waktu (detik) yang diberikan.
4. `Process`

Merupakan sebuah class pada library multiprocessing yang digunakan untuk melakukan pemrosesan paralel dengan menggunakan proses secara beruntun pada komputer

#### 5. Pool

Merupakan sebuah class pada library multiprocessing yang digunakan untuk melakukan pemrosesan paralel dengan menggunakan proses sebanyak jumlah CPU yang terdapat pada komputer.

#### ▪ Line 7-13 (Inisialisasi function yang akan digunakan)

Dibuat sebuah function cetak dengan parameter `i` (`cetak(i)`).

- `bil = i % 2`

Dilakukan perhitungan modulus 2 (...%2) terhadap nilai variabel `i` untuk memeriksa bilangan merupakan bilangan genap atau ganjil. Hasil perhitungan disimpan oleh variabel `bil`.

- `if bil == 0`

Apabila hasil perhitungan modulus adalah 0, maka bilangan (`i`) merupakan bilangan genap.

Dicetak nilai variabel `i`, kalimat “Genap – ID Proses”, dan ID proses yang didapatkan melalui penggunaan function `getpid()`.

- `else`

Apabila hasil perhitungan modulus tidak sama dengan 0, maka bilangan (`i`) merupakan bilangan ganjil.

Dicetak nilai variabel `i`, kalimat “Ganjil – ID Proses”, dan ID proses yang didapatkan melalui penggunaan function `getpid()`.

- `sleep(1)`

Digunakan function `sleep(1)` untuk memberikan jeda selama 1 detik sebelum dilanjutkan untuk eksekusi perintah selanjutnya.

#### ▪ Line 16 (Input bilangan)

Dibuat sebuah field input dengan batasan tipe data integer untuk user dengan teks “Input bilangan: ” yang disimpan dalam variabel `x`. Tujuan dari field input ini adalah untuk menginputkan bilangan yang digunakan sebagai batasan pada program.

#### ▪ Line 19-25 (Sekuensial)

- Dilakukan cetak kalimat “\nPemrosesan Sekuensial” untuk menandakan bahwa program yang berjalan di bawah kalimat ini merupakan pemrosesan sekuensial. \n digunakan untuk mencetak baris baru.

- `sekuensial_awal = time()`

Digunakan function `time()` untuk mengambil waktu (detik) yang disimpan dalam variabel `sekuensial_awal`.

- For loop

Dilakukan perulangan for untuk `i` dalam `range(1, x + 1)` yang di dalamnya dipanggil dan digunakan function `cetak` dengan argument variabel `i`.

\*function `range()` mengembalikan urutan angka yang secara default dimulai dari 0, bertambah dengan interval 1, dan akan berhenti sebelum dari angka yang ditetapkan. Oleh karena itu pada perulangan di atas, `range()` diawali dengan 1 karena bilangan bulat positif dimulai dari bilangan 1, dan diakhiri  $x + 1$  agar didapatkan batasan bilangan sesuai dengan yang diinputkan oleh user.

- `sekuensial_akhir = time()`  
Digunakan function `time()` untuk mengambil waktu (detik) yang disimpan dalam variabel `sekuensial_akhir`.
- Line 28-41 (Kelas Process)
  - Dilakukan cetak kalimat “\nMultiprocessing dengan multiprocessing.Process” untuk menandakan bahwa program yang berjalan di bawah kalimat ini merupakan multiprocessing dengan kelas Process. \n digunakan untuk mencetak baris baru.
  - `process_awal = time()`  
Digunakan function `time()` untuk mengambil waktu (detik) yang disimpan dalam variabel `process_awal`.
  - Diinisialisasikan sebuah list kosong (`[]`) yang dengan nama variabel `kumpulan_proses`.
  - For loop (1)  
Dilakukan perulangan for untuk `i` dalam `range(1, x + 1)` yang di dalamnya dipanggil dan digunakan kelas Process dengan argument `target=cetak` dan `args=(i,)` yang disimpan dalam variabel `p`. Setelah itu, untuk setiap nilai variabel `p` dimasukkan ke dalam list `kumpulan_proses`. Digunakan method `start` terhadap variabel `p` (`p.start()`) untuk memulai proses.
  - For loop (2)  
Digunakan method `join()` terhadap variabel `p` (`p.join()`) untuk menggabungkan kumpulan proses yang telah ditampung agar tidak merambah ke proses selanjutnya.
  - `process_akhir = time()`  
Digunakan function `time()` untuk mengambil waktu (detik) yang disimpan dalam variabel `process_akhir`.
- Line 44-51 (Kelas Pool)
  - Dilakukan cetak kalimat “\nMultiprocessing dengan multiprocessing.Pool” untuk menandakan bahwa program yang berjalan di bawah kalimat ini merupakan multiprocessing dengan kelas Pool. \n digunakan untuk mencetak baris baru.
  - `pool_awal = time()`  
Digunakan function `time()` untuk mengambil waktu (detik) yang disimpan dalam variabel `pool_awal`.
  - Diinisialisasikan kelas `Pool()` yang disimpan dalam variabel `pool`.  
Digunakan method `map()` terhadap variabel `pool` (`pool.map()`) untuk memetakan pemanggilan function `cetak` ke dalam 4 CPU sebanyak yang bilangan yang berada pada `range(1, x + 1)`.

Digunakan method `close()` terhadap variabel `pool` (`pool.close()`) untuk mencegah tugas (task) lain dikirim ke dalam `pool`. Setelah semua tugas (task) selesai, proses akan keluar.

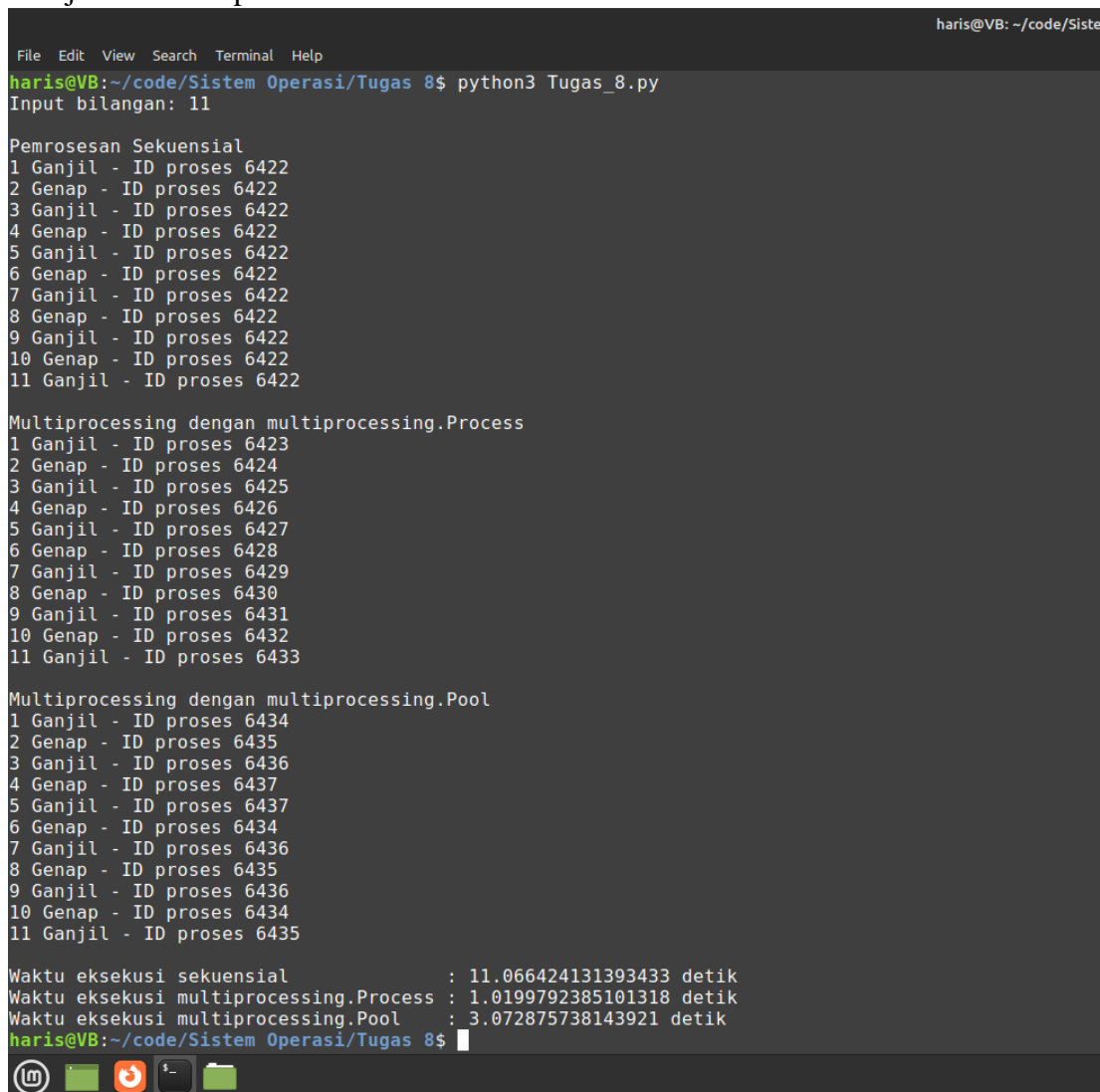
- `pool_akhir = time()`

Digunakan function `time()` untuk mengambil waktu (detik) yang disimpan dalam variabel `pool_akhir`.

- Line 54-56 (Perbandingan waktu eksekusi)

Dilakukan cetak nilai durasi waktu untuk setiap pemrosesan yang didapatkan melalui perhitungan antara waktu awal setiap proses dikurangi dengan waktu akhir setiap proses seperti yang tertera pada gambar di atas.

### 3. Menjalankan Script



```
haris@VB: ~/code/Sistem Operasi/Tugas 8$ python3 Tugas_8.py
Input bilangan: 11

Pemrosesan Sekuensial
1 Ganjil - ID proses 6422
2 Genap - ID proses 6422
3 Ganjil - ID proses 6422
4 Genap - ID proses 6422
5 Ganjil - ID proses 6422
6 Genap - ID proses 6422
7 Ganjil - ID proses 6422
8 Genap - ID proses 6422
9 Ganjil - ID proses 6422
10 Genap - ID proses 6422
11 Ganjil - ID proses 6422

Multiprocessing dengan multiprocessing.Process
1 Ganjil - ID proses 6423
2 Genap - ID proses 6424
3 Ganjil - ID proses 6425
4 Genap - ID proses 6426
5 Ganjil - ID proses 6427
6 Genap - ID proses 6428
7 Ganjil - ID proses 6429
8 Genap - ID proses 6430
9 Ganjil - ID proses 6431
10 Genap - ID proses 6432
11 Ganjil - ID proses 6433

Multiprocessing dengan multiprocessing.Pool
1 Ganjil - ID proses 6434
2 Genap - ID proses 6435
3 Ganjil - ID proses 6436
4 Genap - ID proses 6437
5 Ganjil - ID proses 6437
6 Genap - ID proses 6434
7 Ganjil - ID proses 6436
8 Genap - ID proses 6435
9 Ganjil - ID proses 6436
10 Genap - ID proses 6434
11 Ganjil - ID proses 6435

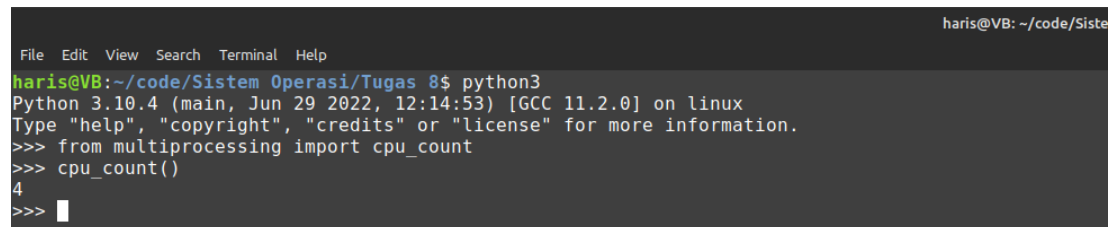
Waktu eksekusi sekuensial           : 11.066424131393433 detik
Waktu eksekusi multiprocessing.Process : 1.0199792385101318 detik
Waktu eksekusi multiprocessing.Pool   : 3.072875738143921 detik
haris@VB:~/code/Sistem Operasi/Tugas 8$
```

- Diinputkan bilangan 11 sebagai batasan oleh user.
- Pada pemrosesasan sekuensial dapat dilihat bahwa untuk setiap bilangan yang dicetak melalui function cetak memiliki ID proses yang sama (6422). Hal tersebut

menandakan bahwa untuk semua pemanggilan function `cetak` ditangani oleh satu proses yang sama.

- Pada multiprocessing dengan kelas `Process` dapat dilihat bahwa untuk setiap bilangan yang dicetak melalui function `cetak` memiliki ID proses yang berbeda danurut (6423 – 6433). Hal ini mendandakan bahwa untuk setiap pemanggilan function `cetak` ditangani oleh satu proses saja.
- Pada multiprocessing dengan kelas `Pool` dapat dilihat bahwa untuk setiap bilangan yang dicetak melalui function `cetak` memiliki ID proses yang berbeda, tetapi perbedaan ID proses tersebut hanya terbatas pada 4 ID proses (6434, 6435, 6436, dan 6437) yang berulang dengan tidak urut. Hal tersebut terjadi karena pada komputer yang digunakan untuk menjalankan script tersebut hanya memiliki 4 CPU.

\*Untuk mengetahui jumlah CPU yang terdapat pada komputer dapat digunakan function `cpu_count()` yang terdapat dalam library multiprocessing sebagai berikut:



```
File Edit View Search Terminal Help
haris@VB: ~/code/Siste
haris@VB:~/code/Sistem Operasi/Tugas 8$ python3
Python 3.10.4 (main, Jun 29 2022, 12:14:53) [GCC 11.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from multiprocessing import cpu_count
>>> cpu_count()
4
>>> 
```

- Pada bagian akhir ditampilkan durasi waktu eksekusi untuk setiap jenis pemrosesan yang telah dijalankan. Dapat dilihat bahwa multiprocessing dengan kelas `Process` memiliki durasi waktu yang paling singkat dibandingkan dengan proses lainnya dan pemrosesan sekuensial menjadi proses yang memiliki durasi waktu terlama.