

Nama : Amanda Aulia

NPM : 21083010048

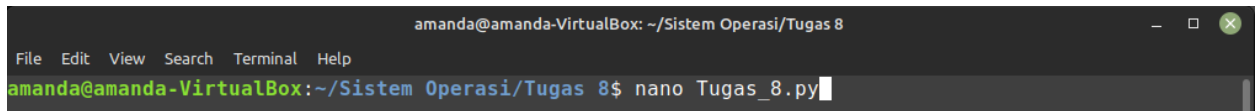
Kelas : Sistem Operasi A

Tugas 8

Multiprocessing

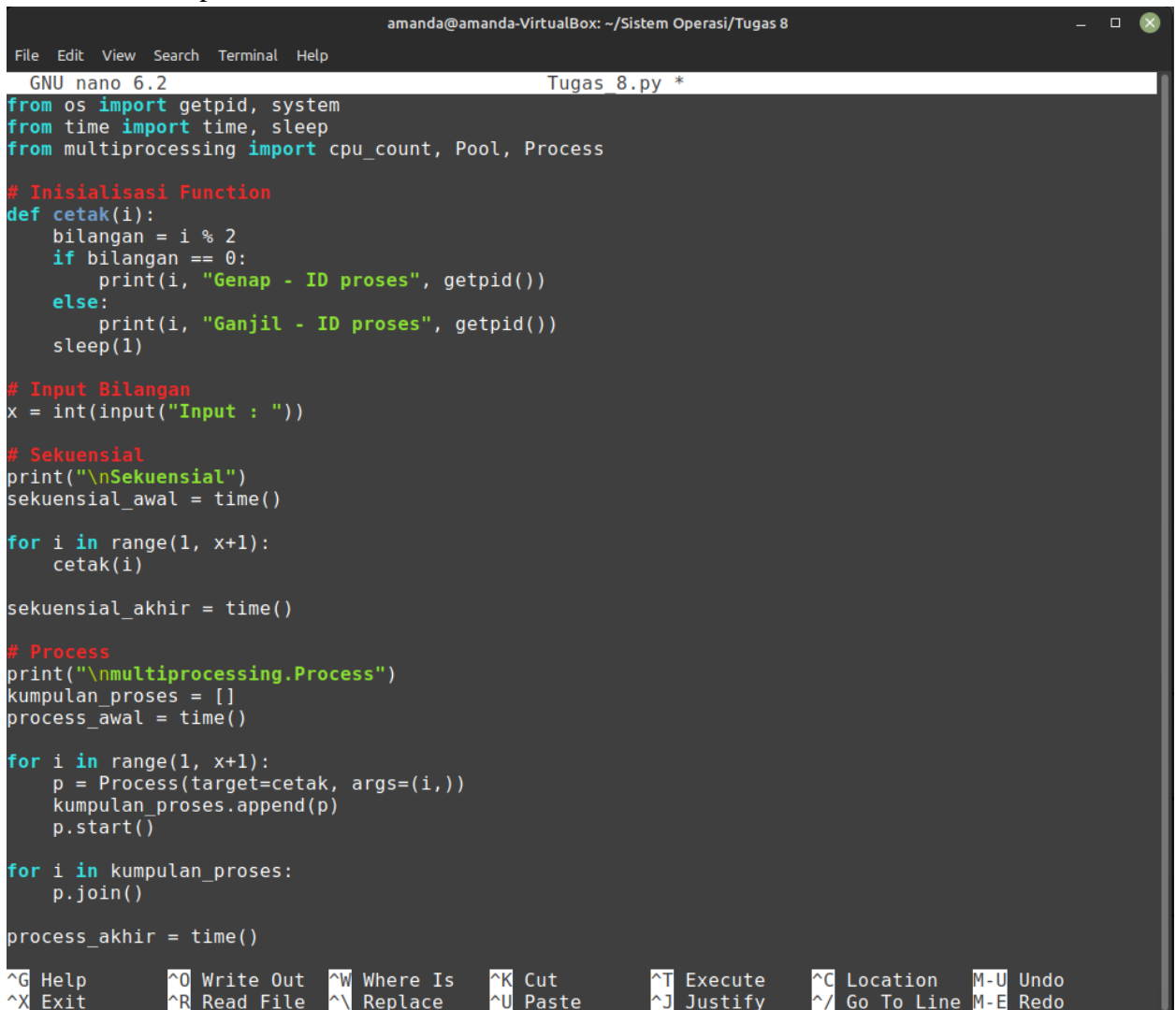
A. Latihan Soal

1. Membuat File



```
amanda@amanda-VirtualBox: ~/Sistem Operasi/Tugas 8
File Edit View Search Terminal Help
amanda@amanda-VirtualBox:~/Sistem Operasi/Tugas 8$ nano Tugas_8.py
```

2. Menuliskan Script



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

# Inisialisasi Function
def cetak(i):
    bilangan = i % 2
    if bilangan == 0:
        print(i, "Genap - ID proses", getpid())
    else:
        print(i, "Ganjil - ID proses", getpid())
        sleep(1)

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

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

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

sekuensial_akhir = time()

# Process
print("\nmultiprocessing.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()

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute  ^C Location  M-U Undo
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify  ^/_ Go To Line M-E Redo
```

```
amanda@amanda-VirtualBox: ~/Sistem Operasi/Tugas 8
File Edit View Search Terminal Help
GNU nano 6.2          Tugas 8.py *
# Sekuensial
print("\nSekuensial")
sekuensial_awal = time()

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

sekuensial_akhir = time()

# Process
print("\nmultiprocessing.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()

# Pool
print("\nmultiprocessing.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   ^C Location  M-U Undo
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^_ Go To Line M-E Redo
```

Pada script ini berisi program yang berfungsi untuk mencetak bilangan bulat positif beserta tipe yaitu ganjil atau genap berdasarkan batasan bilangan yang telah ditentukan oleh user dengan pemrosesan sekuensial, multiprocessing.process dan multiprocessing.pool.

Script:

- Membuat Build-in Libraries

```
from os import getpid, system
from time import time, sleep
from multiprocessing import cpu count, Pool, Process
```

Dilakukan import built-in libraries yang akan digunakan yaitu “getpid” yaitu sebuah function yang mengembalikan (return) ID proses yang sedang berjalan, “time” yaitu sebuah function yang berfungsi untuk mengambil waktu (detik), “sleep” yaitu

sebuah function yang berfungsi untuk menangguhkan eksekusi perintah dalam jumlah waktu (detik) yang diberikan, “process” yaitu sebuah class pada library multiprocessing yang digunakan untuk melakukan pemrosesan paralel dengan menggunakan proses secara beruntun pada komputer, “pool” yaitu sebuah class pada library multiprocessing yang digunakan untuk melakukan pemrosesan paralel dengan menggunakan proses sebanyak jumlah CPU yang terdapat pada komputer, “cpu_count” digunakan untuk melihat jumlah CPU.

- Inisialisasi Function

```
def cetak(i):  
    bilangan = i % 2  
    if bilangan == 0:  
        print(i, "Genap - ID proses", getpid())  
    else:  
        print(i, "Ganjil - ID proses", getpid())  
    sleep(1)
```

Dibuat sebuah function cetak dengan parameter i (cetak(i)) yaitu terdapat perintah `bilangan = i % 2` dengan maksud dilakukan perhitungan modulus 2 terhadap nilai variabel i untuk memeriksa bilangan merupakan bilangan genap atau ganjil. Hasil perhitungan disimpan oleh variabel. Perintah `bil. if bil == 0` berarti jika hasil perhitungan modulus adalah 0, maka bilangan (i) merupakan bilangan genap lalu akan dicetak nilai variabel i, kalimat “Genap – ID Proses”, dan ID proses yang didapatkan melalui penggunaan function `getpid()`. Sedangkan yang disini merupakan `else` berarti jika hasil perhitungan modulus tidak sama dengan 0, maka bilangan (i) merupakan bilangan ganjil lalu akan dicetak nilai variabel i, kalimat “Ganjil – ID Proses”, dan ID proses yang didapatkan melalui penggunaan function `getpid()`. Perintah `sleep(1)` disini digunakan function `sleep(1)` untuk memberikan jeda selama 1 detik sebelum dilanjutkan untuk eksekusi perintah selanjutnya.

- Penginputan Bilangan

```
x = int(input("Input : "))
```

Dibuat sebuah field input dengan tujuan menginputkan bilangan yang digunakan sebagai batasan pada program, banyaknya angka yang akan muncul dan batasannya yaitu tipe data integer untuk user dengan teks “Input : ” yang akan disimpan dalam variabel x.

- Sekuensial

```
print("\nSekuensial")
```

```
sekuensial_awal = time()

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

sekuensial_akhir = time()
```

Dilakukan cetak kalimat “\nSekuensial” untuk menandakan bahwa program yang berjalan di bawah kalimat ini merupakan sekuensial dan perintah \n digunakan untuk mencetak baris baru. sekuensial_awal = time() diartikan sebagai penggunaan function time() untuk mengambil waktu (detik) yang disimpan dalam variabel sekuensial_awal. Selanjutnya dilakukan For loop untuk i dalam range(1, x+1) yang di dalamnya dipanggil dan digunakan function cetak dengan argument variabel i. function range() disini dapat diartikan sebagai pengembalian urutan angka yang secara default dimulai dari 0, bertambah dengan interval 1, dan akan berhenti sebelum dari angka yang ditetapkan. 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. Selanjutnya yaitu sekuensial_akhir = time() yang digunakan untuk mengambil waktu (detik) yang disimpan dalam variabel sekuensial_akhir.

- Process

```
print("\nmultiprocessing.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()
```

Dilakukan cetak kalimat “\nmultiprocessing.Process” untuk menandakan bahwa program yang berjalan di bawah kalimat ini merupakan multiprocessing dengan kelas Process dan perintah \n digunakan untuk mencetak baris baru. Pada process_awal = time() yang digunakan untuk mengambil waktu (detik) yang disimpan dalam variabel process_awal. Lalu dilakukan inisialisasi sebuah list kosong ([]) yang dengan nama variabel kumpulan_proses. Setelah itu dilakukan perulangan yang pertama yaitu For loop (1) 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. Sedangkan pada For loop (2) digunakan method join() terhadap variabel p (p.join()) untuk menggabungkan kumpulan proses yang telah ditampung agar tidak merambat pada proses selanjutnya. Selanjutnya yaitu process_akhir = time() ini digunakan function time() untuk mengambil waktu (detik) yang disimpan dalam variabel process_akhir.

- Pool

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

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

pool_akhir = time()
```

Dilakukan cetak kalimat “\nmultiprocessing.Pool” untuk menandakan bahwa program yang berjalan di bawah kalimat ini merupakan multiprocessing dengan kelas Pool. \n digunakan untuk mencetak baris baru. Selanjutnya yaitu pool_awal = time() Digunakan function time() untuk mengambil waktu (detik) yang disimpan dalam variabel pool_awal. Setelah itu, diinisialisasikan kelas Pool() yang disimpan dalam variabel pool dan digunakan method map() terhadap variabel pool (pool.map()) untuk memetakan pemanggilan function cetak ke dalam 2 CPU sebanyak yang bilangan yang berada pada range(1, x + 1) dan 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. Selanjutnya yaitu pool_akhir = time() yang digunakan function time() untuk mengambil waktu (detik) yang disimpan dalam variabel pool_akhir.

- 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")
```

Dilakukan cetak nilai durasi waktu untuk setiap pemrosesan yang didapatkan melalui perhitungan antara waktu awal setiap proses dikurangi dengan waktu akhir setiap proses.

3. Menjalankan Script

```
amanda@amanda-VirtualBox: ~/Sistem Operasi/Tugas 8
File Edit View Search Terminal Help
amanda@amanda-VirtualBox:~/Sistem Operasi/Tugas 8$ python3 Tugas_8.py
Input : 3

Sekuensial
1 Ganjil - ID proses 2445
2 Genap - ID proses 2445
3 Ganjil - ID proses 2445

multiprocessing.Process
1 Ganjil - ID proses 2446
2 Genap - ID proses 2447
3 Ganjil - ID proses 2448

multiprocessing.Pool
1 Ganjil - ID proses 2449
2 Genap - ID proses 2450
3 Ganjil - ID proses 2449

Waktu eksekusi sekuensial : 3.0407705307006836 detik
Waktu eksekusi multiprocessing.Process : 1.0088825225830078 detik
Waktu eksekusi multiprocessing.Pool : 2.0164802074432373 detik
amanda@amanda-VirtualBox:~/Sistem Operasi/Tugas 8$
```

- Diinputkan bilangan 3 sebagai batasan oleh user. Pada sekuensial dapat dilihat bahwa untuk setiap bilangan yang dicetak melalui function cetak memiliki ID proses yang sama (2445). Hal tersebut menandakan bahwa untuk semua pemanggilan function cetak ditangani oleh satu proses yang sama. Pada multiprocessing.Process dapat dilihat bahwa untuk setiap bilangan yang dicetak melalui function cetak memiliki ID proses yang berbeda dan urut (2446 – 2448). Hal tersebut mendandakan bahwa untuk setiap pemanggilan function cetak ditangani oleh satu proses saja. Pada multiprocessing.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 2 ID proses (2449 dan 2450) yang berulang dengan tidak urut. Hal tersebut terjadi karena pada komputer yang digunakan untuk menjalankan script tersebut hanya memiliki 2 CPU. 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 yang paling lama.
- Untuk mengetahui jumlah CPU yang terdapat pada komputer dapat digunakan function `cpu_count()` yang terdapat dalam library multiprocessing sebagai berikut:

```
amanda@amanda-VirtualBox:~/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()
2
>>> 
```