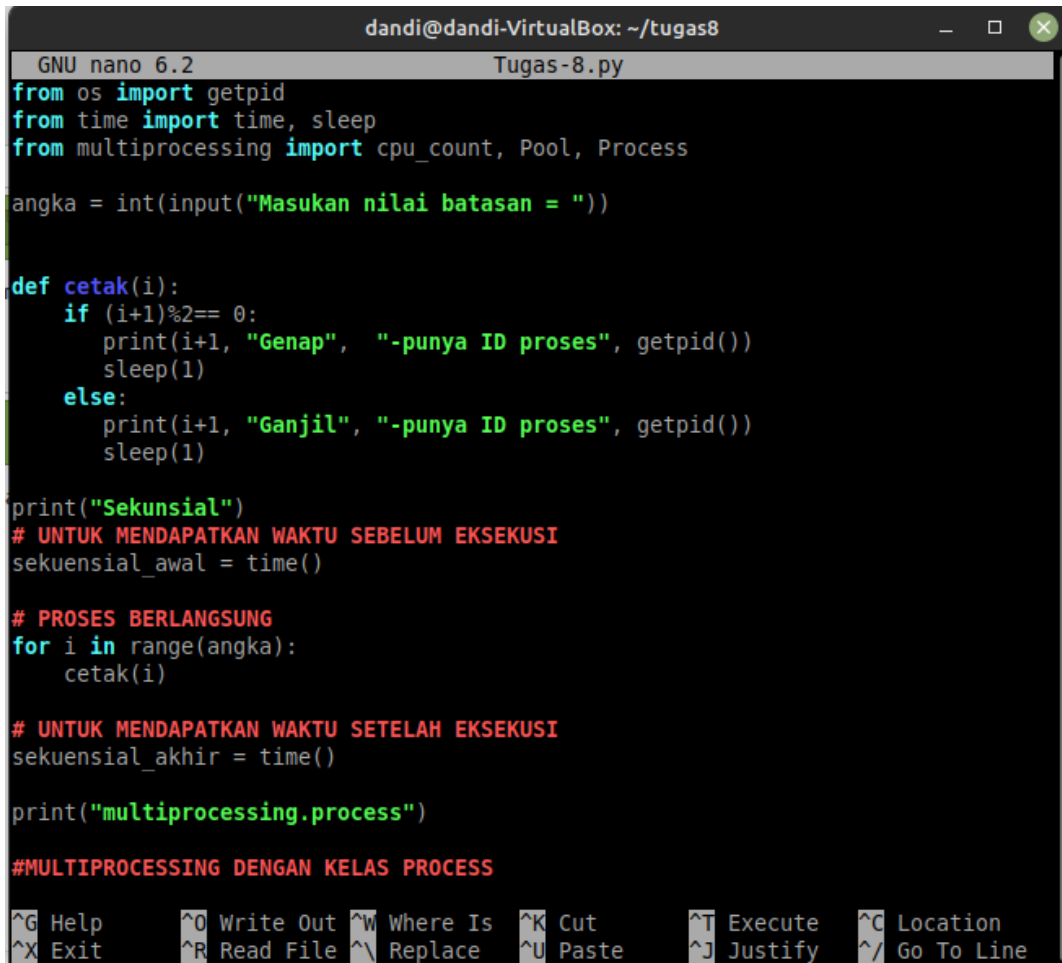


### A. Script dari soal Latihan multiprocessing



```
dandi@dandi-VirtualBox: ~/tugas8
GNU nano 6.2          Tugas-8.py
from os import getpid
from time import time, sleep
from multiprocessing import cpu_count, Pool, Process

angka = int(input("Masukan nilai batasan = "))

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

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

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

# UNTUK MENDAPATKAN WAKTU SETELAH EKSEKUSI
sekuensial_akhir = time()

print("multiprocessing.process")

#MULTIPROCESSING DENGAN KELAS PROCESS

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

```
dandi@dandi-VirtualBox: ~/tugas8
GNU nano 6.2      Tugas-8.py

#MULTIPROCESSING DENGAN KELAS PROCESS
process_awal=time()
for i in range(angka):
    p=Process(target=cetak, args=(i, ))
    p.start()
    p.join()
process_akhir=time()

print("multiprocessing.pool")

# UNTUK Mendapatkan Waktu Sebelum Eksekusi
pool_awal = time()

# PROSES BERLANGSUNG
pool = Pool()
pool.map(cetak, range(angka))
pool.close()

# UNTUK Mendapatkan Waktu Sebelum Eksekusi
pool_akhir = time()

print("Sekuensial :", sekuensial_akhir - sekuensial_awal, "detik")
print("Kelas Process :", process_akhir - process_awal, "detik")
print("Kelas Pool :", pool_akhir - pool_awal, "detik")

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

## B. Pengertian dari script

1. Kita import dulu library yang diperlukan

```
GNU nano 6.2      Tugas-8.py
from os import getpid
from time import time, sleep
from multiprocessing import cpu_count, Pool, Process
```

2. Fungsi ini digunakan untuk mencetak angka dari variabel i beserta ID proses sejumlah parameter yang diberikan. Kita panggil fungsi sleep untuk memberi jeda waktu(detik) sebanyak parameter yang diberikan.

```
angka = int(input("Masukan nilai batasan = "))

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

### 3. Pemrosesan Sekuensial

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

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

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

### 4. Multiprocessing dengan kelas Process

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

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

# UNTUK MENDAPATKAN WAKTU SETELAH EKSEKUSI
sekuensial_akhir = time()

print("multiprocessing.process")
```

Dapat diperhatikan dengan seksama bahwa ID proses tiap memanggil fungsi cetak adalah berbeda-beda. Ini menandakan bahwa tiap pemanggilan fungsi cetak ditangani oleh satu proses saja. Kemudian untuk pemanggilan selanjutnya 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.

### 5. Multiprocess dengan kelas Pool

```
print("multiprocessing.pool")

# UNTUK MENDAPATKAN WAKTU SEBELUM EKSEKUSI
pool_awal = time()

# PROSES BERLANGSUNG
pool = Pool()
pool.map(cetak, range(angka))
pool.close()

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

Jumlah ID proses terbatas pada empat saja karena jumlah CPU pada komputer saya hanyalah 6. Jangan risaukan urutan angka yang dicetak jika tidak berurutan, kan emang ini pemrosesan paralel. Fungsi `map()` itu memetakan pemanggilan fungsi cetak ke dalam 6 CPU sebanyak 10 kali.

### 6. Bandingkan Waktu Eksekusi

```
print("Sekuensial :", sekuensial_akhir - sekuensial_awal, "detik")
print("Kelas Process :", process_akhir - process_awal, "detik")
print("Kelas Pool :", pool_akhir - pool_awal, "detik")
```

Sudah sewajarnya proses sekuensial lebih lambat dibanding multiprocessing namun bukan berarti kita harus melakukan multiprocessing terus menerus, gunakan metode sesuai kebutuhan. Nah apabila barisan kode di atas dikumpulkan jadi satu maka jadinya akan seperti ini.

Output

```
dandi@dandi-VirtualBox:~/tugas8$ nano Tugas-8.py
dandi@dandi-VirtualBox:~/tugas8$ python3 Tugas-8.py
Masukan nilai batasan = 3
Sekunsial
1 Ganjil -punya ID proses 3064
2 Genap -punya ID proses 3064
3 Ganjil -punya ID proses 3064
multiprocessing.process
1 Ganjil -punya ID proses 3075
2 Genap -punya ID proses 3076
3 Ganjil -punya ID proses 3077
multiprocessing.pool
1 Ganjil -punya ID proses 3078
2 Genap -punya ID proses 3078
3 Ganjil -punya ID proses 3078
Sekuensial : 3.0033791065216064 detik
Kelas Process : 3.1210575103759766 detik
Kelas Pool : 3.281996726989746 detik
```