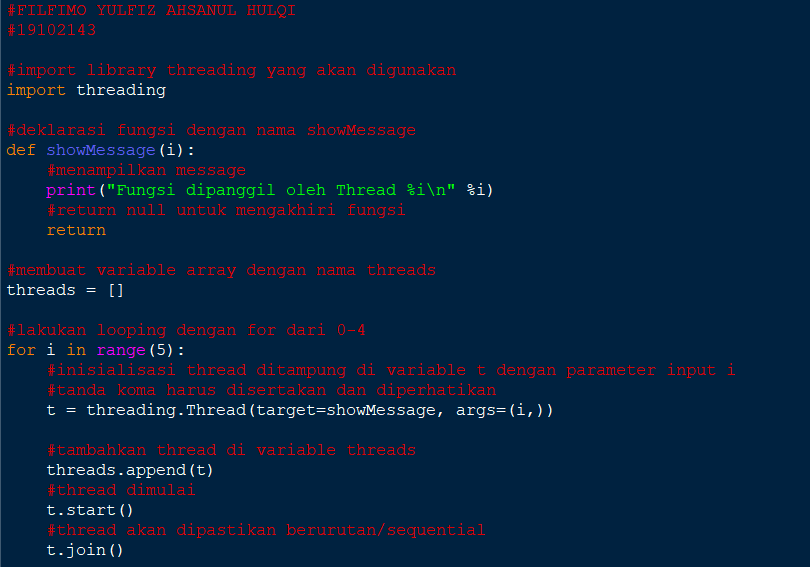
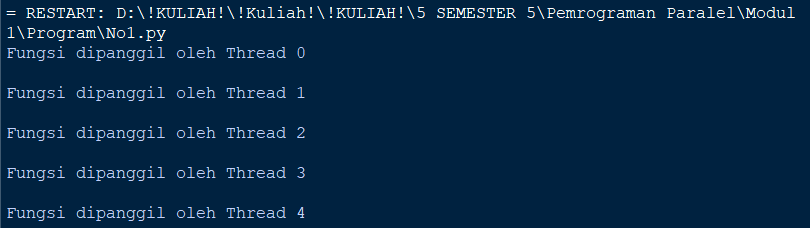
1. Program Thread Sequential



Gambar Program Thread Sequential

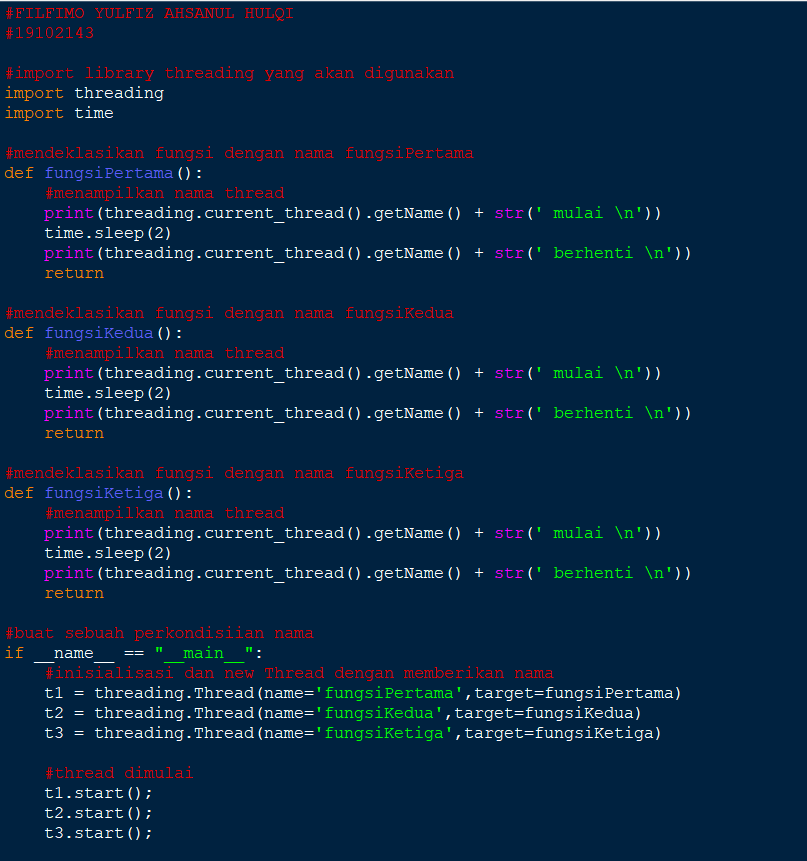


Gambar Output Program Sequential

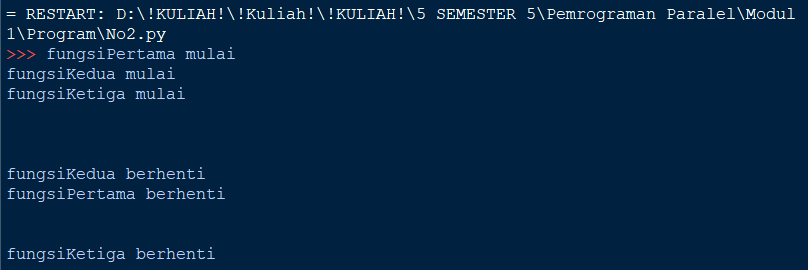
Penjelasan Program

Program diatas adalah contoh program *thread* secara *sequential* atau berurutan, pertama mendeklarasikan terlebih dahulu *library threading* yang telah disediakan oleh Python, selanjutkan buat sebuah fungsi untuk menampilkan pesan atau keterangan dari *thread*, buat variabel *array* dengan nama *threads*, lakukan *looping* 0 – 4 (*array* dimulai dari 0), didalam *looping* lakukan inisialisasi thread yang ditampung di variabel *t* dengan parameter input i, tambahkan *thread* di variabel *threads* dengan *append*, jalankan dengan *start* dan terakhir lakukan *join*, maka *thread* akan dipastikan berurutan.

1. Program Thread Time Sleep



Gambar Program Thread Time Sleep



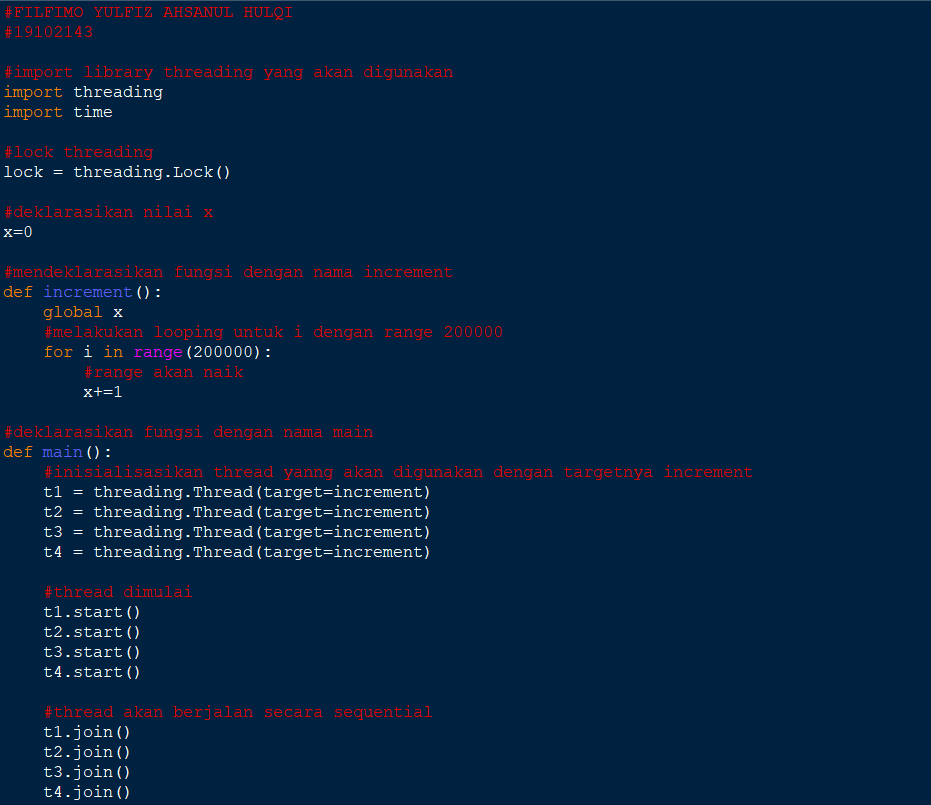
Gambar Output Program Thread Time Sleep

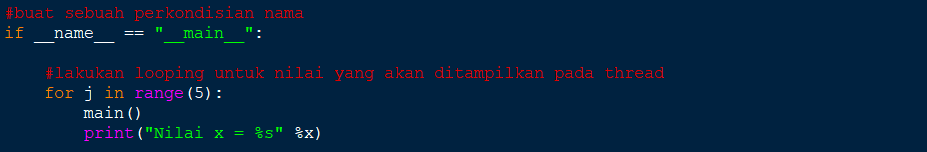
Penjelasan Program

Program diatas adalah contoh program *thread* menggunakan *method time sleep*, pertama mendeklarasikan terlebih dahulu *library threading* dan *time* yang telah disediakan oleh Python, selanjutkan buat sebuah fungsi untuk menampilkan pesan atau keterangan dari *thread*, disini ada tiga buah fungsi yang dibuat, masing-masing fungsi tersebut didalamnya terdapat *method* *sleep*, *method* ini dapat digunakan untuk melakukan *pause* terhadap eksekusi yang terjadi pada *thread* yang sedang digunakan, hasil *output*nya seperti pada gambar diatas, *thread* dimulai secara berurutan tetapi berakhir tidak secara berurutan.

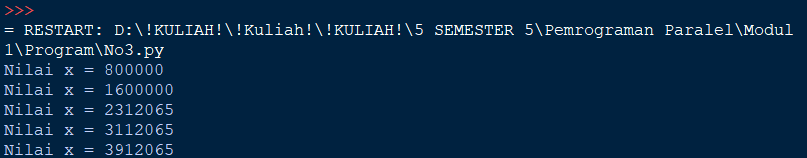
1. Program Thread Race Condition

*Race Condition* adalah situasi dimana beberapa proses mengakses dan memproses data secara bersamaan, sehingga akan saling berebutan dan nilai akhir dari data tersebut akan bergantung pada proses yang terakhir selesai, untuk mencegah hal ini, dapat dilakukan beberapa hal, salah satunya adalah menggunakan *Lock, Acquire* dan *Release*.





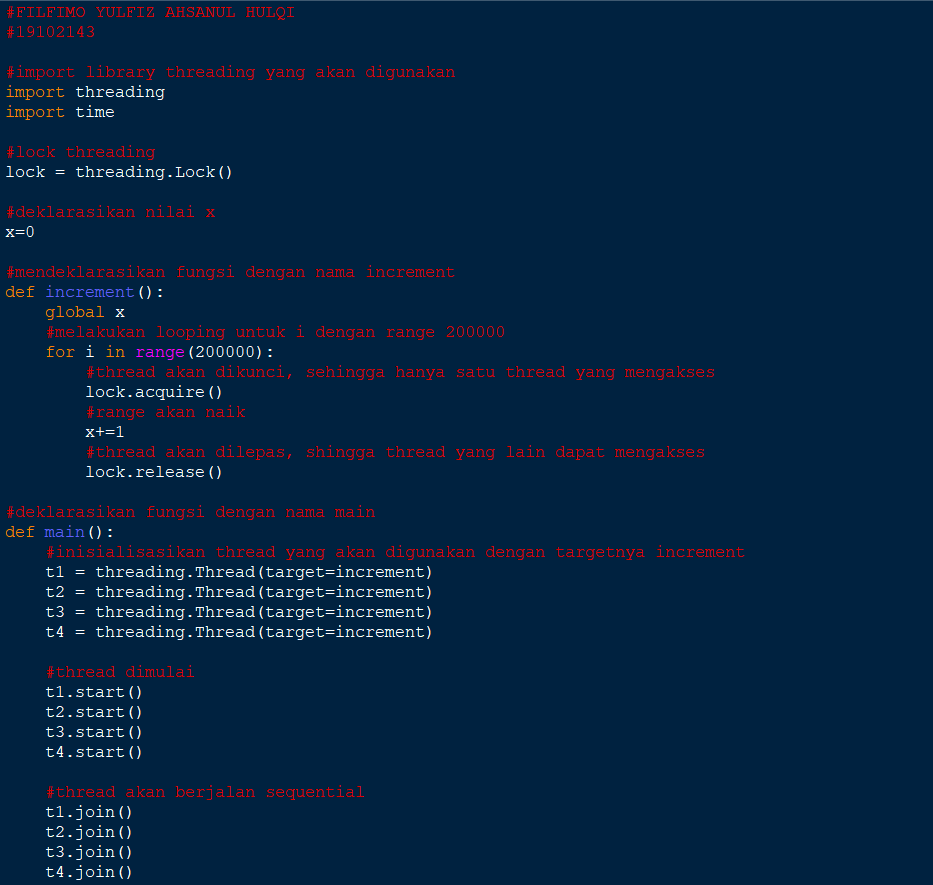
Gambar Program Thread Race Condition tanpa Lock

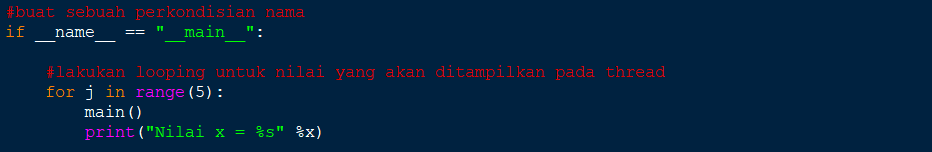


Gambar Output Program Thread Race Condition tanpa Lock

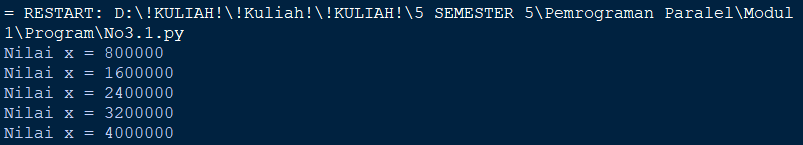
Penjelasan Program :

Program diatas adalah contoh program *thread* *race condition* tanpa *Lock*, pertama mendeklarasikan terlebih dahulu *library threading* dan *time* yang telah disediakan oleh Python, selanjutnya deklarasikan *lock threading* (tapi disini belum digunakan, siapkan untuk program selanjutnya), deklarasikan nilai x, deklarasikan fungsi dengan *increment* untuk melakukan *looping* nilai atau proses dari *thread*, kemudian deklarasikan fungsi dengan nama *main*, didalamnya melakukan inisialisasi *thread* yang akan digunakan dengan target *increment*, kemudian mulai *thread* yang akan berjalan secara *sequential*, buat perkondisian nama, terakhir lakukan *looping* untuk nilai yang akan ditampilkan pada *thread*.





Gambar Program Thread Race Condition menggunakan Lock



Gambar Output Program Thread Condition menggunakan Lock

Penjelasan Program :

Program diatas adalah contoh program *thread* *race condition* menggunakan *Lock*, untuk deklarasi program dan kode program beberapa masih sama seperti sebelumnya yang berbeda adalah pada bagian fungsi *increment* didalamnya terdapat *lock acquire* dan *lock release*, yang akan digunakan agar *thread* tidak saling berebutan dalam melakukan proses, jadi salah satu *thread* akan masuk dan mengunci, jika sudah selesai maka *thread* akan lepas dan diganti dengan proses lainnya, sehingga akan dihasilkan *output* yang lebih baik (urut) daripada *output* sebelumnya.