**LAPORAN TUGAS BESAR SISTEM OPERASI**

“Implementasi dan Analisis Proses Penjadwalan Menggunakan Algoritma CPU Scheduling pada Sistem Operasi”



**Dosen Pengampu : Dr. Marischa Elveny S.TI., M.Kom.**

**Disusun Oleh:**

**Kom A - Kelompok X :**

Rahmat Maulana Miftah 231402022

Mayadi Alamsyah Putra Silalahi 231402046

Refael Juari Siagian 231402055

Sevilla Claudia Depari 231402058

**Jonathan C. Amadeo Sembiring 231402111**

**Sistem Operasi**

**PROGRAM STUDI TEKNOLOGI INFORMASI**

**FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI**

**UNIVERSITAS SUMATERA UTARA**

**2025**

**ABSTRAK**

Sistem operasi memerlukan mekanisme yang efisien untuk mengelola alokasi CPU kepada berbagai proses yang berjalan secara bersamaan. Mekanisme ini dikenal sebagai penjadwalan CPU. Laporan ini membahas implementasi dan analisis perbandingan dari empat algoritma penjadwalan CPU fundamental: First-Come, First-Served (FCFS), Shortest Job First (SJF) non-preemptive, Priority Scheduling non-preemptive, dan Round Robin (RR). Program simulasi dikembangkan menggunakan bahasa Python untuk mengukur dan membandingkan kinerja setiap algoritma berdasarkan metrik kunci, termasuk waktu tunggu rata-rata, waktu respons rata-rata, dan throughput. Hasil analisis menunjukkan bahwa tidak ada algoritma tunggal yang unggul dalam semua skenario; pilihan algoritma yang optimal sangat bergantung pada tujuan spesifik dari sistem, menyoroti trade-off antara efisiensi, keadilan, dan responsivitas.

**BAB I: PENDAHULUAN**

**1.1 Latar Belakang**

Dalam lingkungan komputasi modern, sebuah sistem operasi (SO) menjalankan banyak proses secara konkuren. CPU, sebagai sumber daya komputasi utama, harus dibagikan secara efisien di antara semua proses ini. Proses penentuan proses mana yang akan mendapatkan alokasi CPU dan untuk berapa lama disebut penjadwalan CPU. Keputusan penjadwalan yang buruk dapat menyebabkan kinerja sistem yang lambat, waktu respons yang lama, dan utilisasi sumber daya yang tidak efisien. Oleh karena itu, studi mengenai algoritma penjadwalan CPU menjadi fundamental dalam ilmu komputer untuk memahami bagaimana SO dapat mengelola tugas-tugasnya secara optimal.

**1.2 Tujuan Proyek**

Tujuan dari proyek ini adalah sebagai berikut:

1. Mengimplementasikan empat algoritma penjadwalan CPU dasar (FCFS, SJF, Priority, dan Round Robin) menggunakan bahasa pemrograman Python.
2. Membangun sebuah program simulasi untuk menjalankan algoritma-algoritma tersebut pada satu set proses yang sama.
3. Mengukur kinerja setiap algoritma menggunakan metrik standar: waktu tunggu rata-rata (*average waiting time*), waktu putar-balik rata-rata (*average turnaround time*), waktu respons rata-rata (*average response time*), dan *throughput*.
4. Menganalisis dan membandingkan hasil kinerja untuk memahami kekuatan, kelemahan, dan karakteristik dari setiap algoritma dalam konteks praktis.

**1.3 Ruang Lingkup**

Proyek ini mencakup pengembangan program dari awal hingga akhir, termasuk pendefinisian struktur data proses, implementasi logika untuk setiap algoritma, dan pembuatan skrip untuk simulasi dan visualisasi data. Algoritma SJF dan Priority yang diimplementasikan adalah versi non-preemptive.

**BAB II: DASAR TEORI**

**2.1 Penjadwalan CPU**

Penjadwalan CPU adalah proses pemilihan salah satu proses dari *ready queue* untuk dialokasikan ke CPU. Tujuan utama dari penjadwalan adalah untuk memaksimalkan *throughput* dan utilisasi CPU, serta meminimalkan waktu tunggu, waktu respons, dan waktu putar-balik.

**2.2 Algoritma First-Come, First-Served (FCFS)**

Algoritma ini adalah yang paling sederhana. Proses yang meminta CPU pertama kali akan mendapatkan alokasi CPU pertama kali. Implementasinya menggunakan antrian First-In, First-Out (FIFO). FCFS bersifat non-preemptive, artinya sekali sebuah proses mendapatkan CPU, ia akan menjalankannya hingga selesai.

**2.3 Algoritma Shortest Job First (SJF)**

Algoritma ini mengasosiasikan setiap proses dengan panjang *burst time* berikutnya. Ketika CPU tersedia, ia akan dialokasikan ke proses yang memiliki *burst time* terkecil. Versi non-preemptive dari SJF akan membiarkan proses berjalan hingga selesai sebelum memilih proses terpendek berikutnya dari *ready queue*.

**2.4 Algoritma Priority Scheduling**

Sebuah prioritas diasosiasikan dengan setiap proses, dan CPU dialokasikan ke proses dengan prioritas tertinggi. Dalam implementasi ini, nilai integer yang lebih kecil merepresentasikan prioritas yang lebih tinggi. Sama seperti SJF, versi non-preemptive akan menjalankan proses terpilih hingga selesai.

**2.5 Algoritma Round Robin (RR)**

Algoritma ini dirancang khusus untuk sistem *time-sharing*. Sebuah unit waktu kecil, yang disebut *quantum* atau *time slice*, didefinisikan. *Ready queue* diperlakukan sebagai antrian sirkular. Penjadwal akan mengalokasikan CPU ke setiap proses untuk interval waktu paling lama satu quantum. Jika proses belum selesai, ia akan dipindahkan ke belakang *ready queue*.

**BAB III: IMPLEMENTASI DAN PENGUJIAN**

**3.1 Struktur Program**

Program ini terdiri dari dua file utama: **cpu\_scheduling.py** dan **simulation.py.**

* **cpu\_scheduling.py:** Berisi logika inti.
  + Process: Sebuah dataclass Python yang merepresentasikan sebuah proses dengan atribut seperti pid, arrival\_time, burst\_time, dan priority.
  + CPUScheduler: Sebuah kelas yang berisi implementasi dari keempat algoritma penjadwalan dan metode untuk menghitung metrik kinerja.
* **simulation.py:** Bertindak sebagai *driver* atau eksekutor. File ini berfungsi untuk:
  + Menghasilkan satu set proses uji secara acak.
  + Membuat instance dari CPUScheduler.
  + Menjalankan semua algoritma pada set proses tersebut.
  + Mencetak hasil dalam bentuk tabel dan memvisualisasikannya dalam bentuk grafik batang.

**3.2 Skenario Pengujian**

Untuk pengujian dan analisis, sebuah simulasi dijalankan dengan 5 proses yang dibuat secara acak. Quantum waktu untuk algoritma Round Robin ditetapkan sebesar 4 unit waktu.

Tabel 3.1: Proses Uji yang Digunakan dalam Simulasi

|  |  |  |  |
| --- | --- | --- | --- |
| PID | Waktu Kedatangan | Waktu Eksekusi (Burst Time) | Prioritas |
| 1 | 7 | 14 | 9 |
| 2 | 1 | 15 | 4 |
| 3 | 9 | 1 | 1 |
| 4 | 6 | 17 | 7 |
| 5 | 0 | 17 | 4 |

**3.3 Hasil Pengujian**

Setelah menjalankan skrip simulasi pada data di atas, metrik kinerja untuk setiap algoritma dihitung dan disajikan di bawah ini.

Tabel 3.2: Hasil Kinerja Algoritma FCFS

|  |  |
| --- | --- |
| Metrik | Nilai |
| Rata-rata Waktu Tunggu | 25.4 |
| Rata-rata Waktu Putar-Balik | 40.2 |
| Rata-rata Waktu Respons | 25.4 |
| Throughput | 0.08 |

Tabel 3.3: Hasil Kinerja Algoritma SJF

|  |  |
| --- | --- |
| Metrik | Nilai |
| Rata-rata Waktu Tunggu | 18 |
| Rata-rata Waktu Putar-Balik | 32.8 |
| Rata-rata Waktu Respons | 18 |
| Throughput | 0.08 |

Tabel 3.4: Hasil Kinerja Algoritma Priority

|  |  |
| --- | --- |
| Metrik | Nilai |
| Rata-rata Waktu Tunggu | 21.2 |
| Rata-rata Waktu Putar-Balik | 36 |
| Rata-rata Waktu Respons | 21.2 |
| Throughput | 0.08 |

Tabel 3.5: Hasil Kinerja Algoritma Round Robin (Quantum=4)

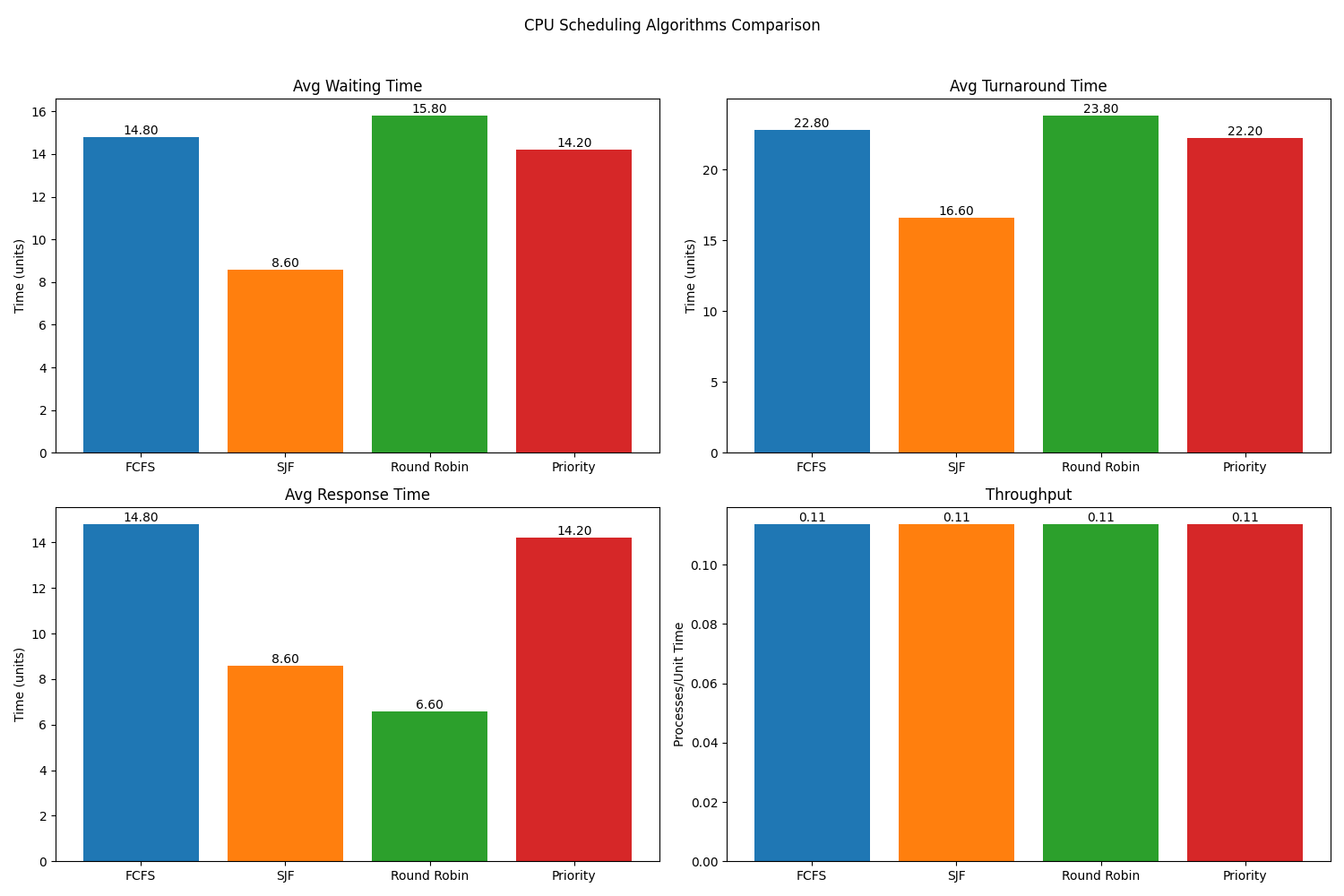
|  |  |
| --- | --- |
| Metrik | Nilai |
| Rata-rata Waktu Tunggu | 30.6 |
| Rata-rata Waktu Putar-Balik | 45.4 |
| Rata-rata Waktu Respons | 8.8 |
| Throughput | 0.08 |

**BAB IV: ANALISIS DAN DISKUSI**

**4.1 Analisis Perbandingan Kinerja**

Berdasarkan hasil yang disajikan pada Bab III, dapat dilakukan analisis perbandingan sebagai berikut:

* **Waktu Tunggu Rata-rata**: Algoritma **SJF** menunjukkan kinerja terbaik dengan waktu tunggu rata-rata terendah (18.00). Ini sesuai dengan sifatnya yang mendahulukan pekerjaan terpendek, sehingga secara efektif mengurangi waktu tunggu akumulatif. Sebaliknya, **Round Robin** memiliki waktu tunggu terburuk (30.60) karena proses yang lebih panjang harus berulang kali masuk ke antrian.
* **Waktu Respons Rata-rata**: Algoritma **Round Robin** secara signifikan mengungguli semua algoritma lain dalam metrik ini dengan waktu respons rata-rata hanya 8.80. Hal ini karena RR dirancang untuk memberikan setiap proses kesempatan berjalan dalam waktu cepat, sehingga sangat cocok untuk sistem interaktif. Algoritma non-preemptive lainnya memiliki waktu respons yang sama dengan waktu tunggunya, yang jauh lebih tinggi.
* **Throughput**: Dalam skenario pengujian ini, *throughput* untuk semua algoritma adalah sama (0.08). Hal ini wajar terjadi pada simulasi dengan jumlah proses yang tetap, di mana total waktu yang dibutuhkan untuk menyelesaikan semua pekerjaan tidak berbeda secara drastis.



Gambar 4.1: Visualisasi perbandingan metrik kinerja untuk keempat algoritma penjadwalan.

**4.2 Kelebihan dan Kekurangan Algoritma**

* **FCFS**:
  + *Kelebihan*: Sangat sederhana untuk diimplementasikan dan dipahami.
  + *Kekurangan*: Tidak efisien. Waktu tunggu rata-rata bisa sangat buruk jika proses pendek tiba setelah proses yang sangat panjang (dikenal sebagai *convoy effect*).
* **SJF**:
  + *Kelebihan*: Terbukti optimal dalam menghasilkan waktu tunggu rata-rata minimum.
  + *Kekurangan*: Tidak praktis di dunia nyata karena tidak mungkin untuk mengetahui panjang *burst time* berikutnya secara pasti. Berisiko menyebabkan *starvation* untuk proses yang panjang.
* **Priority Scheduling**:
  + *Kelebihan*: Memungkinkan implementasi kebijakan prioritas, di mana proses-proses penting bisa didahulukan.
  + *Kekurangan*: Dapat menyebabkan *starvation* untuk proses berprioritas rendah.
* **Round Robin**:
  + *Kelebihan*: Waktu respons yang sangat baik, adil karena setiap proses mendapat jatah CPU.
  + *Kekurangan*: Waktu tunggu dan waktu putar-balik rata-rata cenderung lebih buruk daripada SJF. Kinerjanya sensitif terhadap pemilihan ukuran *quantum*.

**BAB V: KESIMPULAN**

**5.1 Ringkasan**

Proyek ini telah berhasil mengimplementasikan dan menganalisis empat algoritma penjadwalan CPU. Melalui simulasi, telah terbukti secara kuantitatif bahwa setiap algoritma memiliki karakteristik kinerja yang unik. SJF unggul dalam efisiensi waktu tunggu, sementara Round Robin unggul dalam responsivitas. Hal ini mengkonfirmasi prinsip fundamental dalam sistem operasi: tidak ada satu algoritma penjadwalan "terbaik" yang universal. Pemilihan algoritma yang tepat adalah sebuah pertukaran (*trade-off*) yang bergantung pada tujuan dan prioritas sistem yang sedang dibangun.

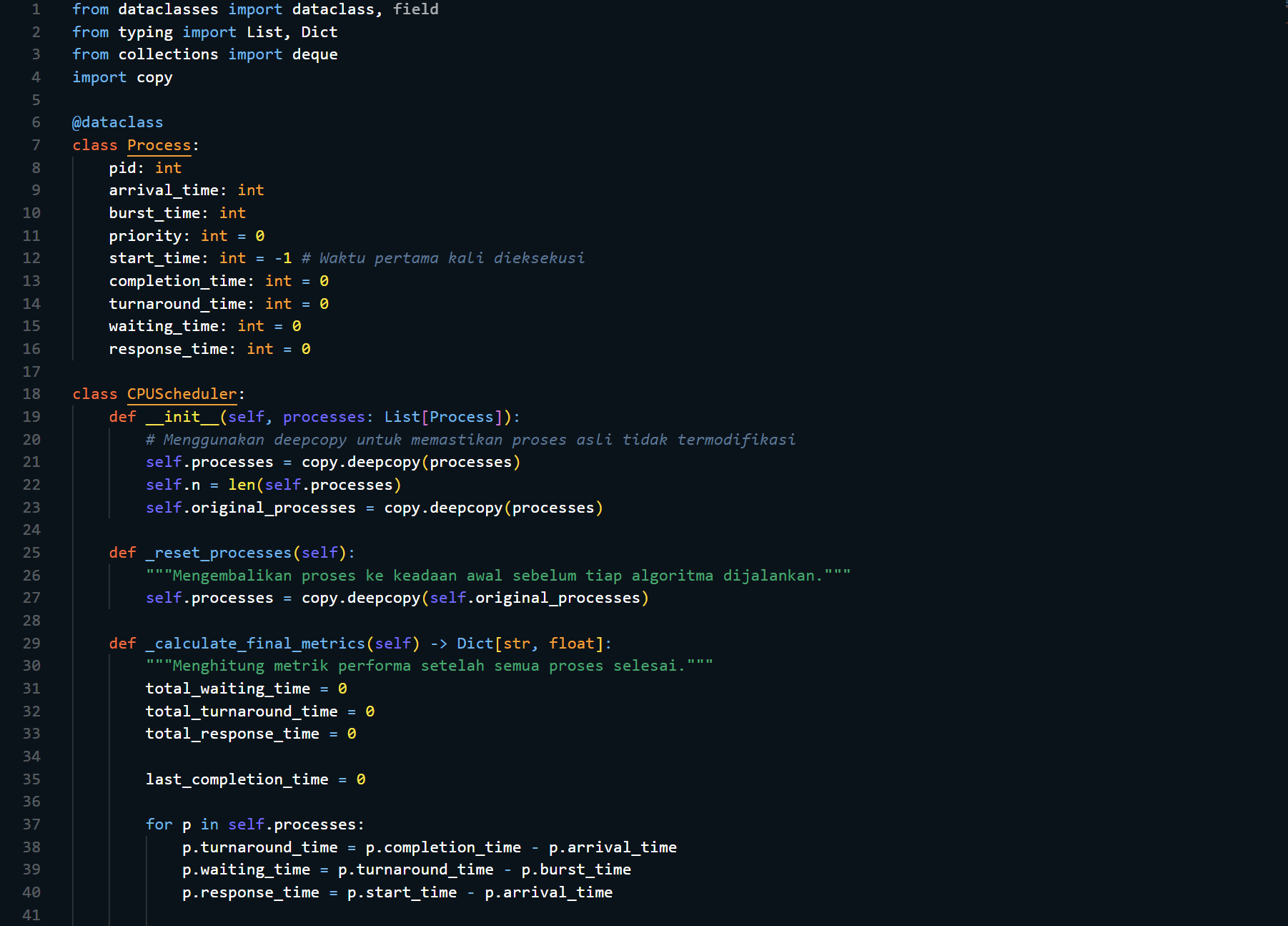
**5.2 Saran**

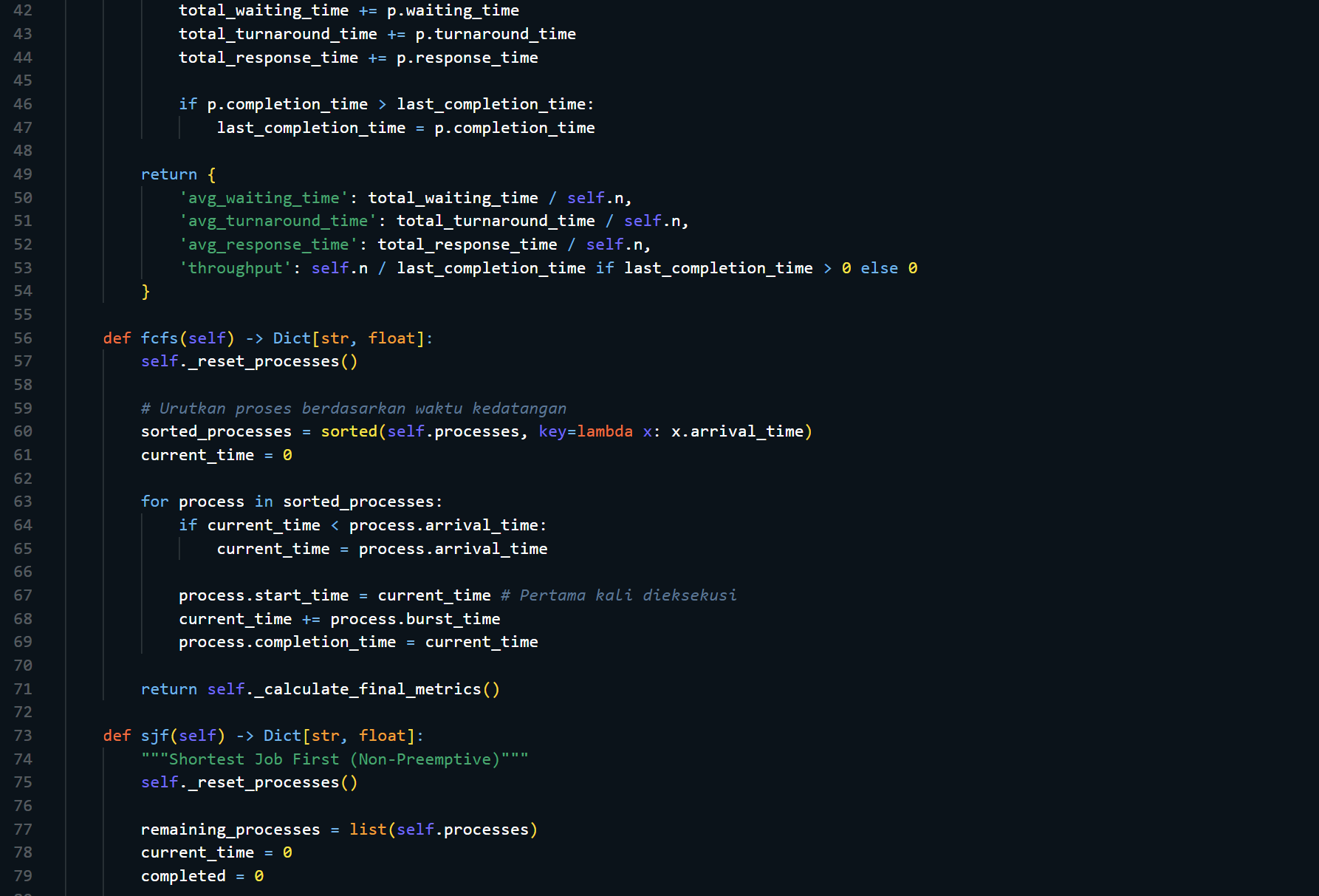
Sebagai pengembangan lebih lanjut, penelitian dapat diperluas untuk mencakup versi preemptive dari algoritma SJF dan Priority. Selain itu, implementasi algoritma yang lebih kompleks seperti *Multilevel Feedback Queue Scheduling* dapat memberikan wawasan yang lebih dalam mengenai teknik penjadwalan yang digunakan pada sistem operasi modern.

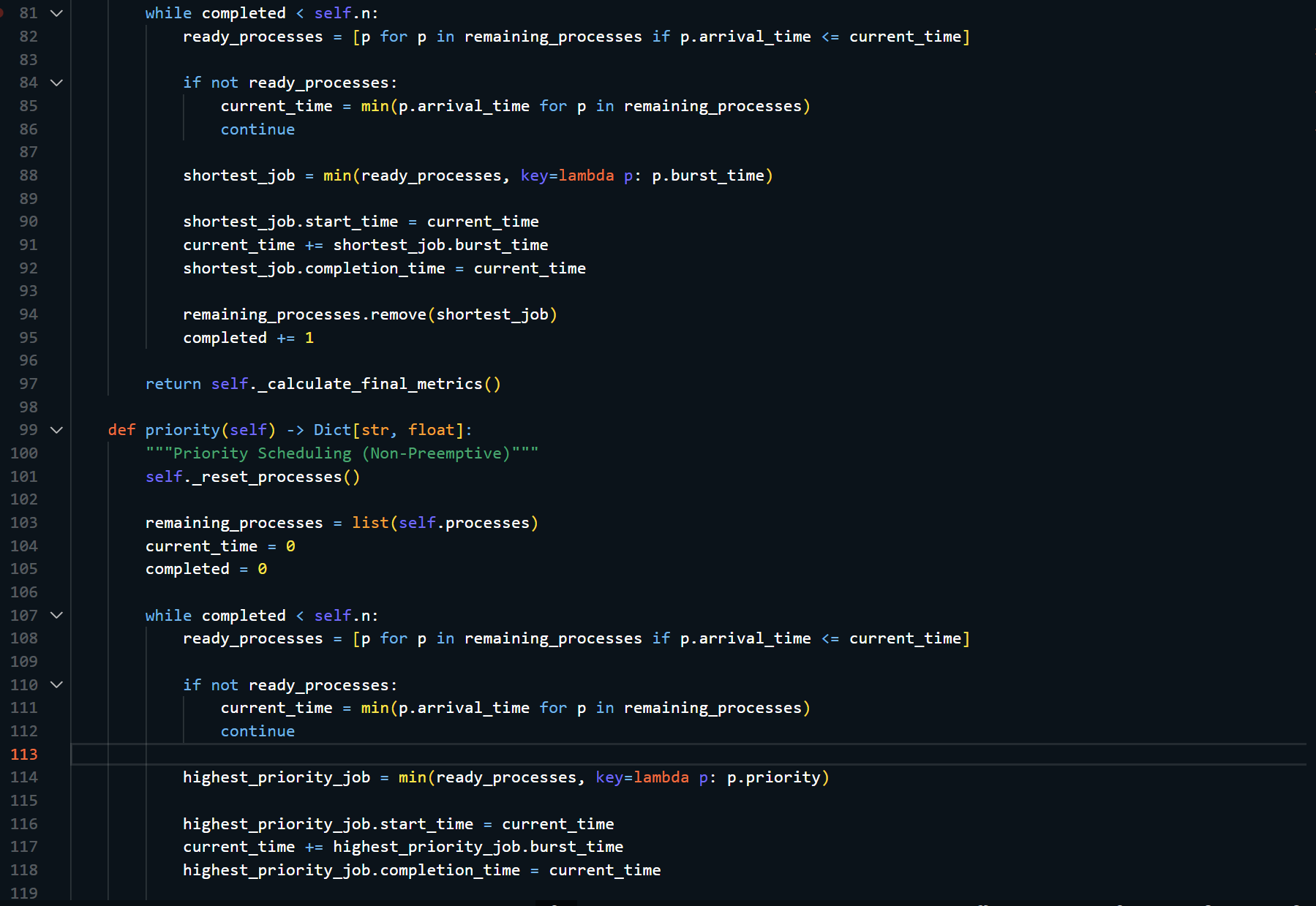
**LAMPIRAN**

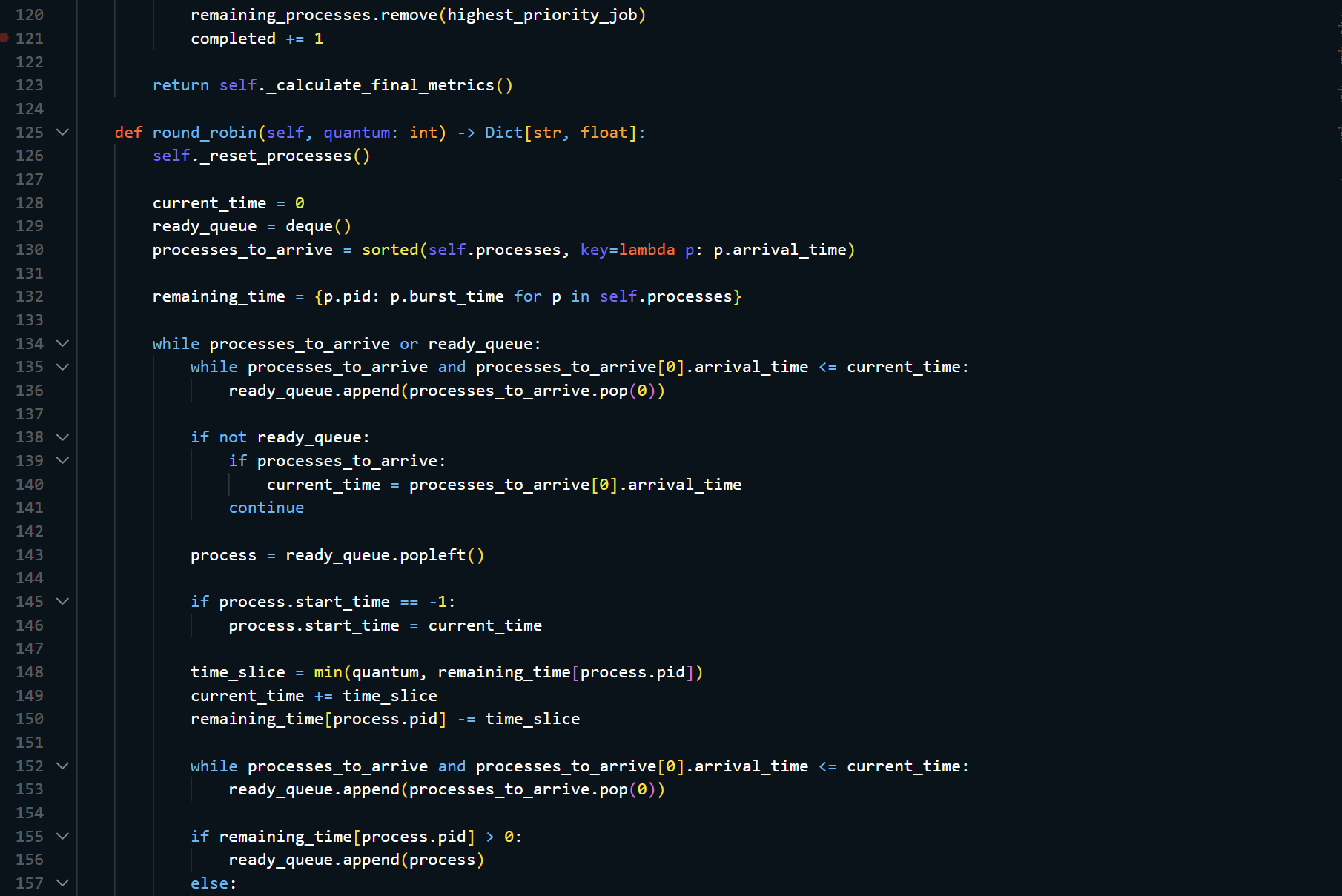
**Kode Sumber**

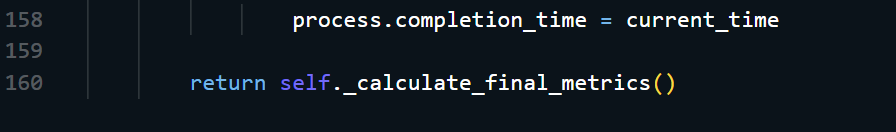
* **cpu\_scheduling.py**











* simulation.py

