Modul Panduan Sistem Operasi
Yang Tidak Dipublikasikan

Penggunaan PyPlot untuk Representasi Data Simulasi Operating Systems

Berkah I. Santoso, ST, MTI

Program Studi Informatika
Fakultas Teknik dan Ilmu Komputer
Universitas Bakrie
Jakarta

2025

HALAMAN PENGESAHAN

1.	Judul PkM	: Modul Panduan Sistem Operasi : Penggunaan PyPlot untuk Representasi Data Simulasi Operating Systems
2.	Ketua Tim Pengabdian	
	a. Nama Lengkap	: Berkah I. Santoso, ST, MTI
	b. NIDN	: 0309068003
	c. Pangkat/Golongan	: Penata Muda Tingkat I / III b
	d. Jabatan	: Asisten Ahli
	e. Telp/Alamat Surel	: 021-5261448 / berkah.santoso@bakrie.ac.id
3.	Anggota Tim Pengabdian	
	a. Dosen	: 1. –
		2. –
	b. Praktisi	: 1. –
		2. –
4.	Peserta	
	a. Mahasiswa	: 1. –
		2. –
	b. Alumni	: 1. –
		2. –
5.	Biaya Kegiatan	
	a. Universitas Bakrie	: Rp 0,-
	b. Sumber lain	: Rp 0,-
6.	Tahun Pelaksanaan	: Tahun 2025

Mengetahui,
Kaprodi Informatika,

Jakarta, 6 Januari 2025

Dr. Iwan Adhicandra, Ph.D, SMIEEE
NIDN : 0018025806

Berkah I. Santoso, S.T., M.T.I.
NIDN : 0309068003

Mengetahui,
Ketua LPkM Universitas Bakrie

Prof. Ardiansyah, Ph.D
NIDN : 0318107501

Abstrak

Representasi simulasi data sebagai penunjang suatu materi perkuliahan, dalam hal ini *Operating Systems* merupakan salah satu cara untuk pemenuhan syarat pembelajaran dalam rangka percepatan penerimaan topik materi yang dirasakan sangat efektif bagi mahasiswa. Mahasiswa sebagai peserta pembelajaran sudah dapat dipastikan memerlukan cara alternatif yang menarik untuk dapat menyerap topik materi pada mata kuliah *Operating System*. Beberapa aplikasi untuk kebutuhan representasi simulasi data baik yang berbentuk *desktop application* seperti Mathematica®, Matlab®, Maple®, GNU Octave, GNUPlot, PyPlot, Sage maupun yang bersifat Software-as-a-Service (SaaS) seperti Sage in cloud, Wolfram Mathematica®, Simul8®merupakan aplikasi simulasi yang bersifat umum digunakan dalam rangka memberikan hasil simulasi sesuai kebutuhan akademik, organisasi nirlaba atau perusahaan yang memiliki lisensi *proprietary* berbayar dan berlisensi GNU General Public License (GPL) tidak berbayar [**?**].

Penggunaan aplikasi untuk kebutuhan akademik dapat memanfaatkan PyPlot yang memanfaatkan pustaka bahasa pemrograman Python berlisensi GPL dan memiliki fitur yang lengkap untuk kebutuhan akademik sebagai terobosan terkait ketersediaan aplikasi untuk representasi simulasi data yang bersifat modular, adaptif dan fungsional. Saat ini PyPlot memiliki sifat *multiplatform* untuk mengurangi keterbatasan pada penggunaan multi *Operating Systems* yang bersifat heterogen. Adapun beberapa tantangan pengguna PyPlot adalah pada mekanisme pemanfaatan kekayaan pustaka Python dan GNU Plot untuk menghasilkan representasi simulasi data, mulai dari penggunaan variabel hingga topik terkait simulasi *Operating Systems*.

Penulis mencoba untuk memberikan panduan penggunaan Pytthon yang diharapkan dapat bermanfaat bagi para mahasiswa sebagai pengguna aplikasi modern yang membutuhkan aplikasi yang tidak berbayar untuk kebutuhan representasi simulasi data pada materi *Operating Systems*. PyPlot dapat menangani berbagai macam kebutuhan representasi simulasi data secara efisien dan mampu melakukan pengelolaan beragam tampilan hasil, walaupun perlu dilakukan pengaturan lebih lanjut terkait pendekatan *scripting* pada *Command Line Interface* (CLI) PyPlot yang memiliki beban komputasi relatif rendah apabila dibandingkan dengan aplikasi simulasi data lainnya.

Daftar Isi

Bab 1

Pendahuluan

1.1 Overview

Penggunaan aplikasi yang dibangun menggunakan bahasa pemrograman Python dan pustaka GNU is Not UNIX (GNU) Plot sehingga membentuk PyPlot, atau sering disebut pada pengembangan lebih lanjut berupa Matplotlib sebagai perangkat bantu visualisasi untuk Python. Matplotlib merupakan pustaka yang dibangun berdasarkan array Numpy dan didesain untuk pustaka yang khusus digunakan untuk memberikan fasilitas bagi kepentingan ilmu pengetahuan berupa SciPy. Pada awalnya diperkenalkan oleh John Hunter pada tahun 2022 untuk keperluan perbaikan iPython agar dapat berfungsi seperti halnya *interactive console* pada Matlab dengan memanfaatkan pustaka GNU Plot [1].

PyPlot memiliki fasilitas penting untuk dapat berjalan dengan baik pada banyak sistem operasi dan pustaka grafis serta tipe output visualisasi, sehingga memperluas cakupan pengguna dan pengembang PyPlot. Kondisi tersebut diperkuat dengan dukungan pengembangan banyak pustaka untuk bahasa pemrograman Python yang menjadi kekuatan PyPlot sehingga banyak digunakan oleh akademisi dalam rangka pengembangan ilmu pengetahuan, khususnya untuk visualisasi hasil penelitian dalam bidang informatika, kedokteran, pemodelan sistem rekomendasi dan lainnya [2].

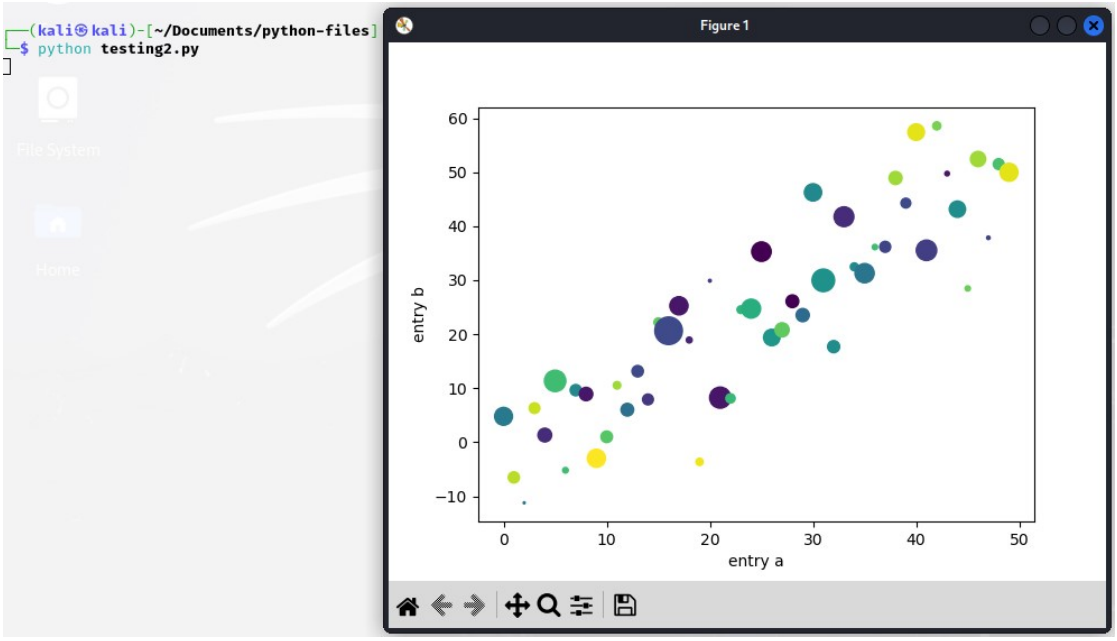
1.2 Mengapa PyPlot ?

Dewasa ini terdapat bermacam-macam bahasa pemrograman komputer yang dapat menyelesaikan semua permasalahan teknis seperti penggunaan C++, Java, Python. Beberapa bahasa pemrograman tersebut merupakan bahasa pemrograman yang menjadi standar industri untuk membangun aplikasi. Permasalahannya adalah ketika bahasa pemrograman tersebut digunakan untuk membangun aplikasi menggunakan metode *prototype* atau sedang berada pada fase konsep pengembangan, seringkali terbentur kebutuhan terkait kecepatan implementasi dan waktu yang sempit untuk melakukan implementasi suatu algoritma tertentu.

Penggunaan PyPlot dirancang untuk memecahkan masalah sempitnya waktu pemenuhan visualisasi tersebut dengan menyelesaikan perhitungan segera dan menampilkan visualisasi hasil secepat mungkin. Pada saat praktisi ataupun akademisi menggunakan Matlab, maka *script* hasil pekerjaan yang digunakan pada Matlab perlu dilakukan penyesuaian agar dapat diterapkan untuk PyPlot terkait *syntax* berekstensi file .m yang digunakan pada Matlab. Aplikasi Matlab ®dari Mathworks ®bersifat fleksibel, dapat diperluas cakupan pemecahan masalah simulasinya [3], berkinerja tinggi dengan berlangganan biaya lisensi aplikasi *proprietary* untuk dapat menggunakan aplikasi Matlab. Sedangkan penggunaan PyPlot dapat dilakukan dengan proses unduh, instalasi dan digunakan tanpa harus berlangganan biaya lisensi pakai

aplikasi karena sifat lisensi GNU Public License (GPL), sehingga bersifat gratis untuk penggunaannya akan tetapi menjadi perangkat tepat guna, khususnya pada visualisasi hasil peneltian seperti peran ilmu komputer pada bidang kedokteran dalam pengelolaan penyakit *cardiovascular*, simulasi untuk menggambarkan representasi data manajemen keuangan [4], visualisasi hasil observasi perilaku interaksi [5] dan bidang lainnya [6].

Pembaca dapat melihat tampilan PyPlot yang mnerupakan pustaka Python telah terinstalasi pada sistem operasi Kali Linux 64bit seperti terlihat pada gambar berikut :



Gambar 1.1: Lingkungan kerja PyPlot menggunakan *console*

Hasil visualisasi pada Gambar 1.1 merupakan eksekusi file Python yaitu testing2 dengan ekstensi .py dari listing berikut ini:

Listing 1.1: Konten file testing2.py sebagai visualisasi pada PyPlot

```
import matplotlib.pyplot as plt
import numpy as np
np.random.seed(19680801)
data = {'a': np.arange(50),
        'c': np.random.randint(0, 50, 50),
        'd': np.random.randn(50)}
data['b'] = data['a'] + 10 * np.random.randn(50)
data['d'] = np.abs(data['d']) * 100

#fig, ax = plt.subplots(figsize=(5, 2.7), layout='constrained')
plt.scatter('a', 'b', c='c', s='d', data=data)
plt.xlabel('entry a')
plt.ylabel('entry b')
plt.show()
```

Berdasarkan listing 1.1 tersebut, pembaca dapat melihat penggunaan pustaka matplotlib.pyplot dan pustaka numpy untuk menampilkan visualisasi fungsi acak dari data yang diberikan pada listing tersebut,

kemudian pengaturan cara tampilan dengan memanfaatkan fungsi scatter berikut label pada sumbu x dan sumbu y. Kesederhanaan penulisan listing untuk menghasilkan visualisasi dengan cepat menggunakan bahasa pemrograman Python dan pustaka yang tepat merupakan salah satu keunggulan PyPlot dalam rangka perluasan pengguna dan pengembangan pustaka lebih lanjut.

Bab 2

Mulai Menggunakan PyPlot

Sebelum pembaca melakkukan penulisan perintah pada *interactive console* Python atau melakukan penulisan pada *file*, terdapat beberapa langkah untuk mempermudah penulisan tersebut, dapat kita lihat pada pembahasan berikut ini.

- Import pustaka yagn diperlukan sekaligus singkatan penyederhanaan penulisan untuk mempermudah eksekusi;
- Visualisasi pada listing dengan memanfaatkan *show*() ;
- Penyimpanan file hasil eksekusi listing dengan menggunakan perintah $savefig()$.

Penjelasan lebih lanjut mengenai import pustaka yang diperlukan sekaligus melakukan singkatan, dapat pembaca lihat pada listing 2.1 berikut ini:

Listing 2.1: Konten file testing2a.py sebagai penyingkat pada PyPlot

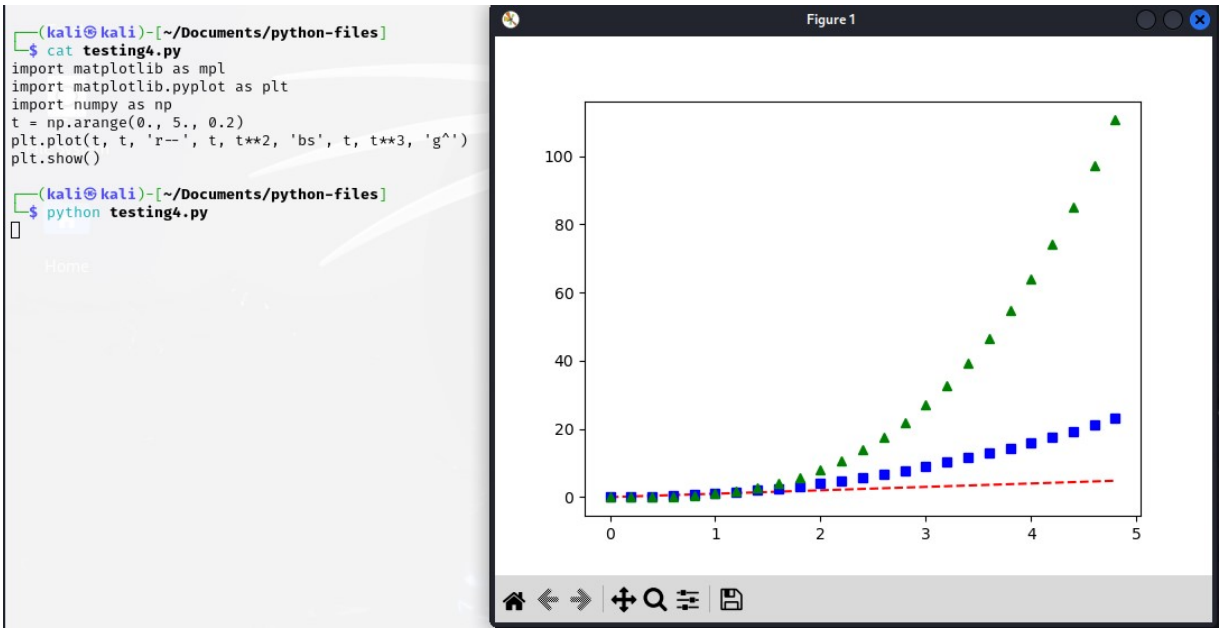
```
1 import matplotlib as mpl
2 import matplotlib.pyplot as plt
3 import numpy as np
```

Listing 2.2 berikut ini menjelaskan visualisasi pada listing dengan memanfaatkan perintah *show*().

Listing 2.2: Konten file testing2a.py sebagai penyingkat pada PyPlot

```
1 import matplotlib as mpl
2 import matplotlib.pyplot as plt
3 import numpy as np
4 t = np.arange(0., 5., 0.2)
5 plt.plot(t, t, 'r--', t, t**2, 'bs', t, t**3, 'g^')
6 plt.show()
```

Hasil eksekusi Listing 2.2 dapat pembaca lihat pada gambar berikut ini:



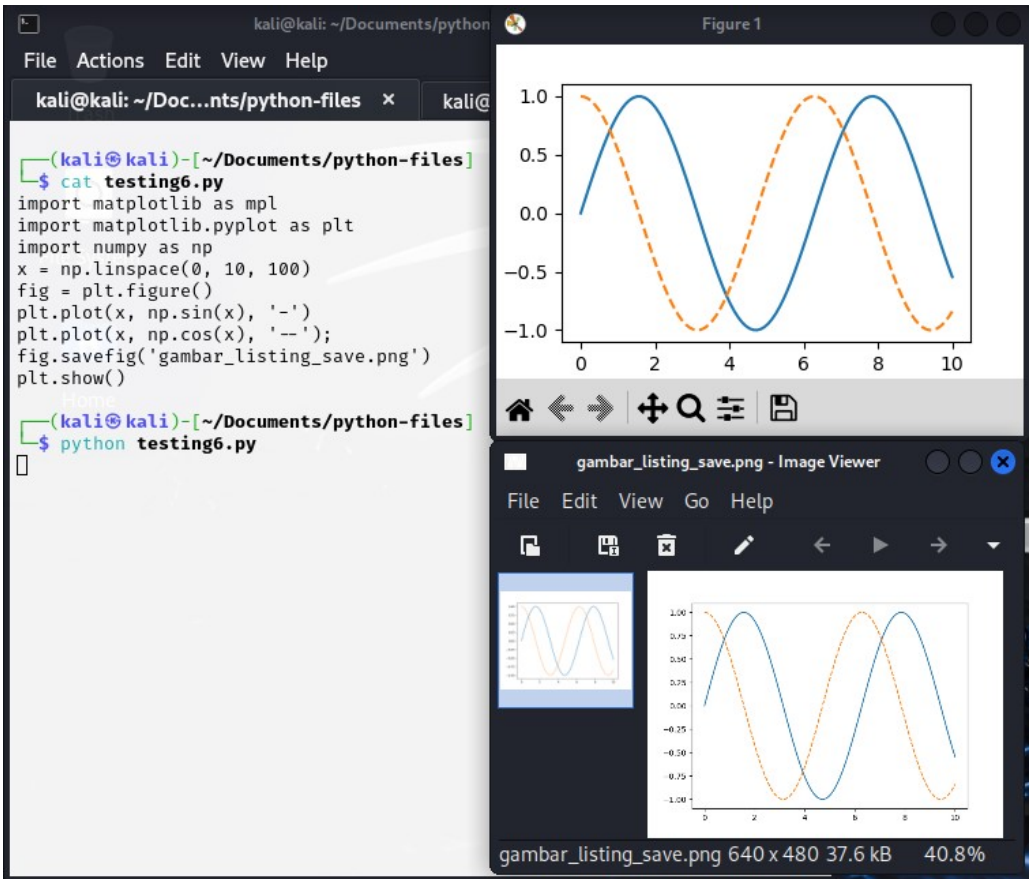
Gambar 2.1: Hasil eksekusi Listing 2.2 menggunakan *show*()

Listing 2.3 berikut ini menjelaskan mengenai penyimpanan file gambar sekaligus menampilkan hasil eksekusi PyPlot menggunakan Python.

Listing 2.3: Konten file testing2a.py sebagai penyingkat pada PyPlot

```
1 import matplotlib as mpl
2 import matplotlib.pyplot as plt
3 import numpy as np
4 x = np.linspace(0, 10, 100)
5 fig = plt.figure()
6 plt.plot(x, np.sin(x), '-')
7 plt.plot(x, np.cos(x), '--');
8 fig.savefig('gambar_listing_save.png')
9 plt.show()
```

Hasil eksekusi Listing 2.3 dapat pembaca lihat pada gambar berikut ini, bagian atas merupakan hasil visualisasi dari eksekusi, sedangkan bagian bawah merupakan visualisasi dari file $gambar_listing_save.png$ menggunakan aplikasi pembaca gambar Ristretto :



Gambar 2.2: Hasil eksekusi Listing 2.3 menyimpan dan menampilkan gambar

2.1 Grid dan Visualisasi Fungsi

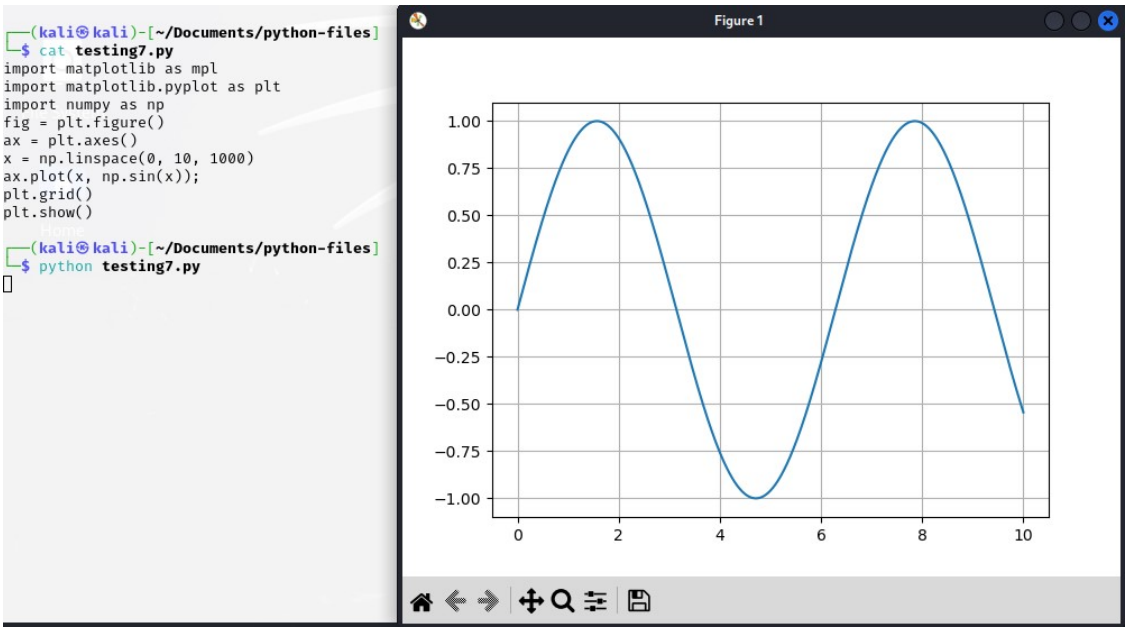
Visualisasi plot sederhana dapat digambarkan pada fungsi $y = f(x)$ yang dapat digambarkan dengan garis pandu grid (*grid*()) pada penggunaan listing berikut ini.

Listing 2.4: Visualisasi plot dan grid untuk menampilkan fungsi sinus pada PyPlot

```
1 import matplotlib as mpl
2 import matplotlib.pyplot as plt
3 import numpy as np
4 fig = plt.figure()
5 ax = plt.axes()
6 x = np.linspace(0, 10, 1000)
7 ax.plot(x, np.sin(x));
8 plt.grid()
9 plt.show()
```

Hasil dari eksekusi Listing 2.4 tersebut dapat pembaca lihat pada tampilan berikut ini, dimana perintah *plt.grid*() menampilkan garis bantu dan fungsi *np.sin*(x) menampilkan grafik sinus dari data yang diberikan yaitu (0, 10, 1000).

Apabila pembaca akan melakukan *merge* grafik dari fungsi Sinus dan fungsi Cosinus pada satu grafik, maka listing 2.5 berikut ini dapat memberikan visualisasi tersebut.



Gambar 2.3: Hasil eksekusi Listing 2.4 PyPlot menampilkan grid dan grafik fungsi Sinus

Listing 2.5: Visualisasi plot dan grid untuk menampilkan fungsi sinus pada PyPlot

```
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
fig = plt.figure()
ax = plt.axes()
x = np.linspace(0, 10, 10000)
ax.plot(x, np.sin(x));
ax.plot(x, np.cos(x));
plt.grid()
plt.show()
```

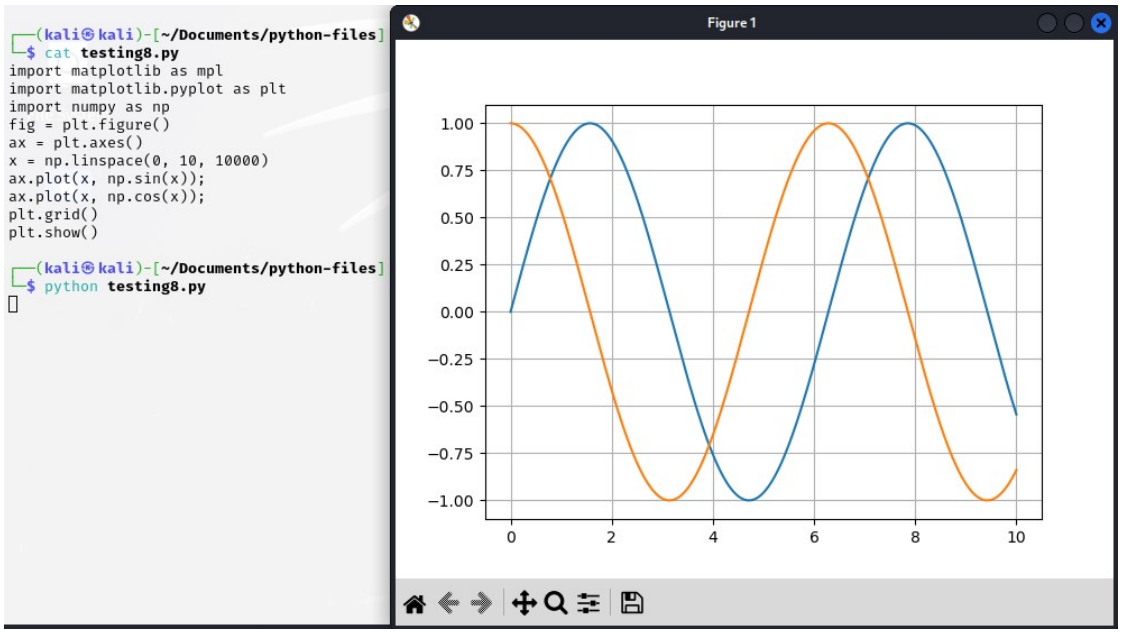
Hasil dari eksekusi Listing 2.5 tersebut dapat pembaca lihat pada tampilan berikut ini, dimana perintah *plt.grid*() menampilkan garis bantu dan fungsi $np.sin(x)$ dan $np.cos(x)$ menampilkan grafik sinus dan cosinus dari data yang diberikan yaitu (0, 10, 10000).

2.2 Visualisasi Variasi Pewarnaan Fungsi

Apabila pembaca ingin melakukan visualisasi grafik fungsi Sinus secara berulang dengan variasi warna, maka Listing 2.6 berikut ini dapat digunakan untuk keperluan tersebut.

Listing 2.6: Visualisasi plot dan grid untuk menampilkan fungsi sinus dan warna berulang pada PyPlot

```
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
fig = plt.figure()
ax = plt.axes()
x = np.linspace(0, 10, 10000)
```



Gambar 2.4: Hasil eksekusi Listing 2.5 PyPlot menampilkan grid dan grafik fungsi Sinus dan Cosinus

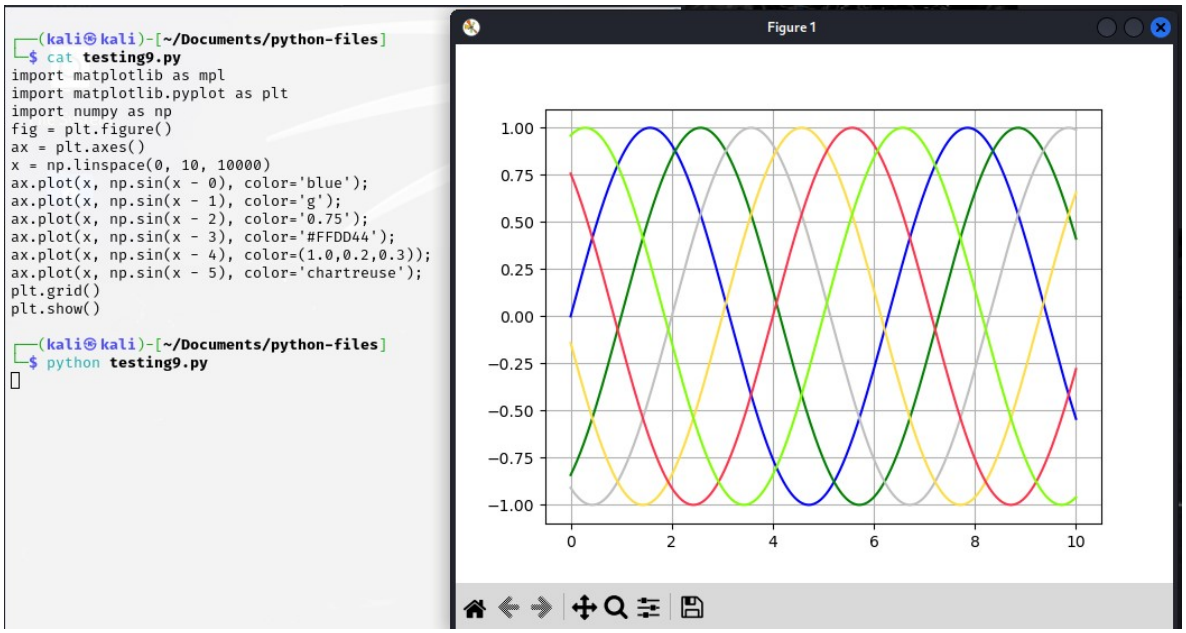
```
ax.plot(x, np.sin(x - 0), color='blue');
ax.plot(x, np.sin(x - 1), color='g');
ax.plot(x, np.sin(x - 2), color='0.75');
ax.plot(x, np.sin(x - 3), color='#FFDD44');
ax.plot(x, np.sin(x - 4), color=(1.0,0.2,0.3));
ax.plot(x, np.sin(x - 5), color='chartreuse');
plt.grid()
plt.show()
```

Hasil dari eksekusi Listing 2.6 tersebut dapat pembaca lihat pada tampilan berikut ini, dimana perintah $plt.grid()$ menampilkan garis bantu dan fungsi $ax.plot(x, np.sin(x - 0), color =' blue')$ menampilkan grafik sinus dengan warna yang bervariasi dari sumbu x = 0 hingga x = 5 berdasarkan data yang diberikan yaitu (0, 10, 10000).

Selanjutnya pembaca dapat melakukan visualisasi grafik dengan bentuk garis yang berbeda-beda, terutama apabila pembaca membutuhkan representasi data yang bermacam-macam pada suatu grafik, seperti terlihat pada Listing 2.7 berikut ini.

Listing 2.7: Visualisasi plot dan grid untuk menampilkan fungsi sinus dan warna berulang dengan variasi bentuk garis pada PyPlot

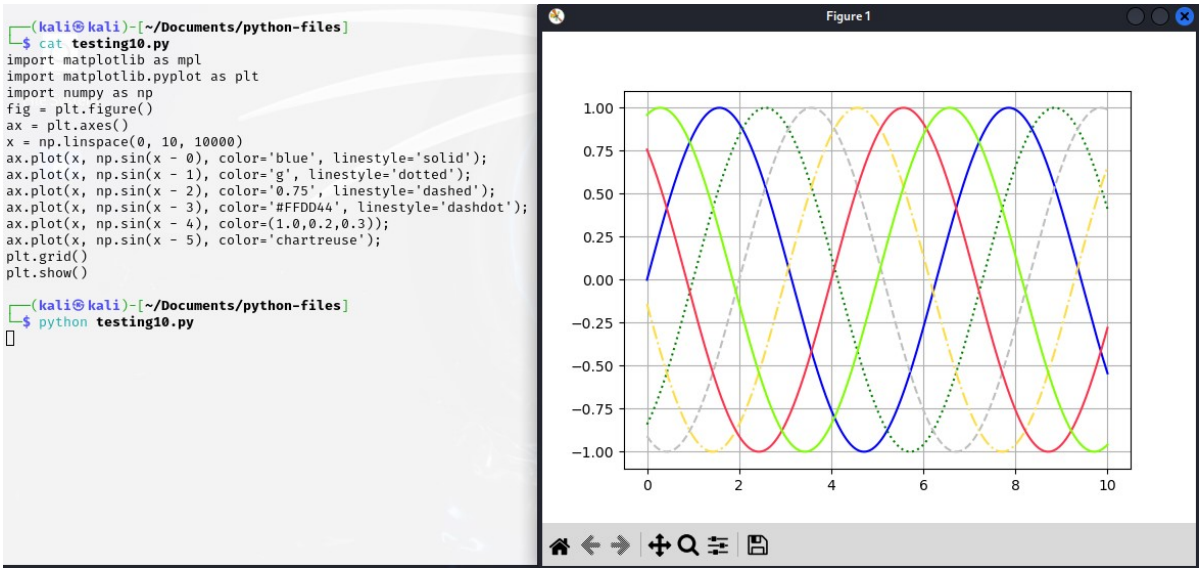
```
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
fig = plt.figure()
ax = plt.axes()
x = np.linspace(0, 10, 10000)
ax.plot(x, np.sin(x - 0), color='blue', linestyle='solid');
ax.plot(x, np.sin(x - 1), color='g', linestyle='dotted');
ax.plot(x, np.sin(x - 2), color='0.75', linestyle='dashed');
ax.plot(x, np.sin(x - 3), color='#FFDD44', linestyle='dashdot');
```



Gambar 2.5: Hasil eksekusi Listing 2.6 PyPlot menampilkan grid dan grafik fungsi Sinus dengan warna garis bervariasi

```
ax.plot(x, np.sin(x - 4), color=(1.0,0.2,0.3));
ax.plot(x, np.sin(x - 5), color='chartreuse');
plt.grid()
plt.show()
```

Hasil dari eksekusi Listing 2.7 tersebut dapat pembaca lihat pada tampilan berikut ini, dimana fungsi $ax.plot(x, np.sin(x-0), color =' blue', linestyle =' solid')$; menampilkan grafik sinus dengan warna yang bervariasi dan bentuk garis berbeda dari sumbu x = 0 hingga x = 5 berdasarkan data yang diberikan yaitu (0, 10, 10000).



Gambar 2.6: Hasil eksekusi Listing 2.7 PyPlot menampilkan grid dan grafik fungsi Sinus dengan warna garis bervariasi dan bentuk garis berbeda

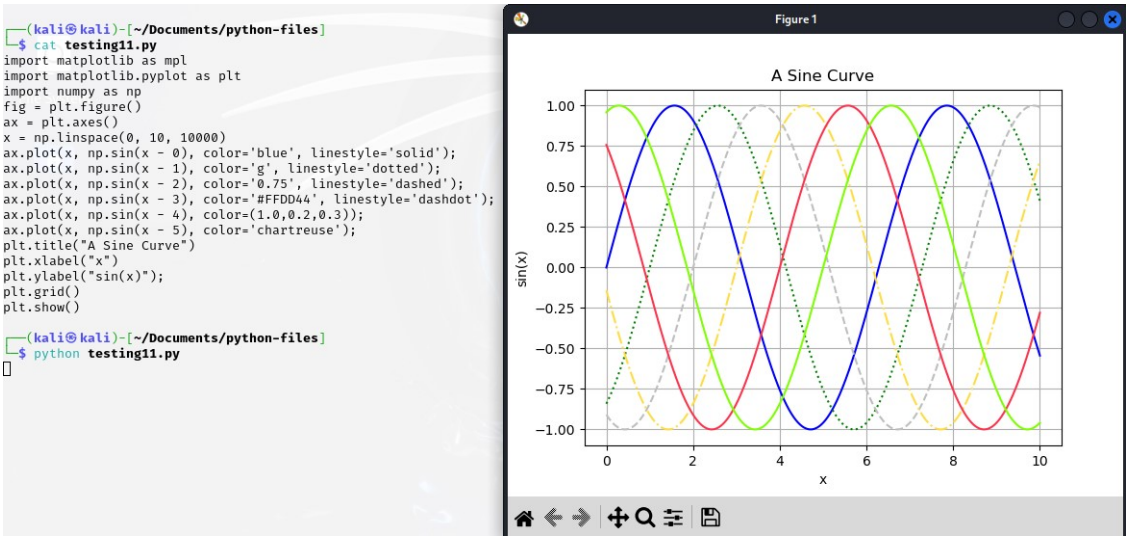
2.3 Visualisasi Label Grafik dari Fungsi

Pada saat pembaca ingin memberikan label pada grafik untuk sumbu X dan sumbu Y, maka dapat menggunakan fungsi *xlabel*()atau *ylabel*(), seperti terlihat pada Listing 2.8 berikut ini.

Listing 2.8: Penggunaan Label pada Sumbu X dan Sumbu Y grafik pada PyPlot

```
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
fig = plt.figure()
ax = plt.axes()
x = np.linspace(0, 10, 10000)
ax.plot(x, np.sin(x - 0), color='blue', linestyle='solid');
ax.plot(x, np.sin(x - 1), color='g', linestyle='dotted');
ax.plot(x, np.sin(x - 2), color='0.75', linestyle='dashed');
ax.plot(x, np.sin(x - 3), color='#FFDD44', linestyle='dashdot');
ax.plot(x, np.sin(x - 4), color=(1.0,0.2,0.3));
ax.plot(x, np.sin(x - 5), color='chartreuse');
plt.title("A Sine Curve")
plt.xlabel("x")
plt.ylabel("sin(x)");
plt.grid()
plt.show()
```

Hasil dari eksekusi Listing 2.8 tersebut dapat pembaca lihat pada tampilan berikut ini, dimana fungsi *plt.xlabel*("x") dan *plt.ylabel*("$sin(x)$"); menampilkan label pada sumbu X dan Sumbu Y dari grafik sinus dengan warna yang bervariasi dan bentuk garis berbeda dari sumbu x = 0 hingga x = 5 berdasarkan data yang diberikan yaitu (0, 10, 10000).



Gambar 2.7: Hasil eksekusi Listing 2.8 PyPlot menampilkan grid, grafik fungsi Sinus dengan warna garis bervariasi dan bentuk garis berbeda serta label pada sumbu X dan sumbu Y

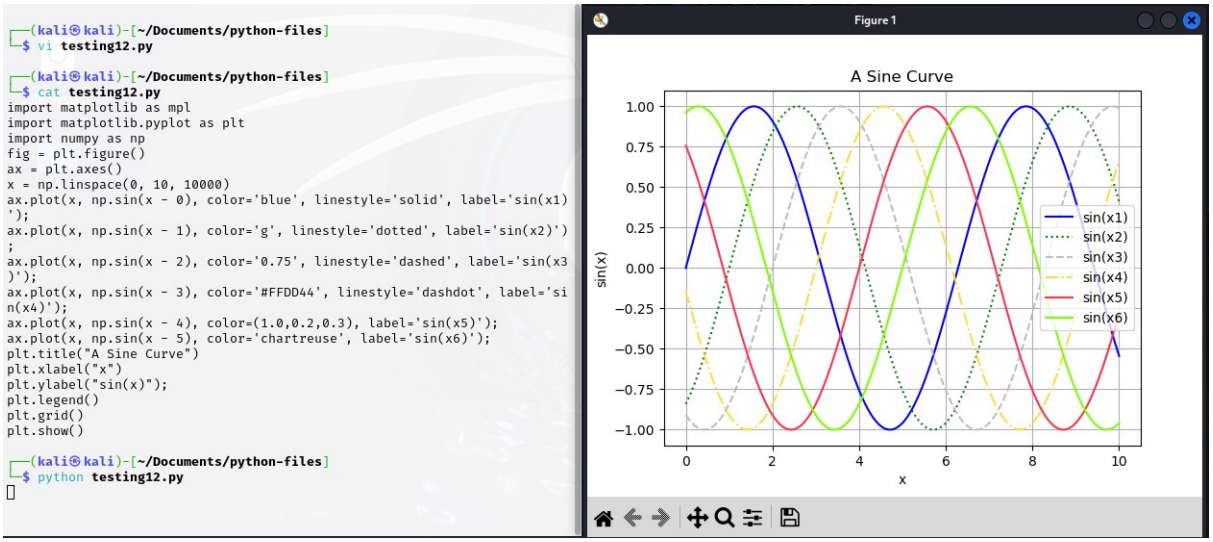
2.4 Visualisasi Label dan Legend Grafik dari Fungsi

Pada saat pembaca ingin memberikan label pada masing-masing kurva untuk grafik untuk sumbu X dan sumbu Y, maka dapat menggunakan fungsi *legend*() dan *label* =″, seperti terlihat pada Listing 2.9 berikut ini.

Listing 2.9: Penggunaan Label dan Legend pada Sumbu X Sumbu Y dan keterangan kurva pada PyPlot

```
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
fig = plt.figure()
ax = plt.axes()
x = np.linspace(0, 10, 10000)
ax.plot(x, np.sin(x - 0), color='blue', linestyle='solid', label='sin(x1)');
ax.plot(x, np.sin(x - 1), color='g', linestyle='dotted', label='sin(x2)');
ax.plot(x, np.sin(x - 2), color='0.75', linestyle='dashed', label='sin(x3)');
ax.plot(x, np.sin(x - 3), color='#FFDD44', linestyle='dashdot', label='sin(x4)');
ax.plot(x, np.sin(x - 4), color=(1.0,0.2,0.3), label='sin(x5)');
ax.plot(x, np.sin(x - 5), color='chartreuse', label='sin(x6)');
plt.title("A Sine Curve")
plt.xlabel("x")
plt.ylabel("sin(x)");
plt.legend()
plt.grid()
plt.show()
```

Hasil dari eksekusi Listing 2.9 tersebut dapat pembaca lihat pada tampilan berikut ini, dimana fungsi *plt.xlabel*(”x”) dan *plt.ylabel*(”$sin(x)$”); menampilkan label pada sumbu X dan Sumbu Y serta Legend pada masing-masing kurva dari grafik sinus dengan warna yang bervariasi dan bentuk garis berbeda dari sumbu x = 0 hingga x = 5 berdasarkan data yang diberikan yaitu (0, 10, 10000).



Gambar 2.8: Hasil eksekusi Listing 2.9 PyPlot menampilkan grid, grafik fungsi Sinus dengan warna garis bervariasi dan bentuk garis berbeda serta label pada sumbu X dan sumbu Y serta legend masing-masing kurva

Bab 3

Simulasi Penjadwalan pada *Operating System*

Round Robin Scheduling merupakan salah satu metode yang digunakan pada sistem operasi dalam rangka pengelolaan waktu eksekusi dari beberapa proses yang saling bersaing mendapatkan sumber daya komputasi (prosesor, memory dan space storage). Penyebutan *Round Robin* disebabkan sistem operasi melakukan rotasi kepada semua proses sesuai dengan alokasi waktu secara tetap atau secara kuantum terhadap masing-masing proses dengan mengesampingkan prioritas.

Tujuan dari metode penjadwalan *Round Robin* adalah memastikan bahwa semua proses mendapatkan kesempatan yang sama untuk dijalankan dan mendapat sumber daya komputasi, sehingga menepati asas *fairness* antar proses. Beberapa sifat dari *Round Robin Scheduling* yang menjadi keunggulan teknik penjadwalan pada sistem operasi adalah aspek *fairness, simplicity, responsiveness*. Sedangkan yang menjadi kekurangan dari *Round Robin Scheduling* adalah pada aspek *overhead, utilisasi* pada saat irisan waktu yang diberikan menjadi terlalu lama.

3.1 Visualisasi Round Robin Scheduling

Terdapat 2 (dua) kondisi yang sering ditemui untuk dipecahkan pada *Round Robin Scheduling*, yaitu:

1. Proses dengan waktu kedatangan sama
2. Proses dengan waktu kedatangan berbeda

Pada kondisi proses dengan waktu kedatangan sama, maka waktu tunggu rata dan waktu penyelesaian dapat dicapai dengan formulasi sederhana [7], yaitu:

$$WaktuPenyelesaian = WaktuPenyelesaian - WaktuKedatangan$$

$$WaktuTunggu = WaktuPenyelesaian - WaktuPecah$$

Sedangkan pada kondisi proses dengan waktu kedatangan berbeda, maka waktu tunggu rata-rata dan waktu penyelesaian dapat dihitung dengan mempertimbangkan antrian dan proses pada antrian, dengan formulasi yaitu:

$$TurnaroundTime = CompletionTime - ArrivalTime$$

$$WaitingTime = TurnaroundTime - BurstTime$$

$$AverageTurnaroundTime = TotalTurnaroundTime/ProcessFrequency$$

$$AverageWaitingTime = TotalWaitingTime/ProcessFrequency$$

3.1.1 *Round Robin Scheduling* untuk Proses dengan Waktu Kedatangan Sama

Pembaca dapat melihat Listing 3.1 berikut ini untuk lebih memahami *Round Rpbin Scheduling* untuk penanganan proses dengan waktu kedatangan yang sama.

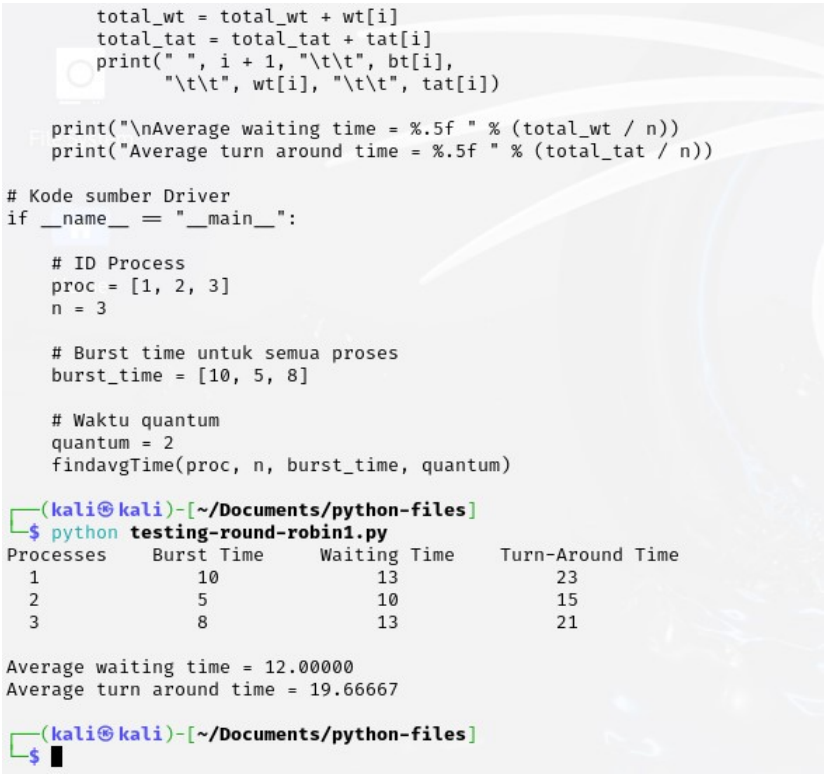
Listing 3.1: Round Rpbin Scheduling untuk penanganan proses dengan waktu kedatangan yang sama

```
# Fungsi untuk mencari waiting time pada semua proses
def findWaitingTime(processes, n, bt,
                    wt, quantum):
    rem_bt = [0] * n
    # Penyalinan burst time kedalam array rt[]
    for i in range(n):
        rem_bt[i] = bt[i]
    t = 0  # Current time
    # Mekanisme menjaga proses yang sedang berjalan berada pada aturan penjadwalan Round Robin sampai semuanya tidak diselesaikan
    while(1):
        done = True
        # Menjalankan proses satu demi satu secara berulang
        for i in range(n):

            # Apbila burst time pada proses lebih besar dari 0 maka hanya memerlukan proses selanjutnya
            if (rem_bt[i] > 0):
                done = False  # There is a pending process

                if (rem_bt[i] > quantum):
                    # Meningkatkan nilai t untuk menunjukkan sebarapa banyak waktu untuk menjalankan proses
                    t += quantum

                    # Mengurangi burst_time dari proses saat ini berdasarkan quantum
                    rem_bt[i] -= quantum

                # Apabila burst time lebih kecil sama dengan quantum maka melakukan siklus akhir pada proses
                else:

                    # Meningkatkan nilai t untuk mencari seberapa banyak waktu yang dibutuhkan oleh suatu proses
                    t = t + rem_bt[i]

```

```
                    # Waiting time merupakan pengurangan dari current time dikurangi waktu
                         yang digunakan untuk menjalankan proes
                    wt[i] = t - bt[i]

                    # Pada saat proses telah dieksekusi sepenuhnya maka burst time menjadi
                         0
                    rem_bt[i] = 0

        # Jika semua proses menjadi bernilai 0
        if (done == True):
            break

# Fungsi untuk menghitung turn around time
def findTurnAroundTime(processes, n, bt, wt, tat):

    # Perhitungan turnaround time
    for i in range(n):
        tat[i] = bt[i] + wt[i]


# Fungsi untuk menghitung waktu tunggu rata-rata dan waktu kerja.
def findavgTime(processes, n, bt, quantum):
    wt = [0] * n
    tat = [0] * n

    # Fungsi untuk mendapatkan waktu tunggu dari semua proses
    findWaitingTime(processes, n, bt,
                    wt, quantum)

    # Fungsi untuk mendapatkan waktu kerja semua proses
    findTurnAroundTime(processes, n, bt,
                       wt, tat)

    # Menampilkan proses terhadap detail atribut proses
    print("Processes    Burst Time     Waiting",
          "Time    Turn-Around Time")
    total_wt = 0
    total_tat = 0
    for i in range(n):

        total_wt = total_wt + wt[i]
        total_tat = total_tat + tat[i]
        print(" ", i + 1, "\t\t", bt[i],
              "\t\t", wt[i], "\t\t", tat[i])

    print("\nAverage waiting time = %.5f " % (total_wt / n))
```

```
76      print("Average turn around time = %.5f " % (total_tat / n))
77
78  # Kode sumber Driver
79  if __name__ == "__main__":
80
81      # ID Process
82      proc = [1, 2, 3]
83      n = 3
84
85      # Burst time untuk semua proses
86      burst_time = [10, 5, 8]
87
88      # Waktu quantum
89      quantum = 2
90      findavgTime(proc, n, burst_time, quantum)
```

Apabila dijalankan, maka hasil dari Listing 3.1 tersebut adalah seperti terlihat pada gambar berikut ini:



Gambar 3.1: Hasil eksekusi Listing 3.1 Round Robin Scheduling dengan waktu kedatangan sama

3.1.2 *Round Robin Scheduling* untuk Proses dengan Waktu Kedatangan Berbeda

Pembaca dapat melihat Listing 3.2 berikut ini untuk lebih memahami *Round Rpbin Scheduling* untuk penanganan proses dengan waktu kedatangan yang berbeda.

Listing 3.2: Round Rpbin Scheduling untuk penanganan proses dengan waktu kedatangan yang berbeda

```
# Listing Python untuk Round Robin Scheduling dengan waktu kedatangan berbeda
def queueUpdation(queue, timer, arrival, n, maxProccessIndex):
    zeroIndex = -1
    for i in range(n):
        if(queue[i] == 0):
            zeroIndex = i
            break

    if(zeroIndex == -1):
        return
    queue[zeroIndex] = maxProccessIndex + 1


def checkNewArrival(timer, arrival, n, maxProccessIndex, queue):
    if(timer <= arrival[n-1]):
        newArrival = False
        for j in range(maxProccessIndex+1, n):
            if(arrival[j] <= timer):
                if(maxProccessIndex < j):
                    maxProccessIndex = j
                    newArrival = True

        # Menambahkan indeks jika terdapat proses yang datang
        if(newArrival):
            queueUpdation(queue, timer, arrival, n, maxProccessIndex)


def queueMaintainence(queue, n):
    for i in range(n-1):
        if(queue[i+1] != 0):
            queue[i], queue[i+1] = queue[i+1], queue[i]


timer, maxProccessIndex = 0, 0
avgWait, avgTT = 0, 0
print("\nEnter the time quanta :", end=" ")
tq = int(input())
print("\nEnter the number of processes :", end=" ")
n = int(input())
arrival = [0]*n
burst = [0]*n
```

```
wait = [0]*n
turn = [0]*n
queue = [0]*n
temp_burst = [0]*n
complete = [False]*n
print("\nEnter the arrival time of the processes :", end=" ")
for i in range(n):
    arrival[i] = int(input())

print("\nEnter the burst time of the processes :", end=" ")
for i in range(n):
    burst[i] = int(input())
    temp_burst[i] = burst[i]

for i in range(n):
        # Initializing the queue and complete array
    complete[i] = False
    queue[i] = 0

while(timer < arrival[0]):
        # Incrementing Timer until the first process arrives
    timer += 1
queue[0] = 1

while(True):
    flag = True
    for i in range(n):
        if(temp_burst[i] != 0):
            flag = False
            break

    if(flag):
        break

    for i in range(n and queue[i] != 0):
        ctr = 0
        while((ctr < tq) and (temp_burst[queue[0]-1] > 0)):
            temp_burst[queue[0]-1] -= 1
            timer += 1
            ctr += 1

            # Melakukan update antrian sampai semua proses sampai pada sistem
            checkNewArrival(timer, arrival, n, maxProccessIndex, queue)

        if((temp_burst[queue[0]-1] == 0) and (complete[queue[0]-1] == False)):
            # turn currently stores exit times
```

```
            turn[queue[0]-1] = timer
            complete[queue[0]-1] = True

        # Melakukan pemeriksaan kondisi prosesor apakah sedang dalam kondisi idle
        idle = True
        if(queue[n-1] == 0):
            for k in range(n):
                if(queue[k] != 0):
                    if(complete[queue[k]-1] == False):
                        idle = False
        else:
            idle = False

        if(idle):
            timer += 1
            checkNewArrival(timer, arrival, n, maxProccessIndex, queue)

        # Mengelola isian proses setelah setiap knodisi prasyarat terpenuhi pada antrian
        queueMaintainence(queue, n)

for i in range(n):
    turn[i] = turn[i] - arrival[i]
    wait[i] = turn[i] - burst[i]

print("\nProgram No.\tArrival Time\tBurst Time\tWait Time\tTurnAround Time\n")

for i in range(n):
    print(i+1, "\t\t", arrival[i], "\t\t", burst[i],
          "\t\t", wait[i], "\t\t", turn[i], "\n")
for i in range(n):
    avgWait += wait[i]
    avgTT += turn[i]
print("\nAverage wait time : ", (avgWait//n))
print("\nAverage Turn Around Time : ", (avgTT//n))
```

Bibliography

[1] J. VanderPlas, *Python Data Science Handbook, 2nd Edition*, 2nd ed. O'Reilly Media, Inc., 2022. 3

[2] Y. A. Seliverstov, S. A. Seliverstov, N. V. Podoprigora, A. L. Starichenkov, and R. S. Naryshkin, "Using the model of a functional rationalizer of consumer behavior in recommendation systems for managing the transport activity of the urban population," in *2020 XXIII International Conference on Soft Computing and Measurements (SCM)*, 2020, pp. 169–173. 3

[3] M. Metcalf, P. Andrés-Martínez, and N. Fitzpatrick, "Realizing quantum kernel models at scale with matrix product state simulation," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis*, ser. SC '24. IEEE Press, 2024. [Online]. Available: https://doi.org/10.1109/SC41406.2024.00086 3

[4] Q. Lin and Y. Yu, "Teaching design and application of python in management accounting—taking cost-volume-profit analysis as an example," in *Proceedings of the 2023 International Conference on Information Education and Artificial Intelligence*, ser. ICIEAI '23. New York, NY, USA: Association for Computing Machinery, 2024, p. 571–576. [Online]. Available: https://doi.org/10.1145/3660043.3660145 4

[5] A. Akkuzu, N. Calvo-Barajas, and G. Castellano, "Behavioural observations as objective measures of trust in child-robot interaction: Mutual gaze," in *Proceedings of the 11th International Conference on Human-Agent Interaction*, ser. HAI '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 452–454. [Online]. Available: https://doi.org/10.1145/3623809.3623960 4

[6] K. Gupta, Shalini, P. Rajawat, A. Srivastava, and A. Dixit, "Transforming cardiovascular disease management through advanced ai and ml-driven predictive modeling and data visualization," in *2024 1st International Conference on Advanced Computing and Emerging Technologies (ACET)*, 2024, pp. 1–5. 4

[7] Madhur, "Round robin scheduling in operating system," accessed January 5th, 2025. [Online]. Available: https://www.geeksforgeeks.org/round-robin-scheduling-in-operating-system/ 14