

TUGAS 6
PEMROGRAMAN JARINGAN B081



Kelompok 14 :

21081010070 Achmad Rozy Priambodo

21081010177 Dzikra Fadhila Akbar Setiawan

21081010312 Alif Wildan Azzahran

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS PEMBANGUNAN NASIONAL "VETERAN"
JAWA TIMUR

2024

Soal Tugas Materi 6

1. Buatlah 5 proses pada program Latihan 1 dengan rincian delay sebagai berikut:
Proses 1: 2
Proses 2: 3
Proses 3: 4
Proses 4: 5
Proses 5: 6
Amati proses yang terjadi dan catat hasilnya
2. Buatlah 5 thread pada program Latihan 3 dengan delay 1 dan melakukan count down sebagai berikut:
Thread 1: 3
Thread 2: 6
Thread 3: 9
Thread 4: 12
Thread 5: 15
3. Buatlah 2 thread dengan delay 1 dan melakukan proses sebagai berikut:
Thread 1: mencetak bilangan ganjil dari 1..10
Thread 2: mencetak bilangan genap dari 1..10

Jawaban :

1. Source Code

```
#!/usr/bin/python
import time

# Define a function
def print_time(proses, delay):
    count = 0
    while count < 5:
        time.sleep(delay)
        count += 1
        print("%s: %s" % (proses, time.ctime(time.time())))

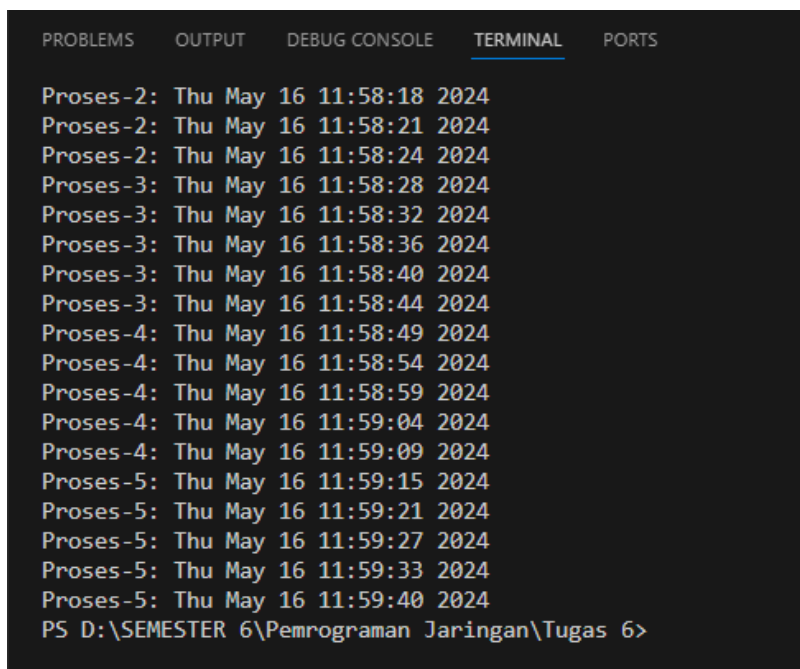
# Create proses with different delays
print_time("Proses-1", 2)
print_time("Proses-2", 3)
print_time("Proses-3", 4)
```

```
print_time("Proses-4", 5)
print_time("Proses-5", 6)
```

Penjelasan :

Code diatas digunakan untuk mendefinisikan fungsi `print_time` yang mencetak waktu saat ini bersama dengan nama proses (`proses`) setiap interval tertentu (`delay`). Fungsi ini dijalankan lima kali dengan `delay` yang berbeda untuk setiap proses. Script tersebut berjalan secara berurutan, sehingga masing-masing proses diselesaikan satu per satu, membuat total waktu eksekusi menjadi cukup lama. Untuk menjalankan proses secara bersamaan, dapat digunakan `threading` atau `multiprocessing`. Contoh penggunaan `threading` diberikan untuk mengurangi waktu eksekusi keseluruhan dengan menjalankan setiap proses secara paralel.

Hasil Output :



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Proses-2: Thu May 16 11:58:18 2024
Proses-2: Thu May 16 11:58:21 2024
Proses-2: Thu May 16 11:58:24 2024
Proses-3: Thu May 16 11:58:28 2024
Proses-3: Thu May 16 11:58:32 2024
Proses-3: Thu May 16 11:58:36 2024
Proses-3: Thu May 16 11:58:40 2024
Proses-3: Thu May 16 11:58:44 2024
Proses-4: Thu May 16 11:58:49 2024
Proses-4: Thu May 16 11:58:54 2024
Proses-4: Thu May 16 11:58:59 2024
Proses-4: Thu May 16 11:59:04 2024
Proses-4: Thu May 16 11:59:09 2024
Proses-5: Thu May 16 11:59:15 2024
Proses-5: Thu May 16 11:59:21 2024
Proses-5: Thu May 16 11:59:27 2024
Proses-5: Thu May 16 11:59:33 2024
Proses-5: Thu May 16 11:59:40 2024
PS D:\SEMESTER 6\Pemrograman Jaringan\Tugas 6>
```

2. Source Code :

```
import time
import threading

# Definisikan fungsi countdown
def countdown(name, start, lock):
    count = start
```

```

        while count > 0:
            with lock:
                print(f"{name}: {count}")
                time.sleep(1)
                count -= 1
        with lock:
            print(f"{name}: Selesai")

# Buat lock
print_lock = threading.Lock()

# Buat thread dengan countdown yang berbeda
threads = []
countdowns = [3, 6, 9, 12, 15]

for i in range(5):
    thread = threading.Thread(target=countdown,
    args=(f"Thread {i+1}", countdowns[i], print_lock))
    threads.append(thread)
    thread.start()

# Tunggu semua thread selesai
for t in threads:
    t.join()

```

Penjelasan :

Program Python ini membuat lima thread yang menjalankan hitungan mundur dengan delay 1 detik untuk setiap hitungan, dimulai dari nilai yang berbeda untuk setiap thread. Fungsi countdown yang didefinisikan menerima nama thread, nilai awal hitungan mundur, dan lock sebagai argumen. Setiap thread mencetak nilai hitungan mundurnya ke konsol dan menunggu 1 detik sebelum mengurangi hitungan, hingga mencapai nol dan mencetak pesan selesai. Untuk memastikan output dari setiap thread tidak tumpang tindih dan tetap rapi, digunakan mekanisme locking dengan `threading.Lock`, yang memastikan hanya satu thread yang dapat mencetak ke konsol pada satu waktu. Setelah semua thread dibuat dan dijalankan, program utama menunggu hingga semua thread selesai dengan memanggil `join` pada masing-masing thread. Mekanisme ini menghasilkan output yang teratur dan mudah dibaca.

Hasil Output :

```

\debugpy\launcher' '51971' '--' 'D:\SEMESTER 6\Pemrograman Jaringan\Tugas 6\No2.py'
Thread 1: 3
Thread 2: 6
Thread 3: 9
Thread 4: 12
Thread 5: 15
Thread 3: 8
Thread 1: 2
Thread 2: 5
Thread 5: 14
Thread 4: 11
Thread 2: 4
Thread 3: 7
Thread 1: 1
Thread 5: 13
Thread 4: 10
Thread 3: 6
Thread 1: Selesai
Thread 2: 3
Thread 4: 9
Thread 5: 12
Thread 2: 2
Thread 3: 5
Thread 5: 11
Thread 4: 8
Thread 3: 4
Thread 2: 1
Thread 4: 7
Thread 5: 10
Thread 3: 3
Thread 2: Selesai
Thread 5: 9
Thread 4: 6
Thread 3: 2
Thread 4: 5
Thread 5: 8
Thread 3: 1
Thread 5: 7
Thread 4: 4
Thread 3: Selesai
Thread 4: 3
Thread 5: 6
Thread 5: 5
Thread 4: 2
Thread 5: 4
Thread 4: 1
Thread 4: Selesai
Thread 5: 3
Thread 5: 2
Thread 5: 1
Thread 5: Selesai
PS D:\SEMESTER 6\Pemrograman Jaringan\Tugas 6>

```

3. Source Code :

```

import time
import threading

# Definisikan fungsi untuk mencetak bilangan ganjil
def print_odd_numbers(lock):
    for number in range(1, 11, 2): # Bilangan ganjil dari
1 hingga 10

```

```

        with lock:
            print(f"Thread 1 (Ganjil): {number}")
            time.sleep(1)

# Definisikan fungsi untuk mencetak bilangan genap
def print_even_numbers(lock):
    for number in range(2, 11, 2): # Bilangan genap dari 1
        hingga 10
        with lock:
            print(f"Thread 2 (Genap): {number}")
            time.sleep(1)

# Buat lock
print_lock = threading.Lock()

# Buat thread
thread_odd = threading.Thread(target=print_odd_numbers,
                               args=(print_lock,))
thread_even = threading.Thread(target=print_even_numbers,
                                args=(print_lock,))

# Mulai thread
thread_odd.start()
thread_even.start()

# Tunggu semua thread selesai
thread_odd.join()
thread_even.join()

```

Penjelasan :

Dalam kode tersebut, terdapat dua fungsi yang bertujuan mencetak bilangan ganjil dan genap secara bersamaan menggunakan threading di Python. Fungsi `print_odd_numbers` bertanggung jawab mencetak bilangan ganjil dari 1 hingga 10, sementara `print_even_numbers` mencetak bilangan genap dari 2 hingga 10. Kedua fungsi tersebut menggunakan threading untuk melakukan pencetakan secara paralel. Hal ini dilakukan dengan menginisialisasi dua thread terpisah untuk masing-masing fungsi dan menjalankannya secara bersamaan. Dengan menggunakan objek lock, kita memastikan bahwa kedua thread tidak mencetak output secara bersamaan dan mengakses sumber daya bersama (dalam hal ini, konsol) secara bergantian. Dengan demikian, kita mendapatkan output yang terorganisir di mana bilangan ganjil dan genap dicetak dengan label "Ganjil" dan "Genap" masing-masing.

Hasil Output :

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\SEMESTER 6\Pemrograman Jaringan\Tugas 6> & 'c:\Users\rozyyundled\libs\debugpy\adapter\..\..\debugpy\launcher' '52137' '--'
Thread 1 (Ganjil): 1
Thread 2 (Genap): 2
Thread 2 (Genap): 4
Thread 1 (Ganjil): 3
Thread 1 (Ganjil): 5
Thread 2 (Genap): 6
Thread 2 (Genap): 8
Thread 1 (Ganjil): 7
Thread 1 (Ganjil): 9
Thread 2 (Genap): 10
PS D:\SEMESTER 6\Pemrograman Jaringan\Tugas 6>
```