MAKALAH TUGAS BESAR

PEMROGRAMAN FUNGSIONAL

Nama Kelompok :

Zulvikar Harist	202110370311033
Salwa Nur Azizah	202110370311034
Ram Rizky Angkotasan	202110370311036

FAKULTAS TEKNIK

PROGRAM STUDI INFORMATIKA

UNIVERSITAS MUHAMMADIYAH MALANG

BAB I

PENDAHULUAN

1. Latar Belakang

Pada era digital seperti sekarang, pengembangan perangkat lunak telah menjadi bagian integral dari kehidupan sehari-hari. Salah satu aspek penting dalam pengembangan perangkat lunak adalah memahami konsep pemrograman fungsional. Pemrograman fungsional menekankan pada penggunaan fungsi dan ekspresi, yang dapat membantu dalam menciptakan kode yang lebih bersih, modular, dan mudah dimengerti**.**

Tujuan dari pengembangan kode di atas adalah untuk menciptakan program yang dapat mengelola data penonton harian di bioskop. Program ini menggunakan paradigma pemrograman fungsional dengan beberapa elemen penting seperti fungsi murni, ekspresi lambda, dan fungsi generator. Selain itu, program ini juga menunjukkan integrasi antara pemrograman fungsional dan GUI (Graphical User Interface) menggunakan modul tkinter.

Dengan adanya program ini, diharapkan pembaca dapat memahami konsep pemrograman fungsional, integrasi pemrograman fungsional dengan GUI, serta penerapan paradigma ini dalam pengembangan perangkat lunak.

BAB II

IMPLEMENTASI MATERI MODUL

2. Modul 1

2.1 List

Dalam bahasa pemrograman Python, struktur data yang paling dasar adalah sebuah daftar atau list. Setiap elemen-elemen dalam daftar disimpan secara berurutan dan diberi nomor posisi atau indeks seperti halnya array. Indeks pertama dalam list adalah nol, indeks kedua adalah satu dan seterusnya.

```
4   # Tipe Data Sequence
5   tanggal = []
6   penonton = []
7
```

Dalam kode dibawah, tipe data sequence merujuk pada penggunaan dua list: **tanggal** dan **penonton**. Dua list ini digunakan untuk menyimpan data penonton harian, dengan tanggal berisi tanggal kejadian dan penonton berisi jumlah penonton pada tanggal tersebut.

- tanggal: Variabel ini merupakan list yang ditujukan untuk menyimpan tanggal. Setiap elemen dalam list ini akan mewakili sebuah tanggal. List ini dapat diisi dengan nilai-nilai tanggal, dan karena bersifat mutable (dapat diubah), kita dapat menambahkan, menghapus, atau memodifikasi elemen-elemen di dalamnya.
- penonton: Variabel ini juga merupakan list, tetapi ditujukan untuk menyimpan jumlah penonton. Setiap elemen dalam list ini akan mewakili jumlah penonton pada suatu tanggal tertentu. Seperti halnya tanggal, penonton juga bersifat mutable.

3. Modul 2

3.1 Pure Function

Pure Function adalah suatu fungsi yang, ketika dipanggil dengan input yang sama, selalu mengembalikan output yang sama tanpa memiliki efek samping yang dapat dirasakan di luar fungsi tersebut. Artinya, hasil dari pemanggilan fungsi hanya bergantung pada parameter yang diberikan ke fungsi dan tidak bergantung pada

variabel atau kondisi lain di lingkungan eksternal.efek samping dan menghasilkan output hanya berdasarkan input yang diberikan.

a) tambah_data

Fungsi ini dapat dianggap sebagai pure function karena tidak bergantung pada variabel atau state di luar parameter yang diterimanya. Input fungsi hanya terdiri dari parameter tgl, jml_penonton, data_tanggal, dan data_penonton, dan outputnya adalah modifikasi pada list data_tanggal dan data_penonton. Fungsi ini tidak melakukan perubahan atau manipulasi pada variabel di luar parameter.

```
8   # Pure Function
9   def tambah_data(tgl, jml_penonton, data_tanggal, data_penonton):
10      # Menambahkan data penonton harian ke list tanggal dan penonton
11      data_tanggal.append(tgl)
12      data_penonton.append(jml_penonton)
```

3.2 Lambda Expression

Lambda adalah cara untuk membuat fungsi anonim (tanpa nama) dalam bahasa pemrograman Python. Fungsi lambda diciptakan menggunakan kata kunci lambda diikuti oleh parameter, titik dua (:), dan ekspresi yang menjadi nilai kembalian dari fungsi tersebut.

a) hitung_rata_rata

Lambda expression ini digunakan untuk membuat fungsi yang menghitung rata-rata dari suatu data. Fungsi ini akan digunakan nanti dalam beberapa bagian program, terutama dalam list comprehension dan fungsi lainnya.

```
14  # Lambda Expression
15  hitung_rata_rata = lambda data: sum(data) / len(data) if len(data) > 0 else 0
16  # Lambda untuk menghitung rata-rata penonton, dengan perlakuan khusus jika data kosong
17
```

3.3 List Comprehension

List comprehension adalah cara ringkas dan ekspresif dalam membuat list selain tu, list comprehension juga memudahkan kita untuk membuat list baru dengan sintaks yang lebih singkat dan mudah dibaca daripada menggunakan pendekatan tradisional dengan penggunaan loop.

a) penonton_diatas_rata

Bagian ini membuat sebuah list baru (**penonton_diatas_rata**) yang berisi nilai-nilai penonton yang lebih besar dari rata-rata penonton. Ekspresi [pen for pen in penonton if pen > hitung_rata_rata(penonton)] membentuk list baru dengan mengiterasi setiap nilai pen dalam list penonton, kemudian menyaring hanya nilai-nilai yang lebih besar dari rata-rata penonton.

```
58  # List Comprehension
59  penonton_diatas_rata = [pen for pen in penonton if pen > hitung_rata_rata(penonton)]
60  # List comprehension untuk mendapatkan daftar penonton di atas rata-rata
```

3.4 Generator

Generator adalah suatu konsep dalam pemrograman yang mengacu pada fungsi yang menghasilkan serangkaian nilai secara dinamis dan tidak menyimpan seluruh serangkaian nilai tersebut di dalam memori. Generator menggunakan kata kunci yield untuk mengembalikan nilai satu per satu kepada pemanggilnya.

a) data_generator

data_generator adalah suatu fungsi generator yang menggunakan kata kunci yield. Fungsi ini digunakan untuk menghasilkan pasangan nilai (tanggal, penonton) berdasarkan data yang dimasukkan.

```
18  # Generator
19  def data_generator(data_tanggal, data_penonton):
20      # Fungsi generator untuk menghasilkan pasangan (tanggal, penonton)
21      for tgl, pen in zip(data_tanggal, data_penonton):
22          yield tgl, pen
23
```

4. Modul 3

4.1 High Order Function

High-order function adalah konsep dalam pemrograman fungsional di mana fungsi dapat menerima fungsi lain sebagai argumen atau mengembalikan fungsi sebagai hasil. Dengan kata lain, dalam high-order function, fungsi dianggap sebagai nilai yang dapat digunakan dan dioperasikan seperti halnya tipe data lainnya.

a) filter_data

Fungsi **filter_data** adalah contoh dari higher-order function karena mengembalikan fungsi lain (**inner_filter**). Higher-order function ini menerima satu argumen (**minimal_penonton**) dan mengembalikan sebuah fungsi (**inner_filter**). Fungsi yang dikembalikan ini melakukan filtering terhadap data penonton berdasarkan nilai **minimal_penonton**.

```
24  # Higher Order Function
25  def filter_data(minimal_penonton):
26      # Fungsi higher-order untuk melakukan filtering berdasarkan minimal_penonton
27      def inner_filter(data_tanggal, data_penonton):
28          return [(tgl, pen) for tgl, pen in zip(data_tanggal, data_penonton) if pen >= minimal_penonton]
29      return inner_filter
```

5. Modul 4

5.1 Inner Function

Inner function (fungsi dalam) merujuk pada fungsi yang didefinisikan di dalam fungsi lainnya. Dalam bahasa pemrograman Python, inner function bisa dibuat di dalam tubuh fungsi yang lain. Inner function dapat mengakses variabel yang didefinisikan di dalam fungsi induknya, dan ini menciptakan lingkup (scope) tertutup yang disebut juga dengan closure.

a) rata_rata_penonton

Inner function adalah suatu fungsi yang didefinisikan di dalam fungsi lainnya. Pada kode yang diberikan, terdapat dua contoh inner function, yaitu **inner_filter** dan **inner_rata_rata**.

```
31  # Inner Function
32  def rata_rata_penonton(data_penonton):
33      # Fungsi inner untuk mengembalikan fungsi rata-rata tanpa parameter
34      def inner_rata_rata():
35          return hitung_rata_rata(data_penonton)
36      return inner_rata_rata
```

5.2 Decorator

Decorator adalah konsep dalam pemrograman Python yang memungkinkan Anda mengubah atau memperluas perilaku fungsi atau metode tanpa mengubah kode sumber asli. Dengan kata lain, decorator memungkinkan Anda menambahkan fungsionalitas tambahan ke fungsi tanpa mengubah implementasi inti dari fungsi tersebut.

Decorator biasanya diterapkan menggunakan simbol "@" diikuti oleh nama fungsi decorator. Fungsi decorator adalah fungsi tingkat tinggi yang mengambil fungsi lain sebagai argumen dan mengembalikan fungsi baru dengan perubahan atau penambahan fungsionalitas.

a) log_kegiatan & @log_kegiatan

log_kegiatan adalah fungsi decorator yang mengambil fungsi lain (func) sebagai argumen. Fungsi wrapper di dalam decorator mencetak pesan log sebelum dan setelah fungsi yang di-decorate (func) dipanggil. Fungsi wrapper mengembalikan hasil dari pemanggilan fungsi yang di-decorate. Decorator diaplikasikan pada fungsi tampilkan_grafik dengan menggunakan dekorator @log_kegiatan sebelum deklarasi fungsi tersebut.

```
38  # Decorator
39  def log_kegiatan(func):
40      # Decorator untuk mencetak pesan log setiap kali fungsi dipanggil
41      def wrapper(*args, **kwargs):
42          result = func(*args, **kwargs)
43          print(f"Kegiatan selesai: {func.__name__}")
44          return result
45      return wrapper
46
47  @log_kegiatan
48  def tampilkan_grafik(data_tanggal, data_penonton):
49      # Fungsi untuk menampilkan grafik menggunakan Matplotlib
50      plt.plot(data_tanggal, data_penonton, marker='o')
51      plt.title("Grafik Minat Menonton Bioskop")
52      plt.xlabel("Tanggal")
53      plt.ylabel("Jumlah Penonton")
54      plt.xticks(rotation=45)
55      plt.grid(True)
56      plt.show()
```

6. Modul 5

6.1 Library

Sebuah library (pustaka atau pustaka program) adalah kumpulan dari rutin, fungsi, dan alat bantu pemrograman yang telah dikompilasi dan diorganisir agar dapat digunakan oleh program-program komputer. Tujuan utama dari library adalah untuk menyediakan fungsionalitas yang umum digunakan agar tidak perlu dibuat ulang oleh setiap pengembang program.

Library yang digunakan, sebagai berikut :

a) Matplotlib

Library untuk membuat visualisasi grafik atau plot dalam bahasa pemrograman Python. Library ini menyediakan berbagai macam fungsi dan alat untuk membuat grafik, plot, diagram batang, diagram lingkaran, dan visualisasi data lainnya.

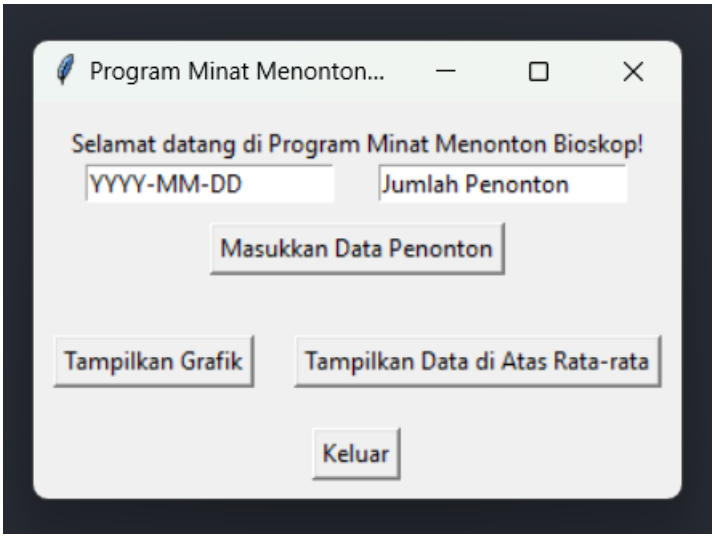
b) Tkinter

Toolkit GUI (Graphical User Interface) standar untuk Python. Library ini menyediakan berbagai widget dan alat untuk membuat antarmuka pengguna grafis, seperti jendela, tombol, kotak teks, dan sebagainya.

```
1   import matplotlib.pyplot as plt
2   from tkinter import Tk, Label, Button, Entry, StringVar, messagebox, Frame
3
```

6.2 GUI

GUI (Graphical User Interface) adalah antarmuka pengguna grafis yang memungkinkan pengguna berinteraksi dengan program komputer menggunakan elemen visual seperti ikon, tombol, dan jendela, alih-alih hanya melalui teks atau baris perintah. GUI dirancang untuk membuat pengalaman pengguna lebih intuitif dan lebih mudah digunakan.



BAB III

PENUTUP

KESIMPULAN

Dalam pengembangan program ini, kami mengimplementasikan konsep pemrograman fungsional dengan menggunakan bahasa pemrograman Python. Program ini memberikan pengguna kemampuan untuk mencatat dan menganalisis data penonton harian di bioskop melalui antarmuka grafis pengguna (GUI). Konsep pemrograman fungsional diaplikasikan dengan menggunakan fungsi murni, ekspresi lambda, generator, higher-order function, dan decorator.

Penerapan pemrograman fungsional pada program ini memberikan kejelasan dalam struktur kode, memisahkan logika program menjadi fungsi-fungsi yang terdefinisi dengan baik. Penggunaan modul matplotlib untuk menampilkan grafik dan tkinter untuk GUI menunjukkan integrasi antara pemrograman fungsional dengan pengembangan aplikasi berbasis GUI.

Dengan demikian, program ini tidak hanya memberikan fungsionalitas yang berguna untuk mencatat data penonton, tetapi juga menunjukkan cara konsep-konsep pemrograman fungsional dapat diterapkan secara efektif dalam pengembangan perangkat lunak.