

**LAPORAN PROYEK AKHIR PRAKTIKUM  
MATA KULIAH ALGORITMA DAN PEMROGRAMAN DASAR**



**LIST DAN REVIEW ANIME**

**Oleh:**

**Kelompok 5**

<b>RAHMAT RIYADI (KETUA)</b>	<b>2409106074</b>
<b>IKHWAN HARIYANTO</b>	<b>2409106082</b>
<b>RAYHAN SETIAWAN</b>	<b>2409106083</b>

**PROGRAM STUDI INFORMATIKA  
FAKULTAS TEKNIK  
UNIVERSITAS MULAWARMAN  
SAMARINDA 2024**

## KATA PENGANTAR

Puji syukur kehadirat Allah Subhanahu wa ta'ala atas limpahan rahmat dan karunia-Nya, sehingga saya dapat menyelesaikan laporan ini dengan baik. Laporan ini disusun sebagai salah satu syarat untuk memenuhi tugas akhir mata kuliah algoritma pemrograman dasar. Penyusunan laporan ini bertujuan untuk mendokumentasikan dan menjelaskan proses pengembangan program **list dan review anime**, serta menganalisis hasil dan manfaat dari implementasinya.

Program list dan review anime yang dibahas dalam laporan ini merupakan sebuah program yang dirancang untuk **memudahkan para penggemar anime dalam mencari informasi dan berdiskusi mengenai anime favorit mereka**. Program ini memiliki fitur-fitur utama, antara lain:

- **Mesin pencari anime:** Memungkinkan pengguna untuk mencari anime berdasarkan judul, tahun rilis, dan genre.
- **Pengurutan anime:** Menampilkan daftar anime berdasarkan abjad atau tahun rilis.
- **Top 20 anime:** Menampilkan daftar 20 anime terpopuler.
- **Kuis pengetahuan anime:** Menguji pengetahuan pengguna tentang anime.
- **Forum diskusi:** Memfasilitasi pengguna untuk berdiskusi dan bertukar informasi tentang anime.

Diharapkan, program ini dapat memberikan manfaat sebagai berikut:

- **Sumber informasi:** Menyediakan data judul anime, tahun rilis, genre, dan jumlah episode.
- **Media diskusi:** Menjadi wadah bagi para penggemar anime untuk berinteraksi dan bertukar pendapat.
- **Hiburan:** Menyediakan kuis pengetahuan anime yang menghibur.
- **Kemudahan akses:** Memudahkan pengguna dalam mencari informasi anime dengan cepat dan mudah.

Dalam proses penyusunan program dan laporan ini, saya menghadapi beberapa kendala, antara lain kendala pememilih judul, pembagian tugas dan program didalamnya. Namun, berkat dukungan dan bantuan dari berbagai pihak, kendala-kendala tersebut dapat diatasi dengan baik. Oleh karena itu, saya ingin mengucapkan terima kasih kepada bang Muhammad Ghazali selaku Aslep pembimbing serta teman-teman anggota kelompok 5 yang telah memberikan dukungan dan motivasi.

## **TAKARIR**

Daftar padanan kata bahasa asing dalam bahasa Indonesia yang digunakan adalah sebagai berikut:

<i>Database</i>	Basis Data
<i>Managemen</i>	Mengatur
<i>Input</i>	Memasukkan

## DAFTAR ISI

KATA PENGANTAR.....	2
TAKARIR .....	4
DAFTAR ISI.....	5
DAFTAR GAMBAR.....	6
BAB I PENDAHULUAN .....	1
1.1 Latar Belakang.....	1
1.2 Kebutuhan Fungsional .....	1
1.3 Rumusan Masalah .....	1
1.4 Batasan Masalah .....	1
1.5 Tujuan .....	1
BAB II PERANCANGAN .....	2
2.1 Analisis Program.....	2
2.2 Flowchart .....	2
2.3 Konsep/Materi Praktikum yang dipakai.....	2
BAB III HASIL DAN PEMBAHASAN .....	3
3.1 Tampilan Program .....	3
BAB IV PENUTUP .....	23
4.1 Kesimpulan.....	23
4.2 Saran .....	23
DAFTAR PUSTAKA.....	24
LAMPIRAN .....	<b>Error! Bookmark not defined.</b>

## **DAFTAR GAMBAR**

Gambar 1 Tampilan awal program.

8

# **BAB I**

## **PENDAHULUAN**

### **1.1 Latar Belakang**

Menjelaskan latar belakang masalah dan rincian masalah secara umum yang menyebabkan dibuatnya program.

### **1.2 Kebutuhan Fungsional**

Semua kebutuhan fungsional program.

### **1.3 Rumusan Masalah**

Menjelaskan masalah yang diambil sebagai tema.

### **1.4 Batasan Masalah**

Menjelaskan batasan-batasan dari masalah yang diambil sehingga pembahasan tidak meluas.

### **1.5 Tujuan**

Tujuan yang akan dicapai setelah ditemukannya masalah.

## **BAB II**

### **PERANCANGAN**

#### **2.1 Analisis Program**

Penjelasan singkat jalan program, penjelasan alur program.

#### **2.2 Flowchart**

Lampirkan flowchart dan jelaskan alurnya.

#### **2.3 Konsep/Materi Praktikum yang dipakai**

Konsep yang kami gunakan dalam pembuatan laporan ini sesuai dengan persyaratan yang terdapat pada modul diantaranya:

1. Fungsi dasar: fungsi yang kami gunakan adalah fungsi print.

Fungsi print pada program merupakan fungsi yang umum dipakai untuk menampilkan suatu keluaran pada layar peraga.



## BAB III

### HASIL DAN PEMBAHASAN

#### 3.1 Tampilan Program

```
66 ''' FUNGSI DAFTAR DAN LOGIN '''
67 # Menyimpan data pengguna baru ke Database
68 def daftar():
69     os.system("cls || clear")
70     print("Membuat Akun")
71     usn = input("Masukkan Username: ")
72     pas = input("Masukkan Password: ")
73     try:
74         with open(file_pengguna, "r") as file:
75             users = json.load(file)
76     except (FileNotFoundError, json.JSONDecodeError):
77         users = {}
78     users[usn] = pas
79     with open(file_pengguna, "w") as file:
80         json.dump(users, file, indent=4)
81     print("Pendaftaran Berhasil\n")
82
```

- **Deskripsi program**

Program ini adalah semua fungsi Python yang memungkinkan pengguna untuk membuat akun baru dengan memasukkan username dan password. Data yang dimasukkan oleh pengguna disimpan dalam file JSON, berfungsi sebagai basis data sederhana untuk menyimpan informasi pengguna.

- **Tujuan program**

- Memungkinkan pengguna untuk mendaftarkan akun baru.
- Menyimpan data username dan password pengguna ke dalam file JSON.
- Memberikan solusi sederhana untuk manajemen akun dalam skala kecil

- **Cara kerja program**

1. Program memulai dengan membersihkan layar terminal untuk menampilkan tampilan yang bersih menggunakan perintah:  
`os.system("cls || clear")`
  - **cls** digunakan untuk sistem Windows.
  - **clear** digunakan untuk sistem berbasis UNIX (Linux/Mac).
2. Program meminta pengguna memasukkan **username** dan **password**:
  - `usn = input("Masukkan Username: ")`
  - `pas = input("Masukkan Password: ")`
3. Program mencoba membaca data pengguna yang tersimpan di file JSON:  
`with open(file_pengguna, "r") as file:`  
`users = json.load(file)`

- Jika file tidak ditemukan (**FileNotFoundError**) atau file kosong/tidak valid (**json.JSONDecodeError**), program akan membuat dictionary kosong (`users = {}`).
4. Program menambahkan pasangan username-password ke dictionary users:
    - `users[usn] = pas`
  5. Program menyimpan kembali data pengguna ke file JSON:
    - `with open(file_pengguna, "w") as file:`  
`json.dump(users, file, indent=4)`
  6. Menampilkan pesan keberhasilan:
    - `print("Pendaftaran Berhasil\n")`

```

83 # Mengecek data pengguna di database
84 def user_login():
85     global kesempatan_login, usn_komen
86     os.system("cls || clear")
87     print(batas1)
88     print("Silahkan login terlebih dahulu!\n".center(70))
89     print(batas1, "\n")
90     print(batas_garis)
91     usn = input("Username: ")
92     pas = input("Password: ")
93     print(batas_garis)
94     try:
95         with open(file_pengguna, "r") as file:
96             users = json.load(file)
97     except (FileNotFoundError, json.JSONDecodeError):
98         users = {}
99     if usn == "admin" and pas == "admin":
100         print()
101         print(batas2)
102         print("SELAMAT SEPULUH BERHASIL LOGIN".center(70))
103         print(batas2, "\n")
104         return "ADMIN"
105     elif usn in users and users[usn] == pas:
106         print()
107         print(batas2)
108         print("ANDA BERHASIL LOGIN".center(70))
109         print(batas2, "\n")
110         usn_komen = usn
111         return "USER"
112     else:
113         print("Login Gagal Silahkan login ulang")
114         input("Tekan enter untuk melakukan login...")
115         return user_login()

```

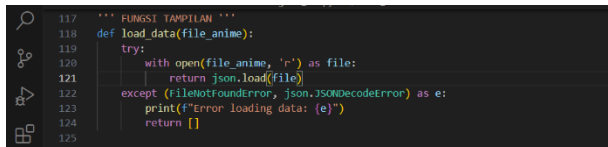
## 1. Deskripsi Program

Program ini adalah fungsi Python bernama `user_login` yang digunakan untuk memverifikasi kredensial pengguna dari file JSON. Program ini memungkinkan pengguna untuk login dengan username dan password mereka. Jika berhasil, program memberikan akses sesuai dengan peran pengguna, yaitu “ADMIN” atau “USER”.

## 2. Cara Kerja Program

- Tampil Awal:
  - Membersihkan layer terminal dengan:  
`os.system("cls || clear")`

- Menampilkan pesan dan garis betas menggunakan variabel seperti **batas1**, **batas2**, dan **batas\_garis** untuk memberikan tampilan yang lebih rapi.
- **Input Login:**
  - Meminta pengguna memasukkan **username** (user) dan **password** (pas) melalui **input**.
- **Membaca Data Pengguna:**
  - Program mencoba membaca file JSON yang menyimpan data pengguna:  
with open(file\_pengguna, "r") as file:  
    users = json.load(file)
  - Jika file tidak ditemukan atau isinya tidak valid, program akan menginisialisasi dictionary kosong (**users = {}**).
- **Validasi Admin:**
  - Jika username dan password yang dimasukkan adalah **admin**, pengguna akan diidentifikasi sebagai “ADMIN”.
- **Validasi User:**
  - Jika username yang dimasukkan ada di database ( usn in users ) dan password sesuai ( users [ usn ] == pas ), pengguna akan diidentifikasi sebagai “USER”.
- **Gagal Login:**
  - Jika username atau password salah, program salah, program akan menampilkan pesan gagal login dan memanggil kembali fungsi **user\_login()** secara rekursif, memungkinkan pengguna untuk mencoba login ulang.
- **Hasil Login.**
  - Program mengembalikan hasil login:
    - “ADMIN” jika pengguna adalah admin.
    - “USER” jika login berhasil sebagai user.
  - Variabel global usn\_komen digunakan untuk menyimpan username pengguna yang berhasil login.



```

117 ''' FUNGSI TAMPILAN '''
118 def load_data(file_anime):
119     try:
120         with open(file_anime, 'r') as file:
121             return json.load(file)
122     except (FileNotFoundError, json.JSONDecodeError) as e:
123         print(f"Error loading data: {e}")
124         return []
125

```

## 1. Deskripsi Program

Program ini adalah sebuah fungsi Python bernama `load_data`, yang dirancang untuk membaca dan memuat data dari file JSON. Data yang dimuat kemudian dapat digunakan dalam program untuk keperluan lain, seperti menampilkan informasi, memproses data, atau menyimpannya ke dalam variabel.

Fungsi ini juga memiliki mekanisme penanganan error untuk menghadapi kasus di mana file tidak ditemukan atau data di dalam file JSON tidak valid.

## 2. Tujuan Program

- Membaca file JSON yang berisi data tertentu (dalam contoh ini, data anime).
- Mengembalikan data yang berhasil dimuat dalam bentuk python object (list/dictionary).
- Menangani error dengan memberikan pesan yang jelas dan mengembalikan nilai default jika terjadi kesalahan.

## 3. Cara kerja Program

### 1. Penerimaan Argumen:

- Fungsi menerima satu argument `file_anime`, yang merupakan nama atau path file JSON yang akan dibaca.

### 2. Membaca File JSON:

- Fungsi mencoba membuka file dengan mode baca ( `'r'` ):
 

```
with open(file_anime, 'r') as file:
    return json.load(file)
```
- Data di dalam file JSON diubah menjadi object python menggunakan **`json.load`**

### 3. Penanganan Error:

- Jika file tidak ditemukan ( `FileNotFoundError` ) atau data di dalam file JSON tidak valid ( `JSONDecodeError` ), fungsi:
  - Mencetak pesan error:
 

```
print(f"Error loading data: {e}")
```
  - Mengembalikan nilai default berupa list kosong ( `[]` ).

#### 4. Hasil:

- Jika file JSON berhasil dibaca, fungsi mengembalikan data dalam bentuk object python.
- Jika terjadi error, fungsi mengembalikan list kosong.

```
126 # Fungsi untuk menyimpan data ke file JSON
127 def simpan_data(file_anime, data):
128     try:
129         with open(file_anime, 'w') as file:
130             json.dump(data, file, indent=4)
131     except IOError as e:
132         print("Error saving data: {e}")
133
```

### 1. Deskripsi Program

Program ini adalah sebuah fungsi python bernama `load_data`, yang dirancang untuk membaca dan memuat data dari file JSON. Data yang dimuat kemudian dapat digunakan dalam program untuk keperluan lain, seperti menampilkan informasi, memproses data, atau menyimpannya ke dalam variabel.

Fungsi ini juga memiliki mekanisme penanganan error untuk menghadapi kasus di mana file tidak ditemukan atau data di dalam file JSON tidak valid.

### 2. Tujuan Program

- Membaca file JSON yang berisi data tertentu (dalam contoh ini, data anime)
- Mengembalikan data yang berhasil dimuat dalam bentuk Python object (list/dictionary)
- Menangani error dengan memberikan pesan yang jelas dan mengembalikan nilai default jika terjadi kesalahan.

### 3. Cara Kerja Program

- Definisi Fungsi:  

```
def simpan_data(file_anime, data):
```

  - `File_anime` : Parameter ini adalah nama file (beserta path-nya jika diperlukan) di mana data akan disimpan.
  - `Data`: Parameter ini adalah data yang akan disimpan ke dalam file. Data tersebut biasanya berupa tipe data python yang kompatibel dengan JSON, seperti dictionary atau list.
- Blok try  

```
try:
```

```
    with open(file_anime, 'w') as file:
```

```
json.dump(data, file, indent=4)
```

- With open(file\_anime, 'w') as file: Membuka file dengan mode tulis ('w'). Jika file sudah ada, konten lama akan di timpa. Jika file belum ada, file baru akan dibuat.
  - Json.dump(data, file, indent=4); Mengubah data python menjadi format JSON dan menyimpannya ke dalam file. Parameter indent=4 digunakan agar data JSON yang disimpan memiliki format yang lebih mudah dibaca (indented).
- Blok except  
except IOError as e:  
print(f"Error saving data: {e}")
    - Except IOError as e :Menangkap kesalahan yang terjadi saat operasi input/output (I/O), seperti jika file tidak dapat dibuka, atau ada masalah saat menyimpan data.
    - Print(f"Error saving data: {e}") : Menampilkan pesan kesalahan yang menjekaskan apa yang terjadi.

```
133  
134 # Fungsi untuk menghapus duplikasi data  
135 def hapus_duplikat(anime_list):  
136     judul_unik = set()  
137     anime_unik = []  
138     for anime in anime_list:  
139         if anime['title'] not in judul_unik:  
140             judul_unik.add(anime['title'])  
141             anime_unik.append(anime)  
142     return anime_unik  
143
```

## 1. Deskripsi Program

Fungsi hapus\_duplikat bertujuan untuk menghapus elemen duplikat dalam sebuah daftar (list) berdasarkan nilai unik dari atribut tertentu dalam setiap elemen. Dalam kasus ini, elemen berupa dictionary yang memiliki atribut 'title', dan fungsi memastikan hanya data dengan judul unik yang dimasukkan ke dalam daftar hasil.

## 2. Penjelasan Kode

### 1. Parameter Fungsi

```
def hapus_duplikat(anime_list):
```

- anime\_list : Parameter berupa daftar (list) yang berisi data anime. Setiap elemen dalam daftar adalah dictionary dengan key-value, salah satunya adalah key 'title'.

### 2. Inisialisai Variabel

```
    judul_unik = set()
```

`anime_unik =`

- `Judul_unik` : Sebuah set yang digunakan untuk menyimpan judul-judul anime yang sudah ditemukan. Set dipilih karena secara efisien dapat memastikan elemen di dalamnya bersifat unik.
- `anime_unik` : Sebuah list yang digunakan untuk menyimpan elemen-elemen dari daftar input yang memiliki judul unik.

### 3. Iterasi pada Daftar

`for anime in anime_list:`

- Fungsi melakukan iterasi (perulangan) pada setiap elemen dalam daftar `anime_list`.
- Setiap elemen dalam daftar diasumsikan adalah sebuah dictionary yang memiliki key 'title'.

### 4. Pengecekan Unik:

`if anime['title'] not in judul_unik:`

- Fungsi memeriksa apakah judul anime ( `anime['title']` ) belum ada set `judul_unik`
- Jika judul tersebut unik:
  1. Tambahkan judul ke dalam set:  
`judul_unik.add(anime['title'])`
  2. Masukkan elemen dictionary ke dalam daftar hasil:  
`anime_unik.append(anime)`

### 5. Mengembalikan Hasil

`return anime_unik`

- Setelah semua elemen diperiksa, fungsi mengembalikan daftar **`anime_unik`**, yang berisi elemen-elemen dari daftar asli tetapi hanya dengan judul yang unik.

```
143
144 # Fungsi untuk mencari anime berdasarkan judul
145 def cari_dengan_judul(anime_list, title):
146     anime_unik_list = hapus_duplikat(anime_list)
147     results = []
148     for anime in anime_unik_list:
149         if title.lower() in anime["title"].lower():
150             results.append(anime)
151     return results
```

## 1. Deskripsi Program:

Fungsi `cari_dengan_judul` digunakan untuk mencari anime dalam sebuah daftar berdasarkan judul tertentu. Pencarian bersifat tidak case-sensitive (tidak membedakan huruf besar atau kecil). Sebelum mencari, fungsi juga memastikan

daftar yang digunakan hanya berisi elemen dengan judul unik, menggunakan fungsi `hapus_duplikat`.

## 2. Penjelasan Kode

### 1. Parameter Fungsi

Fungsi menerima dua parameter:

- `anime_list`: Sebuah daftar (**list**) yang berisi elemen-elemen berupa dictionary, di mana setiap dictionary memiliki key `'title'`
- `title`: Sebuah string yang menjadi kata kunci pencarian. Fungsi mencari elemen dalam daftar yang judulnya mengandung string ini.

### 2. Menghapus Duplikat

```
anime_unik_list = hapus_duplikat(anime_list)
```

- Fungsi `hapus_duplikat` dipanggil untuk memastikan daftar input hanya berisi elemen dengan judul unik.
- `anime_unik_list`: Daftar hasil dari fungsi `hapus_duplikat`, yang berisi elemen-elemen dengan judul unik.

### 3. Inisialisasi Daftar Hasil

```
results = []
```

- Sebuah daftar kosong `results` dibuat untuk menyimpan hasil pencarian, yaitu elemen-elemen yang judulnya mengandung string pencarian (`title`).

### 4. Iterasi pada Daftar Unik

```
for anime in anime_unik_list:
```

- Program melakukan iterasi pada setiap elemen dalam daftar `anime_unik_list`.

### 5. Pencobaan Judul

```
if title.lower() in anime["title"].lower():  
    results.append(anime)
```

- `title.lower()`: String pencarian dikonversi menjadi huruf kecil untuk menghindari masalah perbedaan huruf besar/kecil
- `anime["title"].lower()`: Judul anime dalam elemen dictionary juga dikonversi menjadi huruf kecil.
- Kondisi `if` memeriksa apakah string pencarian `title` ada di dalam judul anime.



- Jika kondisi terpenuhi, elemen tersebut ditambahkan ke daftar hasil `results`.

## 6. Mengembalikan Hasil

`return results`

- Setelah semua elemen diperiksa, fungsi mengembalikan daftar `results`, yang berisi elemen-elemen dengan judul yang sesuai dengan string pencarian.

```

153 # Fungsi untuk mencetak judul anime berdasarkan tahun
154 def tampil_urut_tahun(year):
155     try:
156         with open('data_anime.json', 'r') as file:
157             anime_data = json.load(file)
158             print(f'Daftar judul anime yang dirilis pada tahun {year}:')
159             anime_unik_list = hapus_duplikat(anime_data)
160             tahun = False
161             for anime in anime_unik_list:
162                 if anime['year'] == year:
163                     print(f'- {anime["title"]}')
164                     tahun = True
165             if not tahun:
166                 print("Tidak ada anime yang ditemukan pada tahun tersebut.")
167     except (FileNotFoundError, json.JSONDecodeError) as e:
168         print(f'Error loading anime data: {e}')

```

### 1. Deskripsi Program

Program ini bertujuan untuk menampilkan daftar judul anime yang dirilis pada tahun tertentu berdasarkan data yang tersimpan dalam file `data_anime.json`. Data anime tersebut diorganisasikan dalam format JSON, yang umumnya berupa array berisi objek-objek dengan informasi seperti judul (`title`) dan tahun rilis (`year`).

### 2. Fungsi program

#### 1. Memuat Data dari File JSON:

- Program membuka file `data_anime.json` menggunakan fungsi `open()` dan membaca kontennya.
- Data JSON kemudian di-parse menjadi objek python menggunakan `json.load()`.

#### 2. Menghapus Duplikat:

- Program menggunakan fungsi `hapus_duplikat()` untuk memastikan bahwa data anime tidak mengandung entri yang sama (duplikat).

#### 3. Menampilkan Anime Berdasarkan Tahun:

- Program memeriksa tahun rilis (`year`) dari setiap entri di dalam data.
- Jika ada anime yang dirilis pada tahun yang diminta (`year`), judul anime tersebut ditampilkan.
- Jika tidak ada anime pada tahun tersebut, akan ditampilkan pesan bahwa tidak ada anime yang ditemukan.

#### 4. Penanganan Kesalahan:

- Program menangani dua jenis kesalahan umum:
  - **FileNotFoundError:** Jika file `data_anime.json` tidak ditemukan, program menampilkan pesan kesalahan.
  - **JSONDecodeError:** Jika file JSON rusak atau tidak valid, program juga menampilkan pesan kesalahan.

### 3. Penjelasan Blok Kode

#### 1. Fungsi `tampil_urut_tahun(year)`:

- Parameter `year` digunakan untuk menentukan tahun yang ingin dicari.
- Program menggunakan variabel `Tahun` untuk mencatat apakah ada anime yang ditemukan pada tahun yang diminta. Jika tidak ada, program memberikan pesan yang relevan.

#### 2. Fungsi `hapus_duplikat(anime_data)`:

- Walaupun tidak ada implementasi dalam kode yang diberikan, fungsi ini diasumsikan bertugas membersihkan entri duplikat dari data anime.
- Biasanya, fungsi ini mengembalikan daftar baru tanpa elemen yang sama.

#### 3. Error Handling:

- **Penangan kesalahan memastikan program tidak berhenti secara mendadak jika file tidak ditemukan atau datanya rusak.**

```
270 # Fungsi untuk mencetak judul anime berdasarkan genre dan jumlah episodenya
271 def cetak_berdasarkan_genre(anime_list, genre):
272     print(f"Daftar judul anime berdasarkan genre yang dipilih (contoh: {genre}):")
273     anime_unik_list = hapus_duplikat(anime_list)
274     for anime in anime_unik_list:
275         if genre in anime["genres"]:
276             print(f"- {anime['title']}: {anime['episodes']} episodes")
277
```

#### 1. Deskripsi Program

Program ini digunakan untuk mencetak daftar judul anime yang sesuai dengan genre tertentu berdasarkan data yang disediakan. Data anime berupa list berisi objek-objek yang memiliki informasi seperti judul (`title`), jumlah episode (`episodes`), dan genre (`genres`).

#### 2. Fungsi Program

##### 1. Menghapus Duplikat:

- Program memastikan tidak ada entri duplikat dalam data menggunakan fungsi `hapus_duplikat()`.

##### 2. Mencetak Anime Berdasarkan Genre:

- Program memeriksa apakah genre yang diminta ada dalam daftar genre (`genres`) untuk setiap anime.
- Jika genre cocok, program akan mencetak judul anime beserta jumlah episodenya.

### 3. **Pesan yang Informatif:**

- Program menampilkan informasi yang jelas, termasuk genre yang dipilih oleh pengguna.

## 3. Penjelasan Blok Kode

### 1. **Fungsi** `cetak_berdasarkan_gendre(anime_list, genre):`

#### • Parameter:

- `anime_list`: List data anime yang akan diperiksa.
- `genre`: Genre yang menjadi kriteria pencarian.

- Program mencetak daftar anime yang memiliki genre yang diminta.

### 2. **Penghapusan Duplikat dengan** `hapus_duplikat(anime_list):`

- Fungsi ini, meskipun belum diberikan implementasinya, diasumsikan bertugas membersihkan data dari entri yang sama. Hasilnya adalah daftar anime yang unik

### 3. **Pengecekan Genre:**

- Untuk setiap anime, program memeriksa apakah `genre` ada di dalam `anime["genres"]`.
- Jika genre cocok, judul dan jumlah episode anime akan ditampilkan dalam format yang rapi.

```

178 # Fungsi untuk mesin pencari
179 def mesin_pencari_judul():
180     anime_data = load_data("data_anime.json")
181     anime_data = hapus_duplikat(anime_data)
182     cari_judul = input("Masukkan judul anime yang ingin dicari: ").strip().lower()
183     hasil_pencarian = []
184     for anime in anime_data:
185         if cari_judul in anime["title"].lower():
186             hasil_pencarian.append(anime)
187     if hasil_pencarian:
188         for anime in hasil_pencarian:
189             print(f"Title: {anime['title']}")
190             print(f"Year: {anime['year']}")
191             print(f"Genres: {', '.join(anime['genres'])}")
192             print(f"Episodes: {anime['episodes']}")
193             print()
194     else:
195         print("Anime tidak ditemukan.")

```

## 1. Deskripsi Program

Program ini merupakan mesin pencari judul anime yang memungkinkan pengguna untuk mencari anime berdasarkan kata kunci yang di masukkan. Kata kunci akan

dibandingkan dengan judul anime dalam data, dan hasil pencarian yang sesuai akan di tampilkan.

## 2. Fungsi Program:

### 1. Membuat Data Anime:

- Program membaca data anime dari file JSON bernama `data_anime.json`.
- Data dihapus dari duplikat menggunakan fungsi `hapus_duplikat()`.

### 2. Menerima Input Pengguna:

- Pengguna diminta memasukkan kata kunci yang ingin dicari dalam judul anime.

### 3. Mencocokkan Judul:

- Program memeriksa apakah kata kunci yang dimasukkan oleh pengguna terdapat dalam judul anime, dengan proses pencarian yang tidak sensitif terhadap huruf besar atau kecil.

### 4. Menampilkan Hasil Pencarian:

- Jika ada hasil yang cocok, program menampilkan informasi berikut untuk setiap anime yang ditemukan:
  - Judul anime (`title`).
  - Tahun rilis (`year`).
  - Genre (`genres`).
  - Jumlah episode (`episodes`).
- Jika tidak ada hasil yang cocok, program memberi tahu pengguna bahwa anime tidak ditemukan.

## 3. Penjelasan Blok Kode

### 1. Fungsi `mesin_pencari_judul()`:

- Memuat Data:
  - Menggunakan fungsi `load_data()` untuk membaca data anime dari file `data_anime.json`. Data ini diasumsikan berbentuk list of dictionaries.
  - Data duplikat dihapus menggunakan `hapus_duplikat()` untuk memastikan hasil yang unik

- Input Pengguna:

- Menggunakan `input()` untuk meminta pengguna memasukkan kata kunci pencarian. Kata kunci ini diubah menjadi huruf kecil dengan `lower()` dan dihapus spasi tambahan dengan `strip()` untuk meningkatkan akurasi pencarian.
- Proses Pencarian:
  - Program memeriksa apakah kata kunci pengguna ada di dalam judul (`anime["title"]`), juga dalam huruf kecil.
  - Jika ditemukan kecocokan, anime tersebut ditambahkan ke list `hasil_pencarian`.
- Menampilkan Hasil;
  - Jika `hasil_pencarian` tidak kosong, informasi tentang setiap anime yang cocok ditampilkan.
  - Jika tidak ada kecocokan, ditampilkan pesan bahwa anime tidak ditemukan

```

197 # Fungsi untuk menampilkan anime berdasarkan abjad
198 def tampil_urut_abjad():
199     try:
200         with open('data_anime.json', 'r') as file:
201             data = json.load(file)
202             data = hapus_duplikat(data)
203             hasil = collections.defaultdict(list)
204             for anime in data:
205                 judul = anime['title']
206                 karakter_awal = judul[0].upper()
207                 hasil[karakter_awal].append(judul)
208             for key in hasil.keys():
209                 hasil[key] = sorted(hasil[key])
210             for key in sorted(hasil.keys()):
211                 print(key)
212                 for judul in hasil[key]:
213                     print(f"- {judul}")
214     except (FileNotFoundError, json.JSONDecodeError) as e:
215         print(f"Error loading anime data: {e}")

```

## 1. Deskripsi Program

Program `tampil_urut_abjad()` bertujuan untuk membaca dari sebuah file JSON, mengelompokkan data berdasarkan huruf pertama dari judul, lalu menampilkan hasilnya dalam urutan abjad. Berikut adalah penjelasan rinci tentang fungsi dan logika program:

## 2. Tujuan Program

- Membaca dan memproses data judul anime dari file JSON.
- Mengelompokkan judul berdasarkan huruf awal (A-Z).
- Menyusun daftar judul secara alfabetis di bawah setiap kelompok.
- Menghindari pengulangan data yang sama (duplikat).
- Menampilkan hasil dalam format yang mudah dibaca.

## 3. Cara Kerja Program

- Membuka dan Membaca File JSON

Program mencoba membuka file Bernama `data_anime.json`. Jika file tersebut tidak ditemukan atau data JSON-nya rusak, program akan menampilkan pesan kesalahan.

- Menghapus Duplikat

Dengan menggunakan fungsi `hapus_duplikat`, program memastikan bahwa setiap judul hanya muncul satu kali dalam daftar.

- Mengelompokkan Berdasarkan Huruf Awal

Program menggunakan huruf pertama dari setiap judul untuk mengelompokkan judul-judul tersebut ke dalam kelompok abjad (A-Z). Huruf pertama diubah menjadi huruf besar menggunakan `upper()` agar pengelompokannya konsisten.

- Menyusun Data

- Setiap kelompok huruf disusun secara alfabetis menggunakan `sorted()`.

- Kelompok huruf (A-Z) juga disusun secara alfabetis untuk tampilan yang terorganisir.

- Menampilkan Data

Judul-judul anime ditampilkan dalam format berikut:

A

- Attack on Titan

B

- Bleach

D

- Death Note

Setiap huruf (A, B, C, dst.) menjadi header, dengan daftar judul yang disusun dibawahnya.

- Penanganan Kesalahan

Jika terjadi masalah seperti:

- File JSON tidak ditemukan (**`FileNotFoundError`**).
- Format file JSON tidak valid (**`JSONDecodeError`**).

Program akan menampilkan pesan kesalahan yang relevan, misalnya:

```
Error loading anime data: [error message]
```

```

217 # Fungsi menampilkan pilihan genre
218 def daftar_pilihan_genre():
219     print("1) Action")
220     print("2) Adventure")
221     print("3) Comedy")
222     print("4) Drama")
223     print("5) Fantasy")
224     print("6) Horror")
225     print("7) Mystery")
226     print("8) Music")
227     print("9) Psychological")
228     print("10) Romance")
229     print("11) School")
230     print("12) Sci-Fi")
231     print("13) Shounen")
232     print("14) Slice of Life")
233     print("15) Sports")
234     print("16) Supernatural")
235

```

1. Deskripsi Program Program ini adalah sebuah fungsi sederhana yang digunakan untuk menampilkan daftar pilihan genre anime kepada pengguna. Fungsi ini mencetak 16 genre anime yang umum dengan nomor urut masing-masing, sehingga mempermudah pengguna untuk memilih genre sesuai preferensi mereka.
2. Tujuan Program
  - Memberikan daftar genre anime yang populer atau umum kepada pengguna.
  - Mempermudah pengguna untuk melihat dan memilih genre berdasarkan nomor urut.
  - Mengorganisasikan informasi genre dengan rapi dan terstruktur.
3. Cara Kerja Program
  1. Mencetak Daftar Genre  
Fungsi ini menggunakan `print()` untuk menampilkan daftar genre anime satu per satu. Setiap genre diberi nomor urut dari **1 hingga 16** untuk mempermudah identifikasi.
  2. Menjaga Urutan Genre  
Genre ditampilkan dalam urutan yang logis, mulai dari genre populer seperti **Action** dan **Adventure**, hingga genre yang lebih spesifik seperti **Slice of Life** dan **Supernatural**.
  3. Memformat Output  
Setiap baris output memiliki format:  
`[nomor]) [nama genre]`  
 Contoh:  
 1) Action  
 2) Adventure

```

236 # Fungsi menampilkan top 20 anime
237 def tampilkan_top_20_anime():
238     print("Top 20 Anime:")
239     for i in range(len(top_20_anime)):
240         judul = top_20_anime[i][0]
241         tahun = top_20_anime[i][1]
242         print(f"{i + 1}. {judul} ({tahun})")
243

```

## 1. Deskripsi Program

terbaik berdasarkan data yang disimpan dalam variabel `top_20_anime`. Setiap entri dalam daftar ini berisi informasi berupa judul anime dan tahun rilisnya. Program menyajikan daftar

Fungsi `tampilkan_top_20_anime()` bertujuan untuk menampilkan daftar 20 anime dalam format yang terurut, dengan penomoran dari 1 hingga 20

## 2. Tujuan Program

1. Menampilkan daftar 20 anime terbaik kepada pengguna.
2. Memberikan informasi tambahan berupa tahun rilis dari setiap anime.
3. Menyusun daftar dalam format yang rapi dan mudah dipahami.

## 3. Cara Kerja Program

### 1. Header untuk Daftar

Program mencetak header berupa teks:

```
Top 20 Anime:
```

untuk memberikan konteks kepada pengguna bahwa daftar berikutnya adalah daftar 20 anime terbaik.

### 2. Menggunakan Loop untuk Menampilkan Daftar

Program menggunakan loop `for` untuk mengakses setiap elemen dalam daftar

```
top_20_anime:
```

```
for i in range(len(top_20_anime)):
```

`range(len(top_20_anime))` memastikan loop berjalan sebanyak jumlah elemen dalam daftar.

`i` digunakan sebagai indeks untuk mengakses elemen pada posisi tertentu di `top_20_anime`.

### 3. Mengakses Informasi Judul dan Tahun



- Setiap elemen dalam `top_20_anime` diasumsikan berupa tuple atau daftar berisi dua elemen:

- `judul` (string): Nama anime
- `tahun` (integer atau string): Tahun rilis anime.

- Program memecah elemen menjadi variabel:

```
judul = top_20_anime[i][0]
tahun = top_20_anime[i][1]
```

#### 4. Menampilkan Daftar

- Judul dan tahun anime ditampilkan dalam format rapi

```
print(f"{i + 1}. {judul} ({tahun})")
```

- `i + 1` digunakan untuk memberikan nomor urut yang dimulai dari 1.

```
244 ''' FUNGSI CRUD System'''
245 # Fungsi untuk mengedit anime di dalam json
246 def update_anime(data):
247     anime_data = load_data("data_anime.json")
248     cari_judul = input("Masukkan judul anime yang ingin diperbarui: ")
249     hasil_pencarian = cari_dengan_judul(anime_data, cari_judul)
250     if hasil_pencarian:
251         print("\nHasil pencarian:")
252         for index, anime in enumerate(hasil_pencarian, start=1):
253             print(f"{index}. Title: {anime['title']}")
254             print(f"Year: {anime['year']}")
255             print(f"Genres: {', '.join(anime['genres'])}")
256             print(f"Episodes: {anime['episodes']}")
257             print()
258     try:
259         pilihan = int(input("Pilih nomor anime yang ingin diperbarui: ")) - 1
260         if 0 <= pilihan < len(hasil_pencarian):
261             anime_to_update = hasil_pencarian[pilihan]
262             new_title = input("Masukkan judul anime baru: ").strip()
263             new_year = int(input("Masukkan tahun rilis baru: "))
264             new_genres = input("Masukkan genre baru (pisahkan dengan koma): ").split(", ")
265             new_episodes = int(input("Masukkan jumlah episode baru: "))
266             pembaruan = False
267             for anime in anime_data:
268                 if anime["title"].strip().lower() == anime_to_update["title"].strip().lower():
269                     anime["title"] = new_title
270                     anime["year"] = new_year
271                     anime["genres"] = new_genres
272                     anime["episodes"] = new_episodes
273                     pembaruan = True
274             print("Anime berhasil diperbarui.")
275             break
276         if pembaruan:
277             simpan_data("data_anime.json", anime_data)
278     else:
```

#### 1. Deskripsi Program

Fungsi `update_anime(data)` digunakan untuk memperbarui informasi mengenai anime tertentu yang disimpan dalam file JSON bernama `data_anime.json`. Program memungkinkan pengguna untuk mencari anime berdasarkan judul, memilih anime yang sesuai dari hasil pencarian, dan mengganti informasi seperti judul, tahun rilis, genre, dan jumlah episode.

#### 2. Tujuan Program

1. Memberikan kemampuan untuk memperbarui data anime yang sudah ada di file JSON.
2. Menyediakan antarmuka berbasis teks untuk mencari, memilih, dan mengedit data anime dengan mudah.
3. Menyimpan perubahan yang dilakukan pengguna kembali ke file JSON agar data tetap konsisten.

### 3. Cara Kerja Program

#### 1. Memuat Data Anime

Data anime dimuat dari file JSON menggunakan fungsi `load_data()`, yang membaca file `data_anime.json`.

#### 2. Mencari Anime Berdasarkan Judul

Pengguna diminta memasukkan judul anime yang ingin diperbarui. Program menggunakan fungsi `cari_dengan_judul()` untuk mencari daftar anime yang sesuai. Pencarian dilakukan secara case-insensitive (tidak sensitif terhadap huruf besar/kecil).

#### 3. Menampilkan Hasil Pencarian

Jika ada hasil pencarian, program menampilkan daftar anime yang cocok dalam format berikut:

```
1. Title: Attack on Titan
   Year: 2013
   Genres: Action, Adventure
   Episodes: 25
```

#### 4. Memilih Anime untuk Diperbarui

Pengguna diminta memilih nomor dari daftar hasil pencarian untuk menentukan anime yang akan diperbarui. Program memvalidasi input agar nomor yang dimasukkan valid.

#### 5. Memperbarui Informasi Anime

Pengguna memasukkan data baru untuk judul, tahun rilis, genre, dan jumlah episode.

- Judul Baru: Diambil dengan input string.
- Tahun Baru: Diambil dengan input angka.
- Genre Baru: Diinput sebagai string dengan elemen yang dipisahkan koma, lalu diubah menjadi daftar.
- Jumlah Episode Baru: Diambil dengan input angka.

#### 6. Menyimpan Data yang Diperbarui

Program memperbarui data pada anime yang sesuai dalam daftar `anime_data`. Jika pembaruan berhasil:

- Data diperbarui dalam memori.

- Fungsi `simpan_data()` dipanggil untuk menyimpan perubahan ke file `data_anime.json`

## 7. Penanganan Kesalahan

Program menangani beberapa potensi kesalahan, seperti:

- Input yang tidak valid (angka untuk pilihan).
- Pilihan nomor yang tidak sesuai dengan hasil pencarian.
- Jika anime yang dicari tidak ditemukan

```

287
288 # Fungsi untuk menambahkan anime ke
289 def create_anime(data, title, year, genres, episodes):
290     new_anime = {
291         "title": title,
292         "year": year,
293         "genres": genres,
294         "episodes": episodes
295     }
296     data.append(new_anime)
297     return hapus_duplikat(data)
298

```

### 1. Deskripsi Program

Fungsi `create_anime()` digunakan untuk menambahkan entri baru ke dalam daftar data anime. Fungsi ini menerima informasi berupa judul anime, tahun rilis, genre, dan jumlah episode, lalu menambahkannya ke dalam daftar data yang ada. Setelah penambahan, fungsi memastikan tidak ada entri duplikat dalam data dengan memanggil fungsi `hapus_duplikat()`.

### 2. Tujuan Program

1. Menambahkan data anime baru ke dalam koleksi data yang ada.
2. Mengelola data secara dinamis tanpa harus mengedit file secara manual.
3. Memastikan data yang tersimpan tetap konsisten dengan menghapus entri duplikat.

### 3. Cara Kerja Program

#### 1. Membuat Entri Baru

Fungsi menerima empat parameter utama:

- `title`: Judul anime dalam bentuk string.
- `year`: Tahun rilis anime dalam bentuk integer.
- `genres`: Daftar genre anime dalam bentuk list of strings.
- `episodes`: Jumlah episode dalam bentuk integer.

Data ini dimasukkan ke dalam dictionary baru bernama `new_anime` dengan format:

```
new_anime = {  
    "title": title,  
    "year": year,  
    "genres": genres,  
    "episodes": episodes  
}
```

## 2. Menambahkan Entri Baru ke Data

Entri `new_anime` ditambahkan ke dalam daftar data yang ada menggunakan:

## 3. Menghapus Duplikat

Setelah penambahan, fungsi `hapus_duplikat()` dipanggil untuk memastikan bahwa tidak ada entri yang sama dalam daftar data. Fungsi ini biasanya membandingkan setiap entri dalam daftar berdasarkan atribut tertentu (seperti judul anime).

## 4. Mengembalikan Data yang Diperbarui

Fungsi mengembalikan data yang telah diperbarui dan bebas dari duplikat.

## **BAB IV**

### **PENUTUP**

#### **4.1 Kesimpulan**

Kesimpulan dari program yang telah dibuat.

#### **4.2 Saran**

Saran yang diperlukan.

## **DAFTAR PUSTAKA**