🎥 YouTube Video Manager 📂

> A simple terminal-based Python project to manage your list of YouTube videos using **JSON** file storage. Perfect for beginners learning `file handling`, `functions`, and `basic CRUD` operations!

🚀 Features

✨ **List** all YouTube videos ➕ **Add** a new video with name and duration ✏️ **Update** any existing video info 🗑️ **Delete** a video 💾 Data is saved in a local file (`youtube.txt`) using JSON format

---

🧠 Concepts Used

- `Functions` & modular programming 🧩
- `File Handling` with `json` module 📁
- `Try-Except` error handling ⚠️
- `Match-case` structure (Python 3.10+) 🧪
- `List`, `Dict`, `Input/Output`, and more! 🔁

📦 Project Structure

```
youtube_video_manager.py
youtube.txt  # (auto-created JSON file to store video list)
```

📜 Code Explanation with Emojis

📥 1. Load Video Data

```python
def load_data():
    try:
        with open('youtube.txt', 'r') as file:
            return json.load(file)
    except FileNotFoundError:
        return []
```

👉 This function **loads video data** from `youtube.txt`. If the file is missing, it safely returns an empty list. ✅

💾 2. Save Helper

```python
def save_data_helper(videos):
    with open('youtube.txt', 'w') as file:
        json.dump(videos, file)
```

🩹 This helper function **saves the video list** back to the file in **JSON format**.

🧾 3. List All Videos

```python
def list_all_videos(videos):
    print("\n" + "*" * 70)
    for index, video in enumerate(videos, start=1):
        print(f"{index}. {video['name']}, Duration: {video['time']}")
    print("*" * 70)
```

👀 Display all stored videos in a numbered list. Clear & structured output. 🧾

➕ 4. Add New Video

```python
def add_video(videos):
    name = input("Enter video name: ")
    time = input("Enter video time: ")
    videos.append({'name': name, 'time': time})
    save_data_helper(videos)
```

🎧 Prompt user to add a new video — appends it to the list and saves! 🎉

✏️ 5. Update Existing Video

```python
def update_video(videos):
    list_all_videos(videos)
    index = int(input("Enter the video number to update"))
    if 1 <= index <= len(videos):
        name = input("Enter the new video name")
        time = input("Enter the new video time")
        videos[index-1] = {'name':name, 'time': time}
        save_data_helper(videos)
    else:
        print("Invalid index selected")
```

🛠️ Lists current videos, asks user which one to update, and edits that entry. 💡

🗑️ 6. Delete a Video

```python
def delete_video(videos):
    list_all_videos(videos)
    index = int(input("Enter the video number to be deleted"))
    if 1<= index <= len(videos):
        del videos[index-1]
        save_data_helper(videos)
    else:
        print("Invalid video index selected")
```

❌ Delete a video by index. Prevents crashes with safe checks! 🛡️

🧠 7. Main Logic – App Menu

```python
def main():
    videos = load_data()
    while True:
        print("\n Youtube Manager | choose an option ")
        print("1. List all youtube videos ")
        print("2. Add a youtube video ")
        print("3. Update a youtube video details ")
        print("4. Delete a youtube video ")
        print("5. Exit the app ")
        choice = input("Enter your choice: ")

        match choice:
            case '1':
                list_all_videos(videos)
            case '2':
                add_video(videos)
            case '3':
                update_video(videos)
            case '4':
                delete_video(videos)
            case '5':
                break
            case _:
                print("Invalid Choice")
```

💡 This is the main **menu-driven interface** using `match-case` (cleaner than if-else).

🧪 Sample `youtube.txt` Output

```
[
  {
    "name": "Learn Python Basics",
    "time": "15:32"
  },
  {
    "name": "React Crash Course",
    "time": "1:10:45"
  }
]
```

🎯 How to Run

```
python youtube_video_manager.py
```
