Report Videos Check

[Document subtitle]

Nguyen Thanh Vinh

2023

Contents

[State 1: Basic Understanding 2](#_Toc152810807)

[State 2: Outline Implementation 3](#_Toc152810808)

[I. Create\_video\_list.py 3](#_Toc152810809)

[II. Update\_video.py 4](#_Toc152810810)

[State 3: Basic Working Version 5](#_Toc152810811)

[State 4: Testing and Validation 5](#_Toc152810812)

[State 5: Innovations and Future Development 5](#_Toc152810813)

[Conclusion 5](#_Toc152810814)

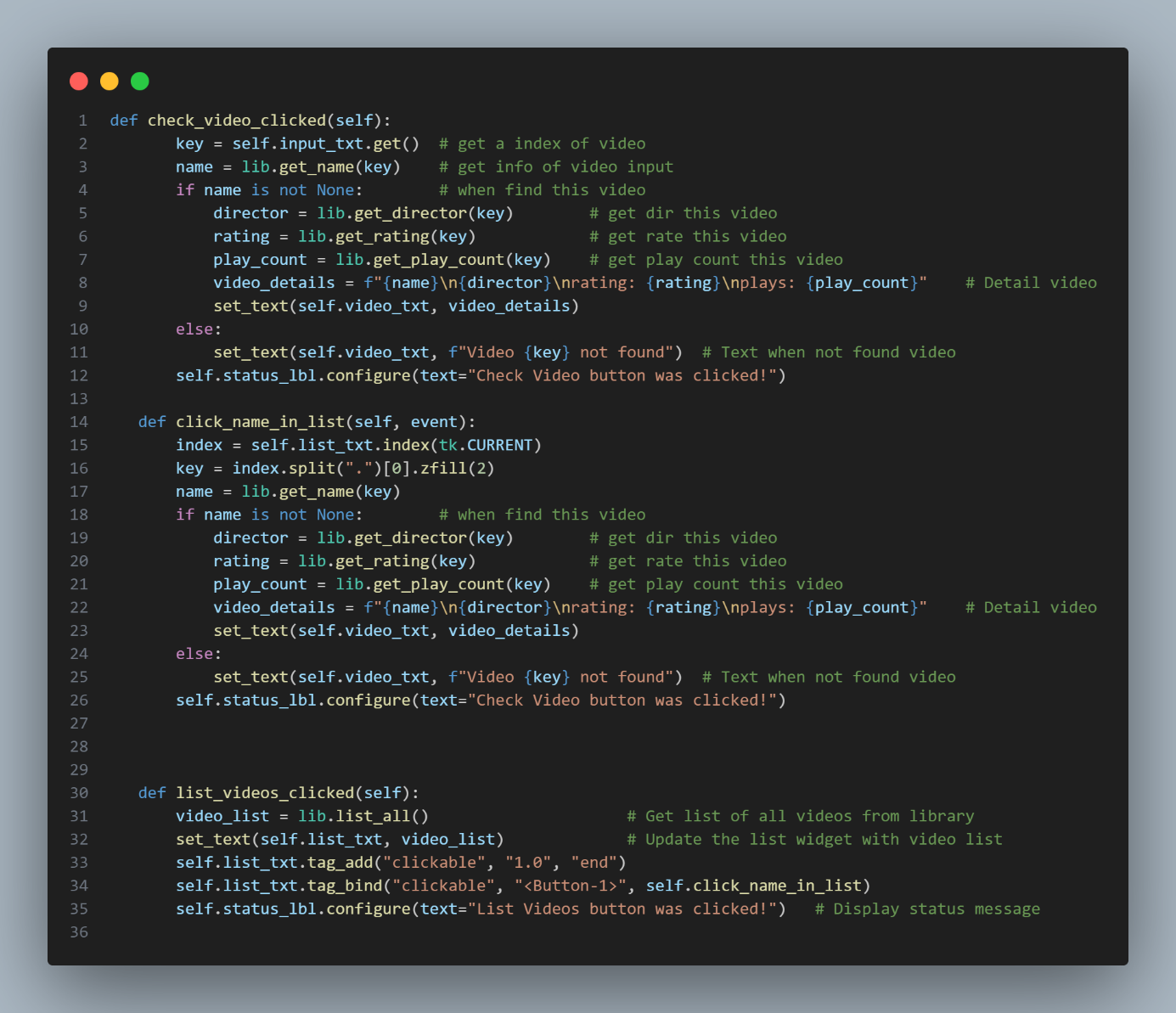
# State 1: Basic Understanding

In this stage, the **check\_video.py** script was thoroughly reviewed and documented. Each function’s purpose and functionality were explained with comments. For example, the **set\_text** function was documented as follows:

def set\_text(text\_area, content):   #Set text when open label

    text\_area.delete("1.0", tk.END) #Delete previous content

    text\_area.insert(1.0, content)  #Insert new content



# State 2: Outline Implementation

## Create\_video\_list.py

This outline provides a structured approach to building a video playlist application with Tkinter, featuring the ability to add, remove, play, and reset videos in the playlist. The methods are organized to handle different aspects of the application's functionality.

1. Initialization:

* The \_\_init\_\_ method initializes the main attributes and sets up the initial state of the application.
* The window is configured with a specific geometry and title
* **self.playlist** is a list that will store the selected videos for the playlist.

1. **ScrolledText Widgets:**

* Two ScrolledText widgets (**self.list\_txt** and **self.playlist\_text**) are created to display the list of all videos and the playlist, respectively.
* The tag\_configure method is used to configure a tag named 'clickable' for later use.

1. Entry and Buttons:

* An Entry widger (‘**self.video\_number\_entry**’) is created to input the video number
* Two buttons (‘**self.btn\_add\_video’** and ‘**self.btn\_remove\_video’**) are created to add and remove videos from the playlist, respectively
* Two more buttons (‘self.btn\_play\_video’ and ‘self.btn\_reset\_video’) are create to play the playlist and reset it

1. Methods:

* ‘**list\_videos\_clicked**’: Fetches the list of all videos from the library (‘**lib.list\_all()**’) and displays it in the **self.list\_txt** widget using the set\_text function.
* ‘**update\_playlist\_text**’: Updates the **self.playlist\_text** widget with the current playlist content and binds the "clickable" tag to each video for future interaction.
* ‘**display\_error\_message’**: Displays an error message label with the given message on the window.
* **‘add\_video’**: Gets the video name based on the entered video number, adds it to the playlist if valid, and updates the playlist text.
* **‘add\_video\_clicked’**: Handles the event when a video in the list is clicked. Adds the corresponding video to the playlist and updates the UI.
* **‘remove\_video’**: Removes a video from the playlist based on the entered video number and updates the UI.
* **‘play\_playlist’**: Iterates through the playlist, increments the play count for each video in the library, and prints the results.
* **‘reset\_playlist’**: Clears the playlist and updates the UI.

1. Main Block:

* The ‘if \_\_name\_\_ == “\_\_main\_\_” ‘: block creates a Tkinter window (‘**window**’), configures fonts, initializes the ‘**CreateVideo**’ class, and starts the Tkinter event loop with ‘**window.mainloop()**

## Update\_video.py

1. Initialization:

* The ‘**\_\_init\_\_**’ method sets up the initial state of the update video window.
* The window is configured with a specific geometry and title.
* Labels and entry widgets are created to input the video number and new rating.
* A Text widget (**self.video\_txt**) is used to display video information.
* Buttons for updating and checking video information are created.

1. Method ‘update\_rate’:

* This method is called when the "**Update**" button is clicked.
* It retrieves the video number and new rating from the entry widgets.
* If both fields are non-empty, it checks if the video exists using **lib.get\_rating(key)**
* If the video exists, it updates the rating using **lib.set\_rating(key, new\_rating)**.
* A success message is shown using messagebox.showinfo.
* The video number and new rating entry fields are cleared, and the video text is reset.
* If the video is not found, an appropriate message is displayed.

1. Method ‘check\_video\_clicked’:

* This method is called when the "**Check**" button is clicked.
* It retrieves the video number from the entry widget.
* If the video number is non-empty, it retrieves the video details (name, director, rating, play count) using library functions.
* The video details are formatted and displayed in the ‘**self.video\_txt**’ Text widget.
* If the video is not found, an appropriate message is displayed.
* The status label (‘**self.status\_lbl**’) is updated with a message indicating that the "**Check Video**" button was clicked.

1. Main block:

* The ‘**if \_\_name\_\_** == “**\_\_main\_\_**” ’: block create a Tkinter window (‘**window**’), configures fonts, initializes the ‘**UpdateVideo**’ class, and startss the Tkinter event loop with ‘**window.mainloop()**’

# State 3: Basic Working Version

# State 4: Testing and Validation

# State 5: Innovations and Future Development

# Conclusion