F01 Group 1H

Group Members (Student ID):
Sun Sitong (1007132)
Luvena Liethanti (1007105)
Pankti Shah (1007130)
Yee Jia Zhen (1006969)
Ho Xiaoyang (1007006)

Documentation

It is required for this program to use the Tkinter library and pandas library.
Pandas library is used to read the data from a csv file that we sourced from kaggle. After the dataset is input, it is converted into a dictionary, in which the keys are “song title”, ”artist”, and different filtrations for users to select. Each key corresponds to a list of music names or artist names.

The initial window for the user interface is set by root = tk. TK(), and the basic setting is assigned to the window.
To build multiple buttons and dropdown lists more efficiently, class objects for setting them are built separately.

class buttons(tk.Button):
This is the class object for the creation and adjustment of buttons. The _init_ method **def __init__(self, *args, **kwargs)** creates a button automatically when the class is called.

This class object also contains other methods:

1. **def button_state(self, state='active'),** which control the state of buttons

class dropdowns(OptionMenu):
This is the class object for the creation and adjustment of dropdown lists. The _init_ method **def __init__(self, *args, **kwargs)** creates a dropdown list with a pink background automatically when the class is called.

class main_info:
This is the class object to specify the behaviour of the “generate playlist” button, to print the playlist generated from clicking the options in the button to ensure that the playlist can be generated and is correct.

def button_activate(*args):
This function determines when to activate the button for the user to use. Its psuedocode is as follows:

1. Assign the drop down list’s state to a variable
2. Compare if the drop down list’s state is displaying the default text
    a. If true the button mentioned in the argument is disabled
    b. Else the button is activated

def onclick():
This function opens a new window and hides the first window.

def get_playlist_for_display():
This function returns the curated playlist that would be displayed after the user clicks on the button with “Generate Playlist” text. The psuedocode for this function is as follows:

1. If the selected drop down option is Genre it assigns the playlist variable to the list of songs and artists corresponding to the chosen genre.
2. If not then if the selected drop down option is Length it assigns the playlist variable to the list of songs and artists corresponding to the chosen length.
3. If it is neither genre nor length and the selected drop down option is year it assigns the playlist variable to the list of songs and artists corresponding to the chosen year.
4. Lastly if the drop down option is popularity it assigns the playlist variable to the list of songs and artists corresponding to the chosen popularity level.

def generate_playlist_genre(clicked):
This function picks songs from the dictionary according to the genre selected and stores it in another separate dictionary. The pseudocode is as follows:

1. An empty dictionary is created and assigned the name “playlist”.
2. Two empty lists are created, for the title and the artist.
3. If ‘Pop’ is the selected genre, then the list of pop songs in the database is picked out.
4. In the for-loop, every song and its corresponding title and artist are assigned the variables “first” and “second”, which are then appended into the empty lists for title and artist repectively.
5. The two lists are then added into the same empty dictionary that was created at the start of the function. The playlist is then returned to the new window.
6. The same thing is done for the other genres (dance, rap, rock, R&B, emo, edm and hip hop).

def generate_playlist_year(clicked):
This function picks songs from the dictionary according to the year range selected and stores it in another separate dictionary. The pseudocode is as follows:

1. An empty dictionary is created and assigned the name “playlist”
2. Six empty lists are created, for the title and the artist, and for the four year ranges.
3. In the for-loop, if the user selects the "Before 2000s” category, songs in the dictionary that were made before the year 2000 will be appended to the empty list, ls_1900s. The same is done for the 2000-2010, 2010-2015, and 2015-2021 lists.
4. If the user selects the “Before 2000s” category, the list ls will be assigned to ls_1900s. The same is done if the user selects the other three categories.
5. In the for-loop, every song in ls will be cross-checked with the dictionary, in order to separate the songs’ title and artist into the two empty lists (lstitle and lsartist).

6. The two lists are then added into the same empty dictionary that was created at the start of the function. The playlist is then returned to the new window.

def generate_playlist_length(clicked):
This function picks songs from the dictionary according to the length range selected and stores it in another separate dictionary.
The pseudocode is as follows:

1. An empty dictionary is created and assigned the name “playlist”
2. Six empty lists are created, for the title and the artist, and for the four length ranges.
3. In the for-loop, if the user selects the "1 to 2 min” category, songs in the dictionary that were 1 to 2 minutes long will be appended to the empty list, ls_2. The same is done for the 2- 3 min, 3- 4 min, and 4- 5 min lists.
4. If the user selects the “1 to 2 min” category, the list ls will be assigned to ls_2. The same is done if the user selects the other three categories.
5. In the for-loop, every song in ls will be cross-checked with the dictionary, in order to separate the songs’ title and artist into the two empty lists (lstitle and lsartist).
6. The two lists are then added into the same empty dictionary that was created at the start of the function. The playlist is then returned to the new window.

def generate_playlist_popularity(clicked):
This function picks songs from the dictionary according to the popularity level selected and stores it in another separate dictionary.
The pseudocode is as follows:

1. An empty dictionary is created and assigned the name “playlist”
2. Six empty lists are created, for the title and the artist, and for the four popularity ranges.
3. In the for-loop, if the user selects the "0 to 75 %” category, songs in the dictionary that were 0 to 75 % popular will be appended to the empty list, ls_75. The same is done for the 75 to 100% popularity lists.
4. If the user selects the “0 to 75 %” category, the list ls will be assigned to ls_75. The same is done if the user selects the other three categories.
5. In the for-loop, every song in ls will be cross-checked with the dictionary, in order to separate the songs’ title and artist into the two empty lists (lstitle and lsartist).
6. The two lists are then added into the same empty dictionary that was created at the start of the function. The playlist is then returned to the new window.

def appear(filter):
This is the function to create the secondary level dropdown list (named **drop2**) for users to make more detailed selections. The choices of users are stored at **clicked2**, which is used later to determine what music to display to the users. The pseudocode is as follows:

1. If the user selects to choose their songs by genre, then options_genre (the list of genre) will be stored in the second dropdown list.

2. The same is done if the user selects year, length, or popularity.
3. The moment the user clicks an option in the first dropdown list, the words “Filter: <Selected Option>” will appear in place of the “filter” dropdown list, which will disappear.

def reset():
This function resets the users’ choice by returning the window to its original state. When the user clicks the reset button, the two dropdown lists will return to their default value, showing “Filter” and “Selection Option”. The dropdown list “Filter” will appear again in its original place, and the “Selection Option” list will be hidden until a filter is selected from the first dropdown list. The text which states the selected filter will also be hidden and will be updated accordingly when a new filter is selected. The state of the button “generate playlist” will change to “disabled”.

def new_window(filter, type):
This function creates a second window which will display the playlist generated. A frame and canvas are added to this window for extra customization and also to include a scrollbar as the size of the content is beyond the canvas’ size.

It takes in two arguments, filter and type which will be used to display the filter and option selected by the user from the two dropdown lists.

This function contains a nested function
1. **def _on_mouse_wheel(event)**, which allows the user to scroll the page using their mouse wheel instead of clicking on the arrow buttons

The pseudocode is as follows:
1. A toplevel window is created and is assigned to new_window.
2. Resize is disabled for new_window.
3. Width and height of the window is set to be the same as the parent window.
4. A frame of the same size as new_window is created and is assigned to frame.
5. A canvas of the same size as new_window is created in the frame and assigned to my_canvas.
6. A scrollbar widget is created and is assigned to scrollbar. It is set to be vertical, placed at the right side of the window and fills the height of the window.
7. A command is passed to my_canvas such that the scrollbar is usable on my_canvas.
8. The scrollbar is configured such that it controls the whole window.
9. _on_mouse_wheel binds scrolling action to mouse wheel
10. A new frame is created with enough space to contain the contents which will be declared later and is assigned to top_msg. It is placed at the top of the window.
11. A new window is created in my_canvas and it is the top_msg frame itself.

12. A new frame is created with enough space to contain the playlist and is assigned to playlist_table. It is placed after top_msg .
13. A new window is created in my_canvas and it is the playlist_table frame itself.
14. Initialise altura, a variable that will be used later to alter the height of playlist_table.
15. Display "😊Enjoy your playlist😊" in Times New Roman, font size 20, underlined in top_msg frame and centered.
16. Display filter and option selected in Times New Roman, font size 16 in top_msg frame and left-aligned in the following format:
    {filter}: {option}
17. Creates a button at the top left of new_window with text "Go back" which allows user to return to the parent window root. It reopens root and destroys new_window.
18. The playlist to be shown is stored in playlist variable.
19. Display "Artist" in first row, first column in Arial, font size 12 and underlined.
20. Display " " (two spaces) in first row, second column.
21. Display "Title" in first row, third oclumn in Arial, font size 12 and underlined.
22. Initialise i to be 1
23. Iterate through the playlist dictionary and display the artist and title of the song in their respective rows and columns where row = i and add 1 to i. Add 30 to altura. Configured playlist_table such that its height is altura.

def label_center(lw):
This function takes in the width of a widget and returns the x coordinate where the widget should be placed at such that it is aligned at the center of the window.

def label_size_canvas(canvas, object):
This function returns a tuple with the width and height of a widget in a specified canvas.