Blusic

#Blue & Music

www.blusic.com

TEAMMATES

Kulanith Busabong
65110141

Paveetida Tiranatwittayakul
65110145

Saranya Vichakyotin
65110150

Watcharaphong Chiangthong
65110157

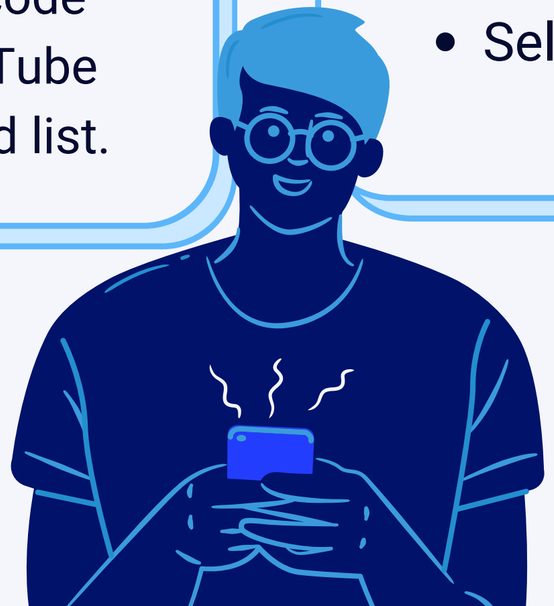
Table Of Content

Introduction

This Blusic Application is a Playlist Management Program with a graphical user interface (GUI) created using Tkinter. It allows users to add and remove songs, sort the playlist, and display song information. The code extracts song details from YouTube links and stores them in a linked list.

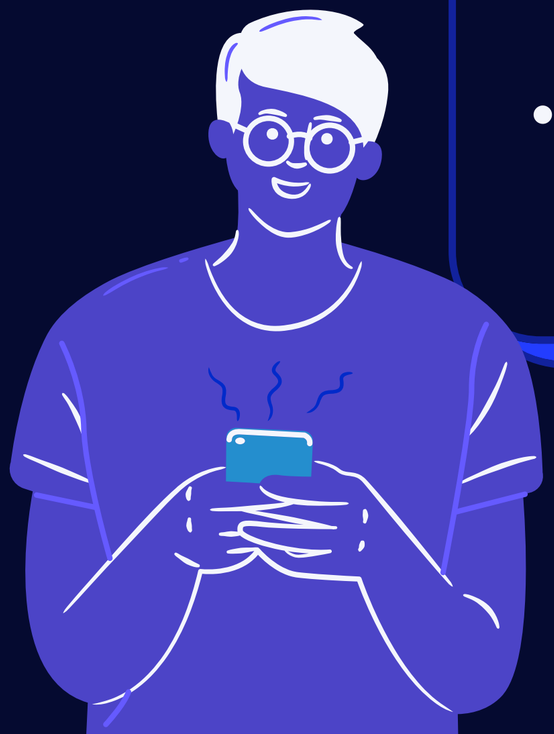
Key features include:

- Add music link from YouTube
- Search for song name
- Remove song at sorted position
- Display all songs in playlist
- Select sorting order and display



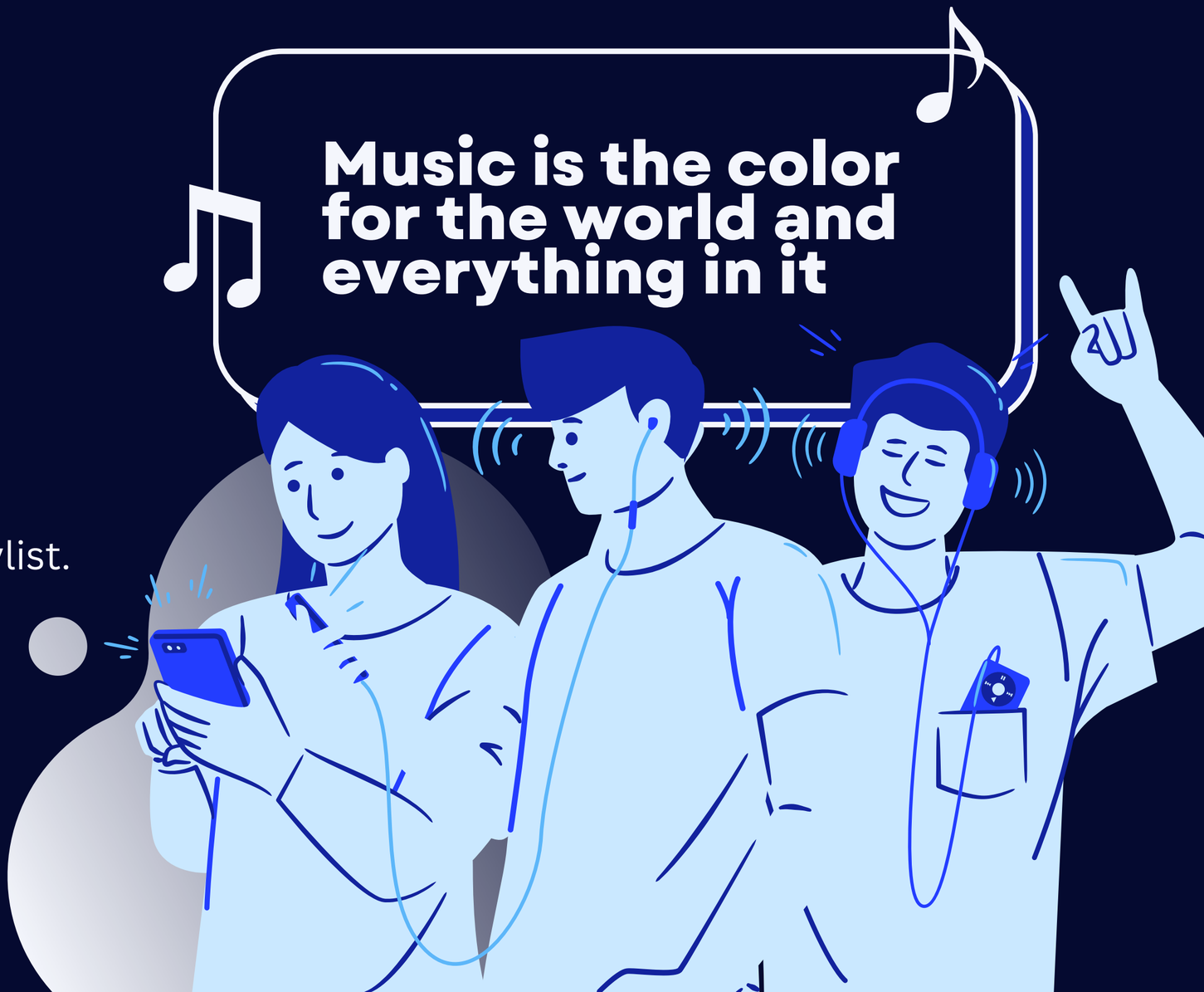
Why we choose this project?

- Playlists are commonly used in music and video platforms, and efficient sorting algorithms directly impact user experience.
- Playlists introduce complexities like user preferences, collaborative filtering, recommendations, and dynamic updates, offering diverse problem-solving challenges.
- Playlist sorting resonates with a wide audience, from tech enthusiasts to music and content lovers, making project outcomes engaging and widely relatable.



Why we choose this process?

1. Merge Sort:
   - Stable sorting.
   - Efficient for large playlists
   - Natural divide-and-conquer approach.
   - Easy merging of sorted playlists.
2. Binary Search:
   - Efficient searching for sorted data.
   - Enhances user navigation in a sorted playlist.
3. Linked List:
   - Efficient for insertions and deletions.
   - Low memory overhead and adaptable to dynamic changes.



Algorithm Applied

MERGE SORT

Use the “Merge sort” method to **sort playlists**. Uses an iterative sorting algorithm to repeatedly divide the playlist into halves, sorting each half. Then put them back together in the order in which they were arranged.

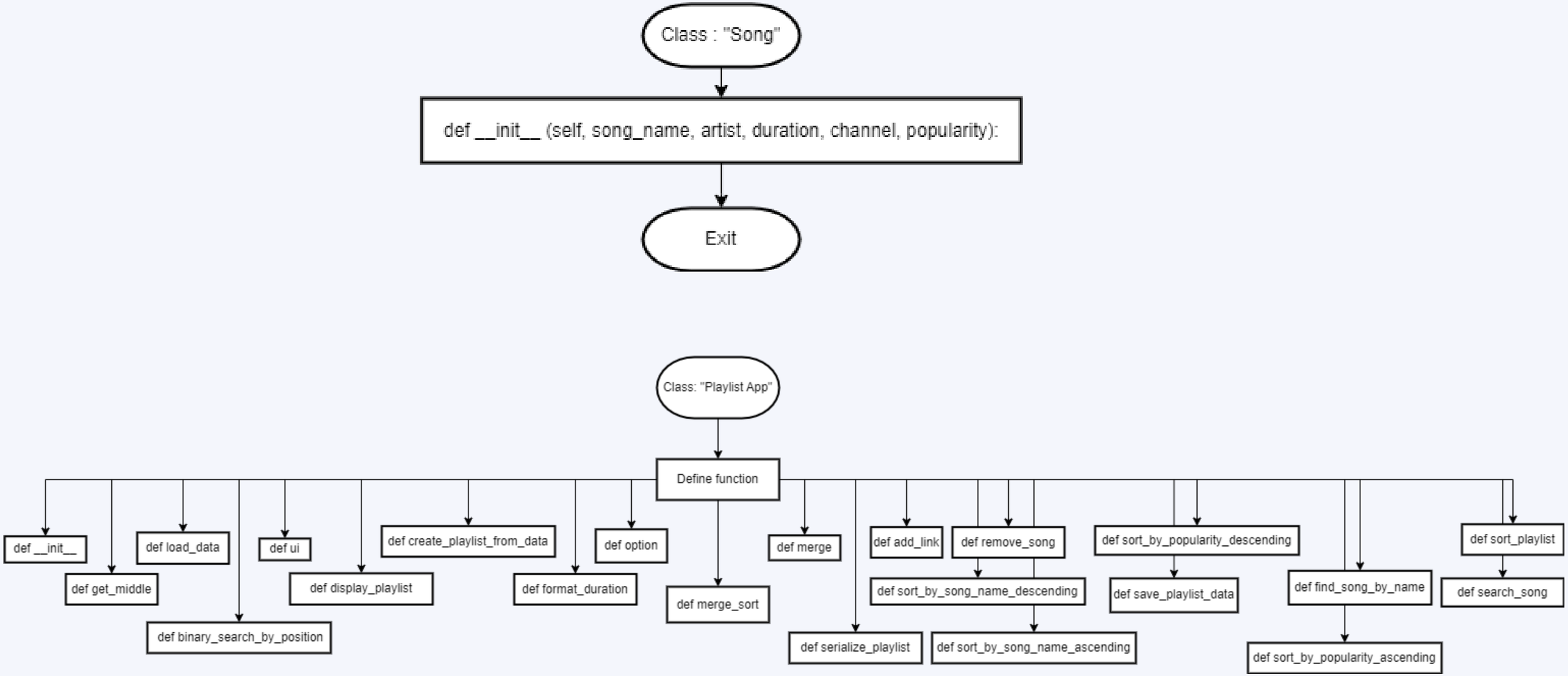
BINARY SEARCH

Use the “Binary search” method to **compare the current with original position**. Binary search is used to efficiently search for songs in a specified location.

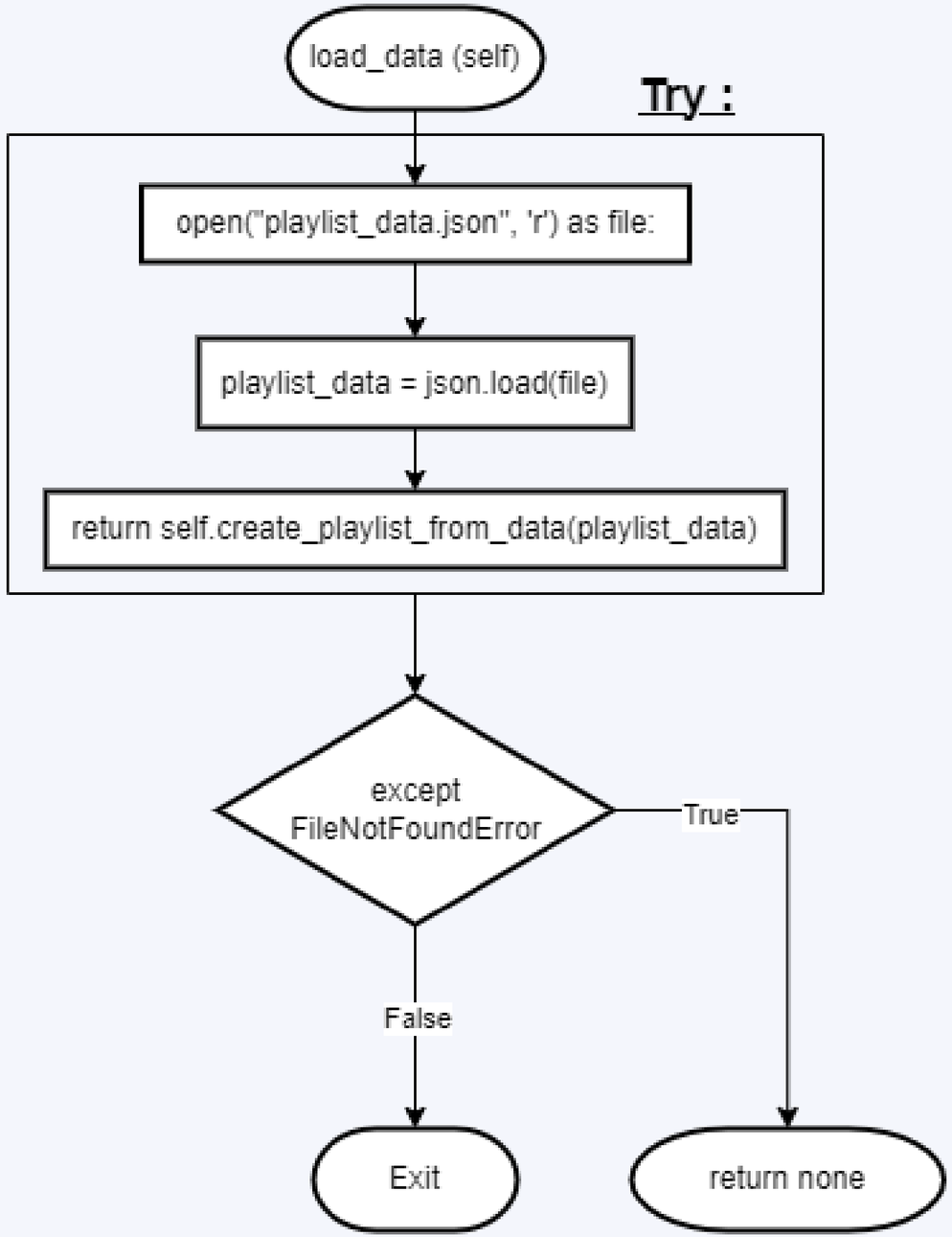
LINKED LIST

The use of a linked list allows for **dynamic management** of the playlist. Songs are added, removed, displayed, and sorted within the linked list data structure.

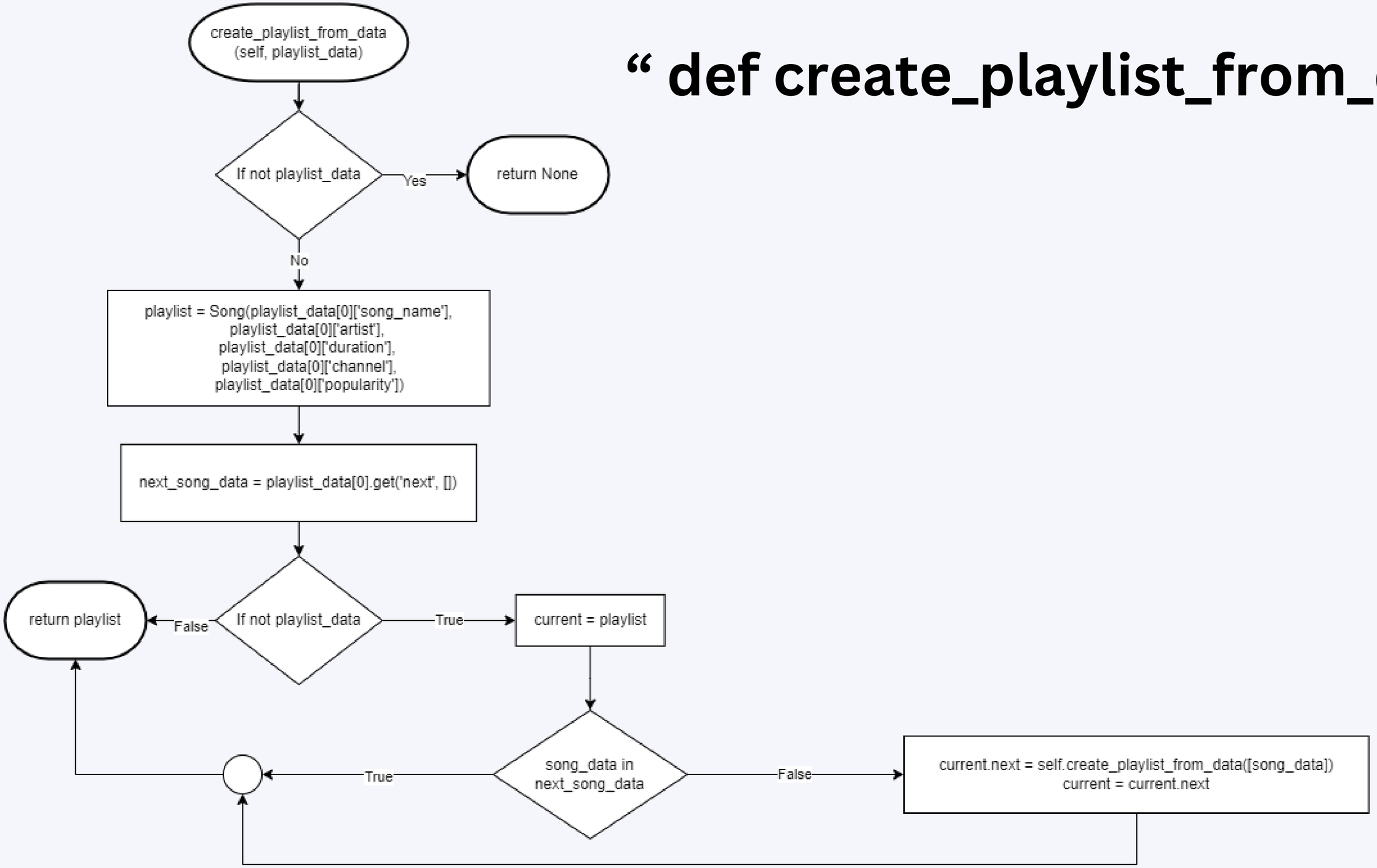
Flowchart


Class : "Song"
def __init__ (self, song_name, artist, duration, channel, popularity):
Exit
Class: "Playlist App"
Define function
def __init__
def get_middle
def load_data
def binary_search_by_position
def ui
def display_playlist
def create_playlist_from_data
def format_duration
def option
def merge_sort
def merge
def serialize_playlist
def add_link
def remove_song
def sort_by_song_name_descending
def sort_by_song_name_ascending
def sort_by_popularity_descending
def save_playlist_data
def find_song_by_name
def sort_by_popularity_ascending
def sort_playlist
def search_song

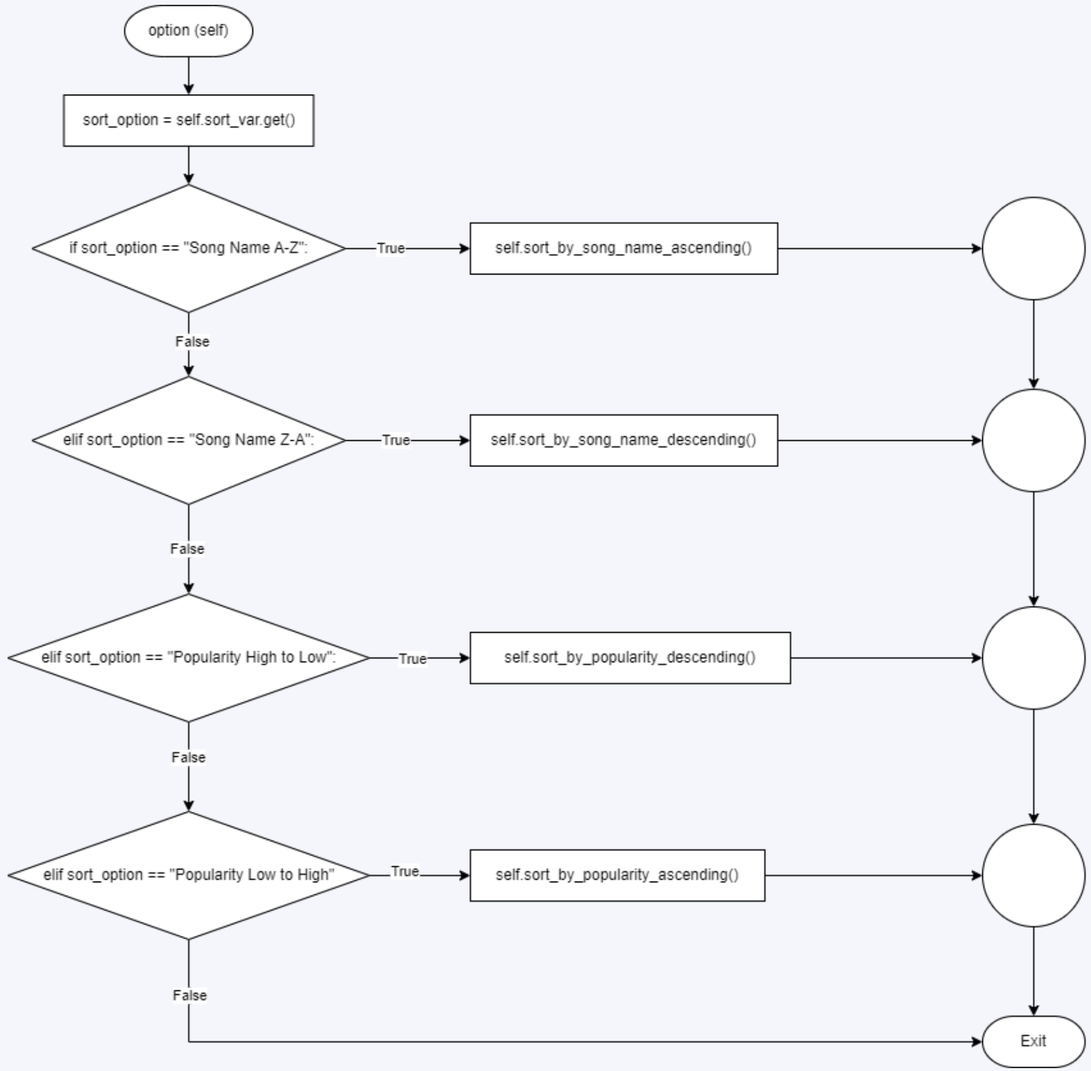
“ def load_data”



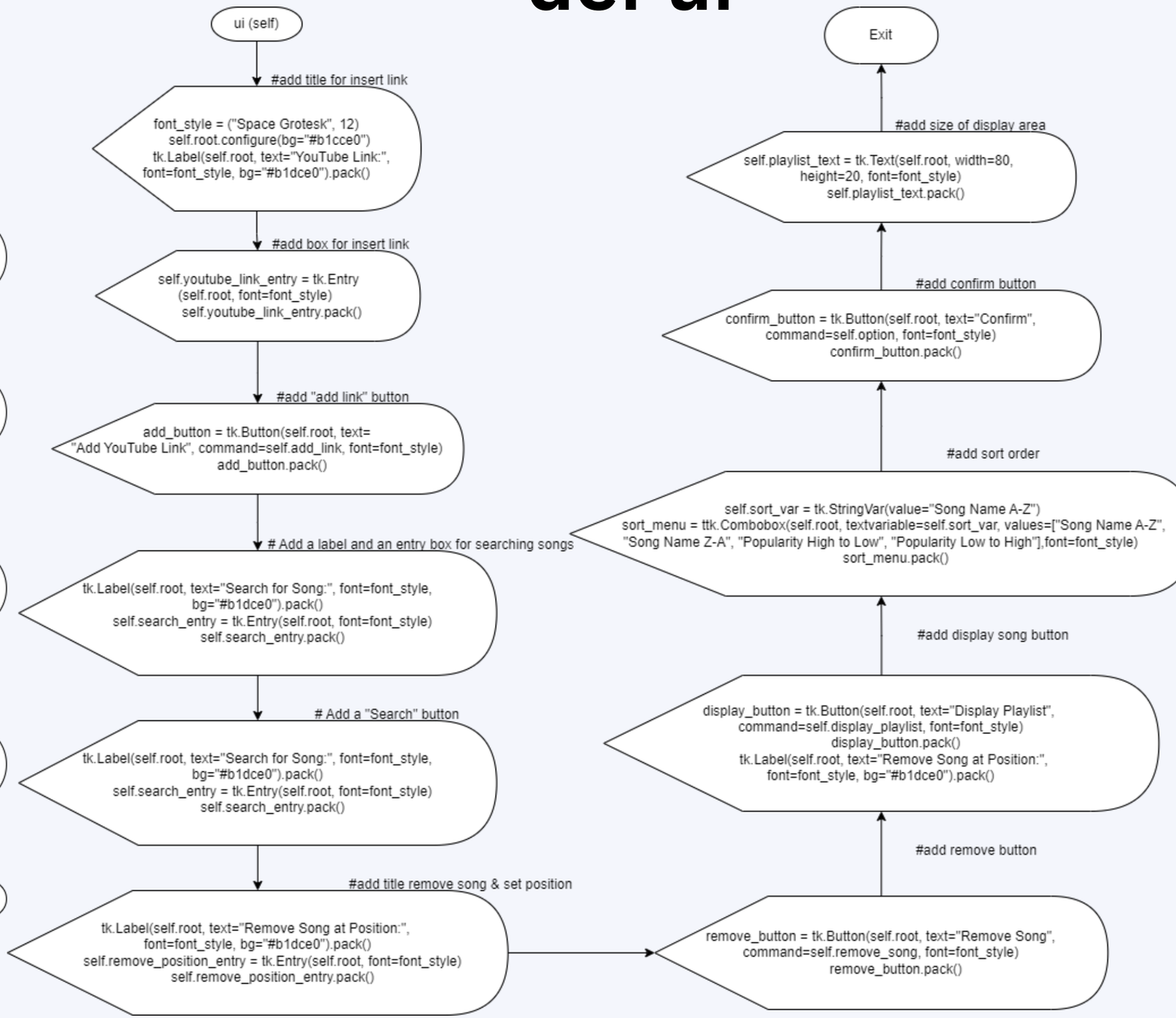
“ def create_playlist_from_data”



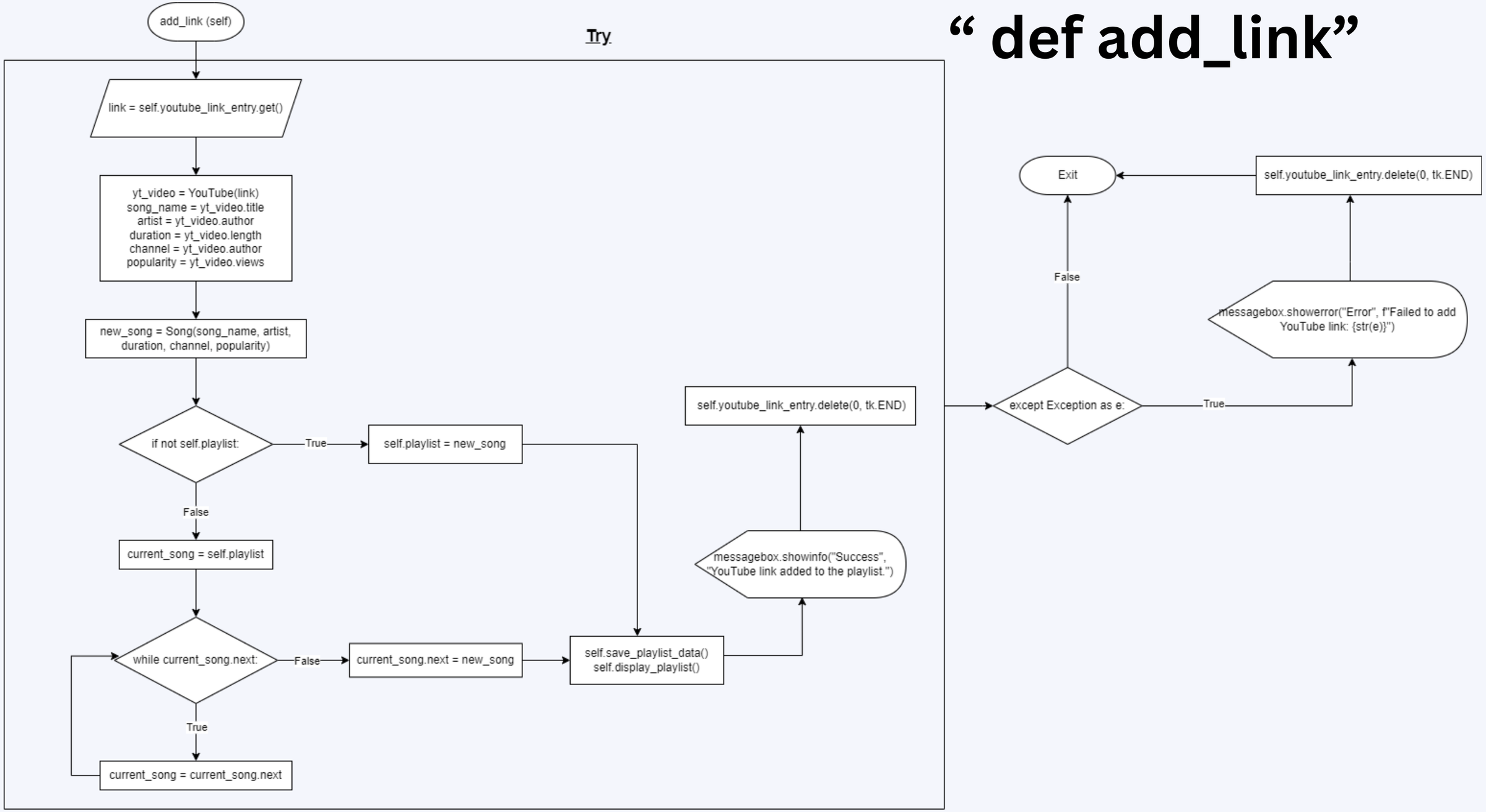
“ def option”



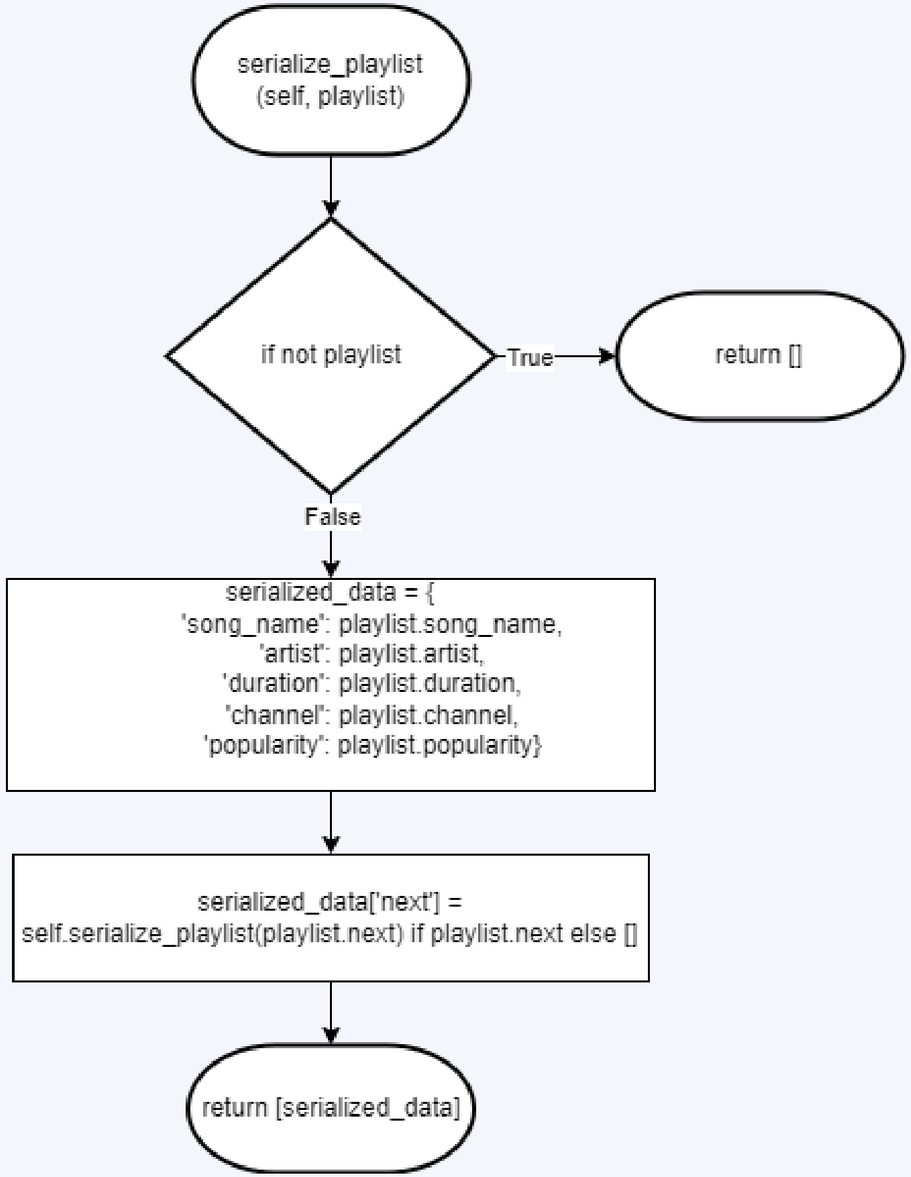
“ def ui”



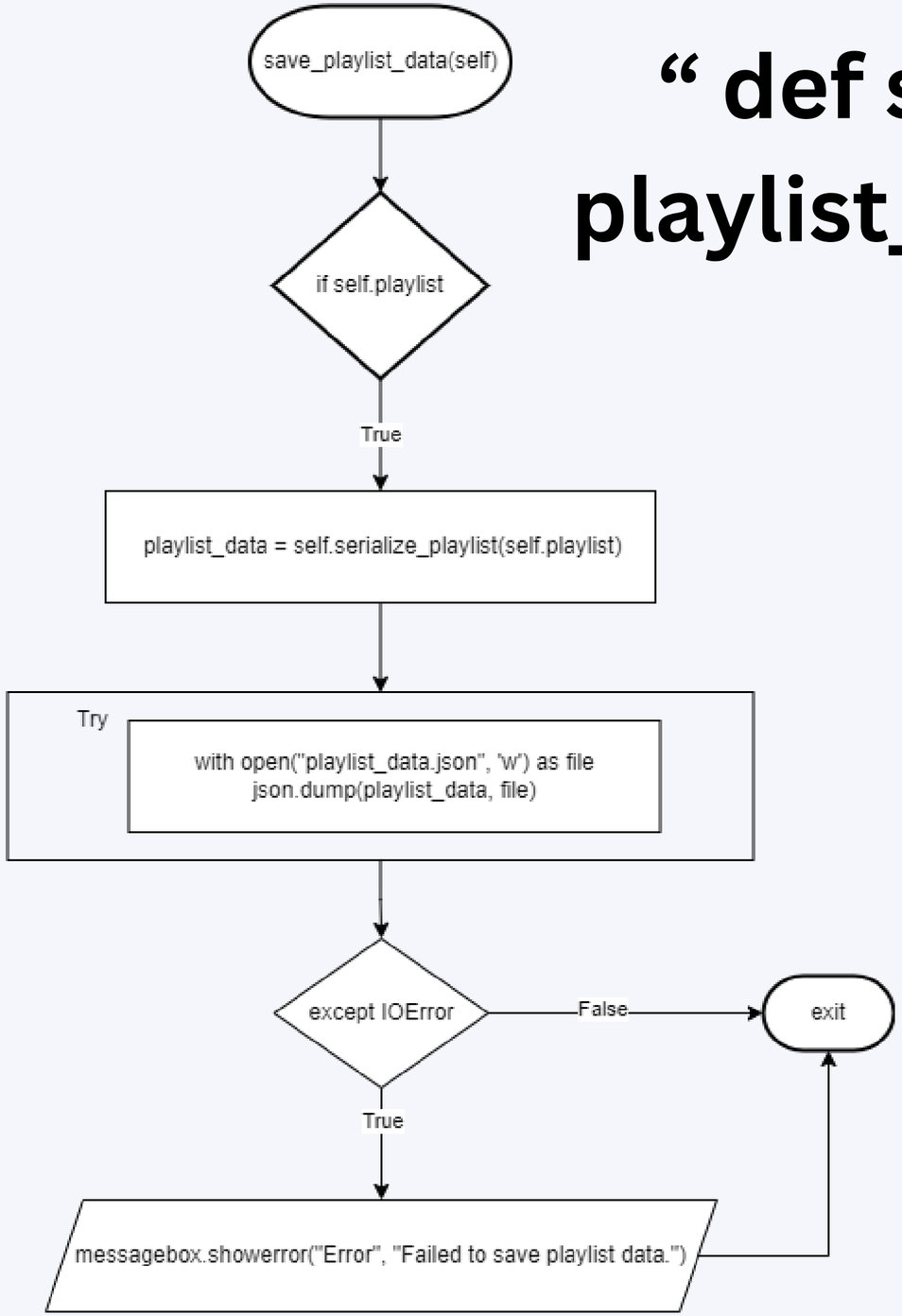
“ def add_link”


add_link (self)
Try
link = self.youtube_link_entry.get()
yt_video = YouTube(link)
song_name = yt_video.title
artist = yt_video.author
duration = yt_video.length
channel = yt_video.author
popularity = yt_video.views
new_song = Song(song_name, artist, duration, channel, popularity)
if not self.playlist:
True
self.playlist = new_song
False
current_song = self.playlist
while current_song.next:
False
current_song.next = new_song
True
current_song = current_song.next
self.save_playlist_data()
self.display_playlist()
messagebox.showinfo("Success", "YouTube link added to the playlist.")
self.youtube_link_entry.delete(0, tk.END)
except Exception as e:
True
messagebox.showerror("Error", f"Failed to add YouTube link: {str(e)}")
self.youtube_link_entry.delete(0, tk.END)
Exit
False

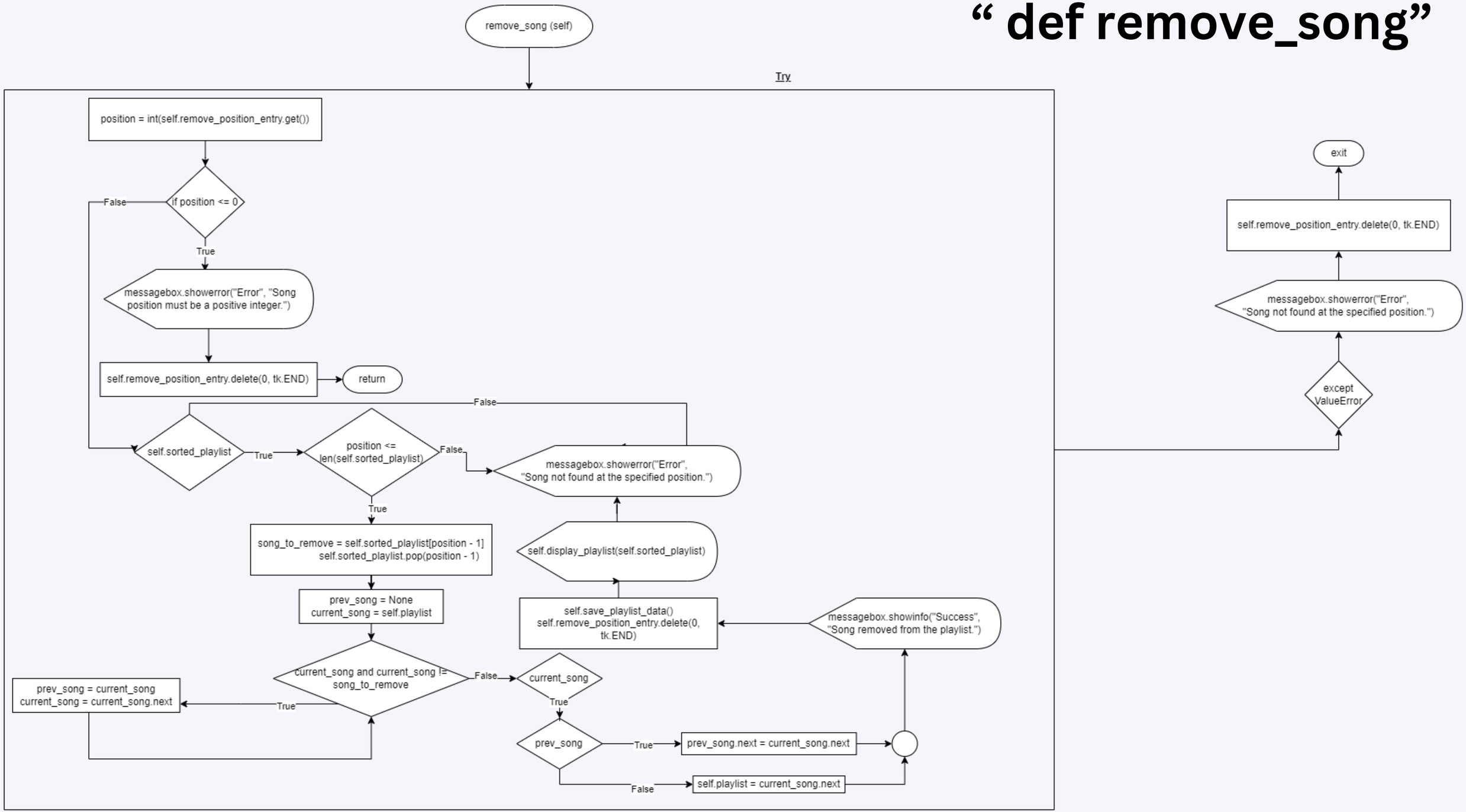
“ def serialize_playlist”



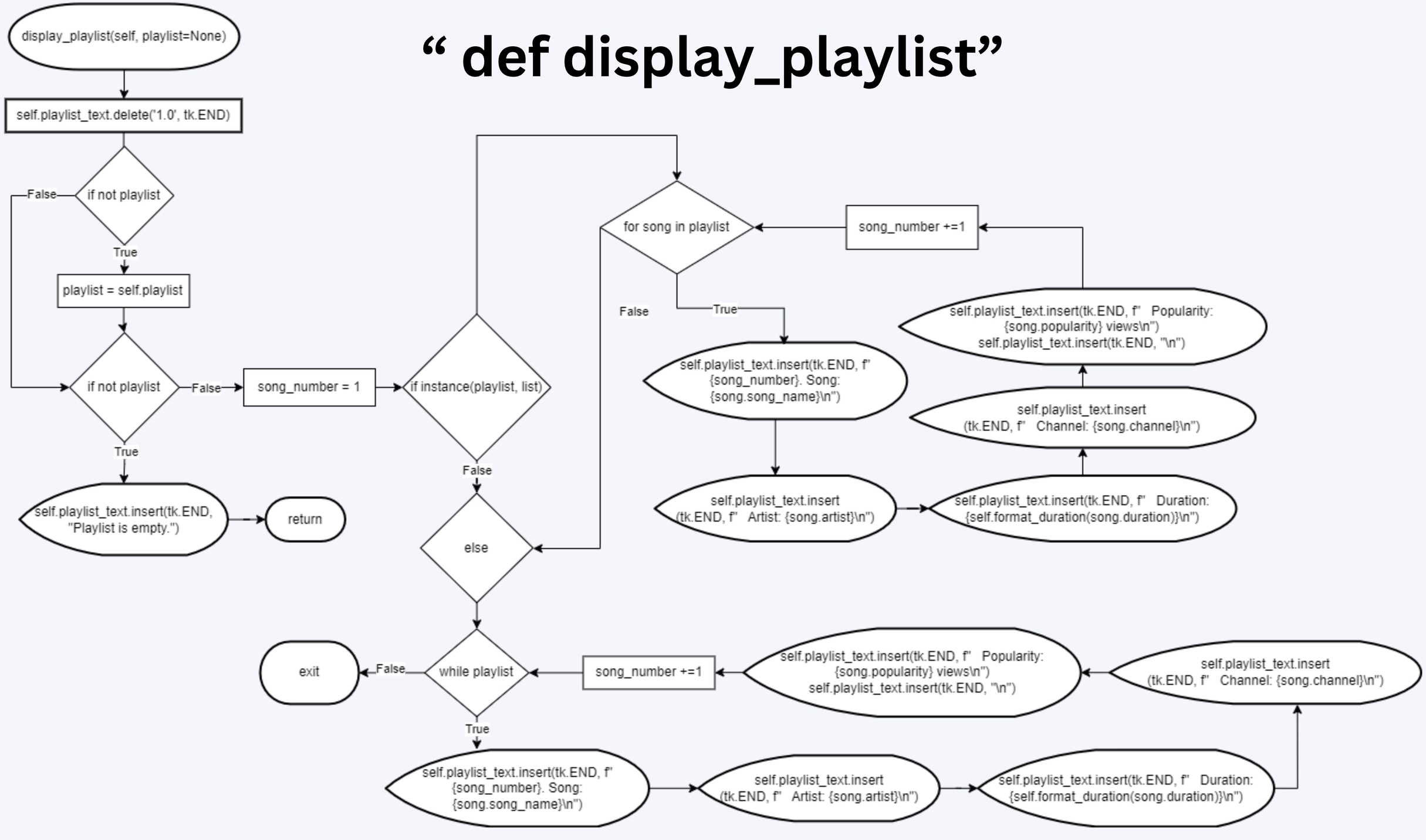
“ def save playlist_data”



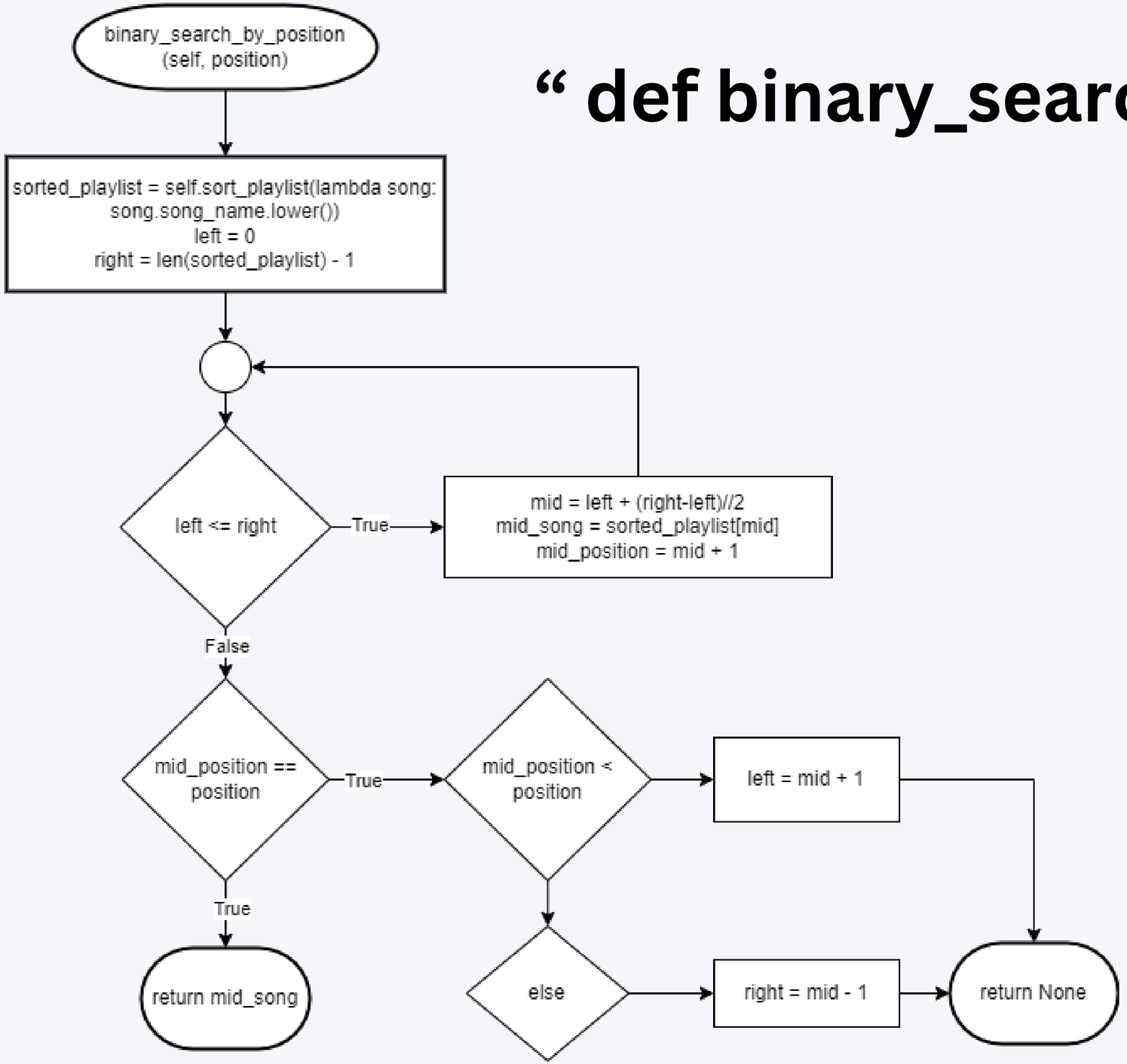
“ def remove_song”



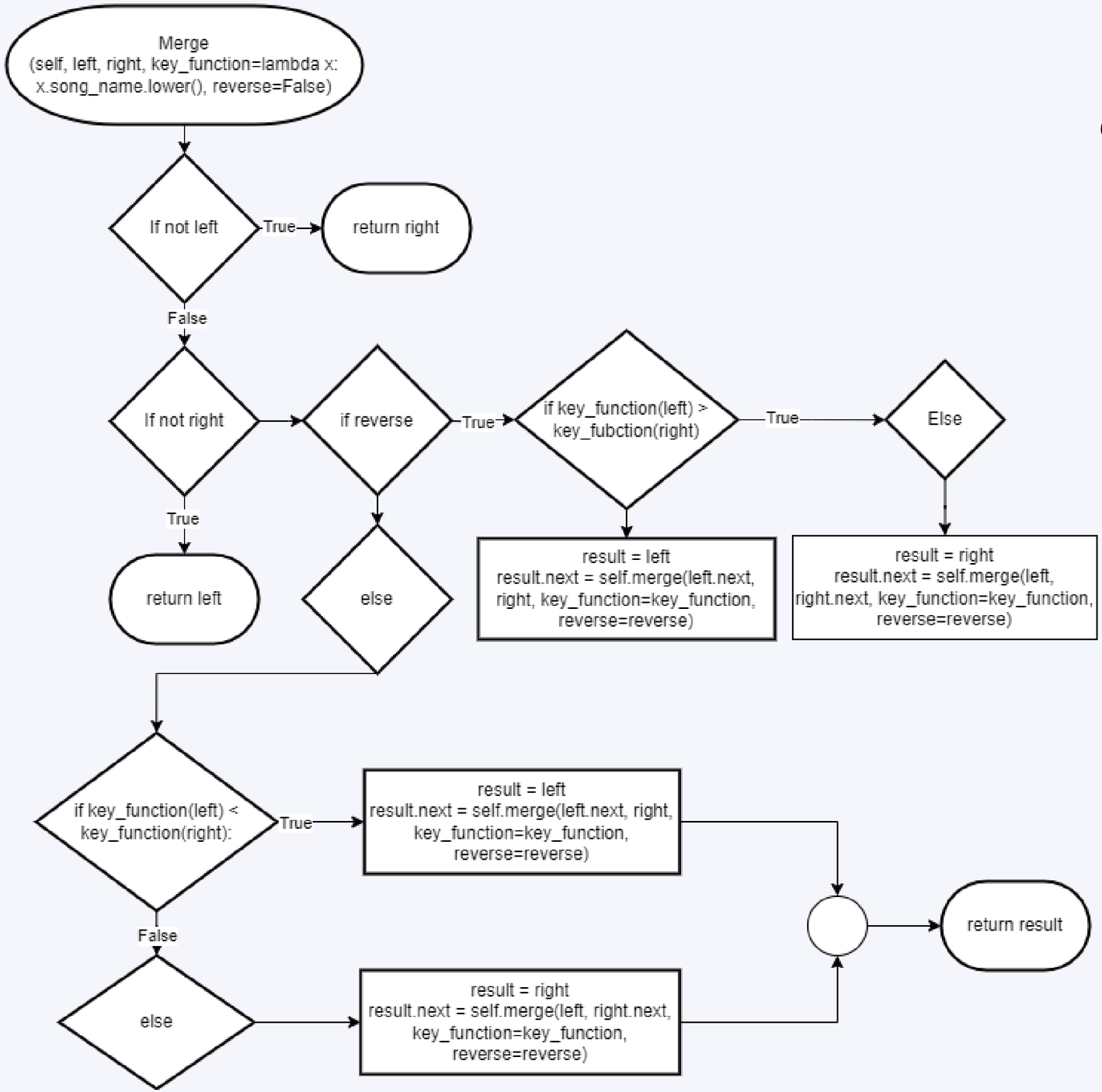
“ def display_playlist”



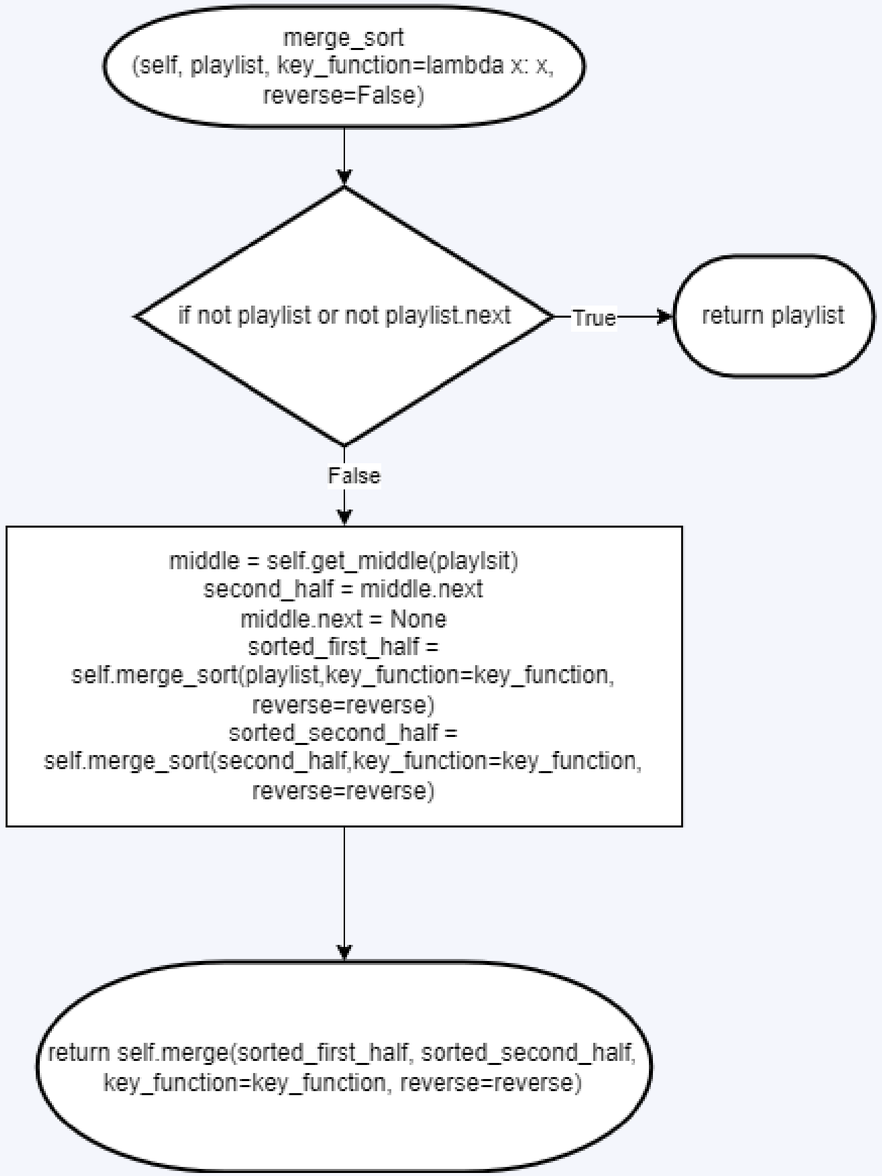
“ def binary_search_by_position”



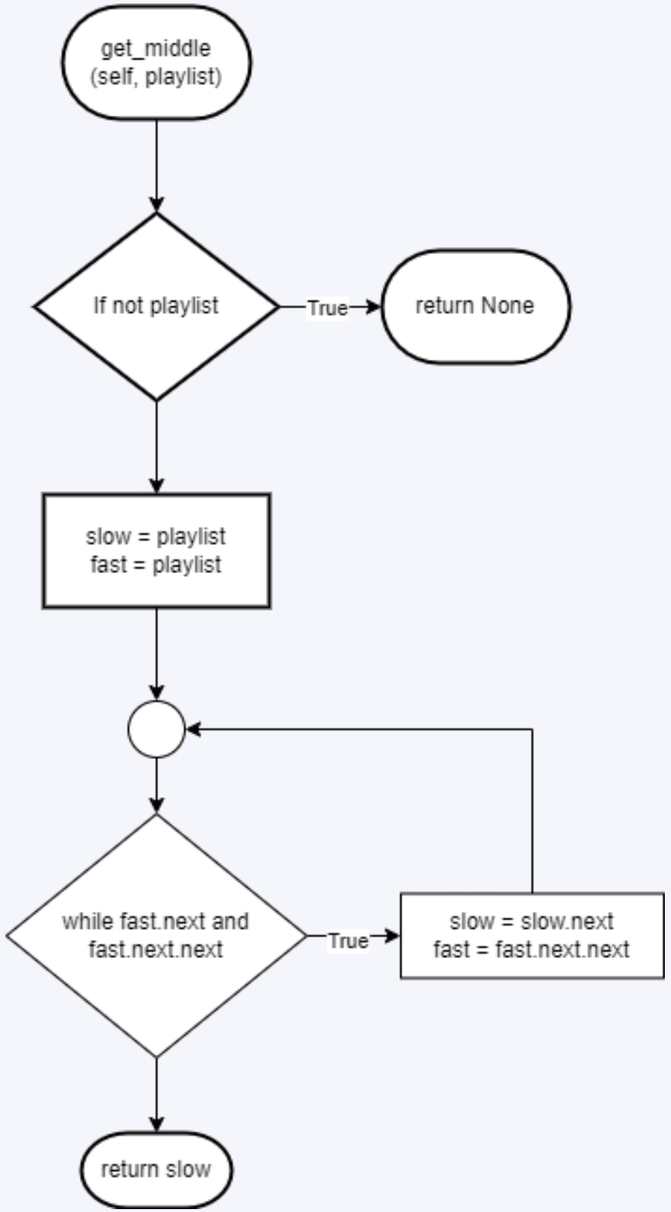
“ def merge”


Merge
(self, left, right, key_function=lambda x:
x.song_name.lower(), reverse=False)
If not left
True
return right
False
If not right
True
return left
if reverse
True
else
if key_function(left) >
key_fubction(right)
True
result = left
result.next = self.merge(left.next,
right, key_function=key_function,
reverse=reverse)
Else
result = right
result.next = self.merge(left,
right.next, key_function=key_function,
reverse=reverse)
if key_function(left) <
key_function(right):
True
result = left
result.next = self.merge(left.next, right,
key_function=key_function,
reverse=reverse)
False
else
result = right
result.next = self.merge(left, right.next,
key_function=key_function,
reverse=reverse)
return result

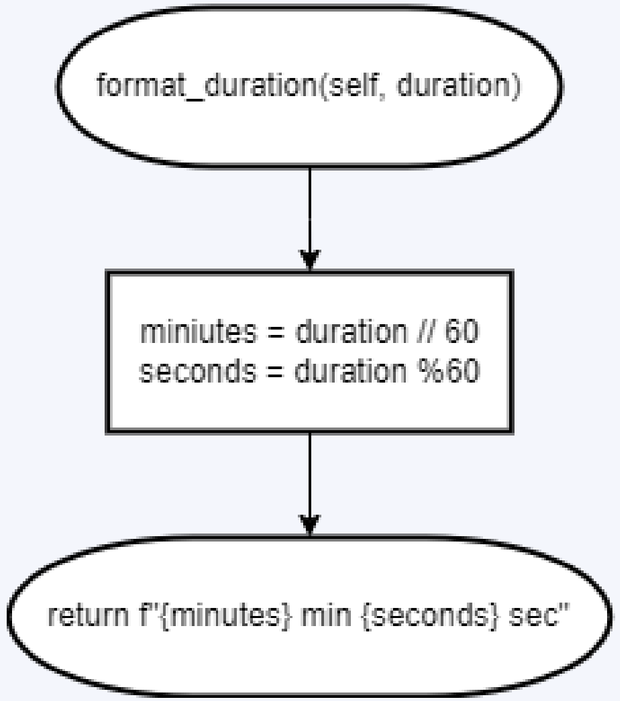
“ def merge_sort”



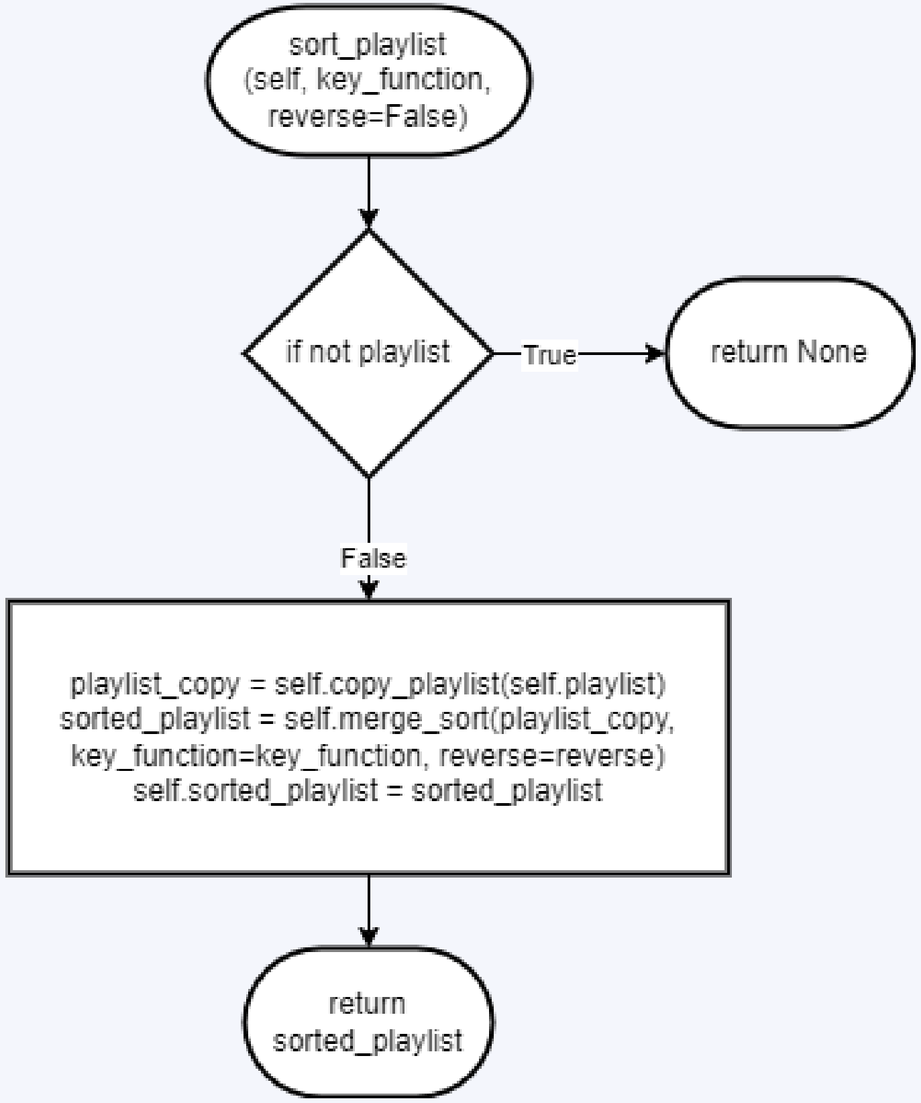
“ def get_middle”

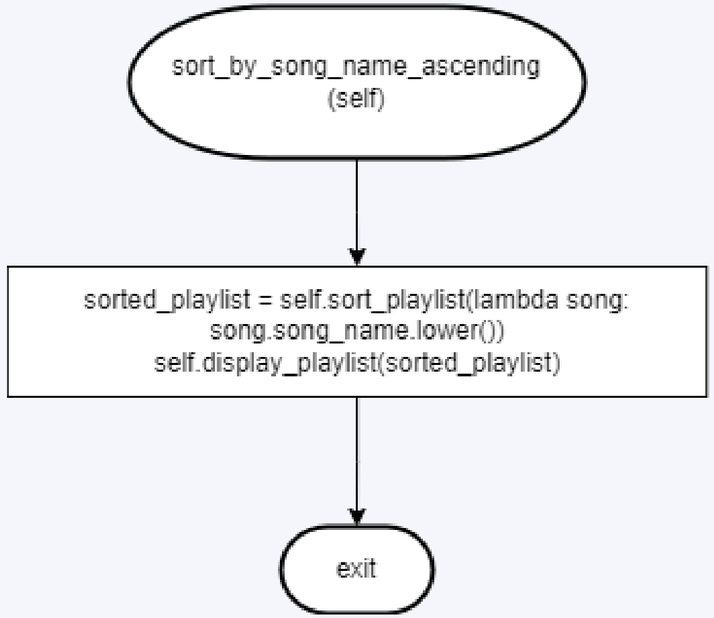


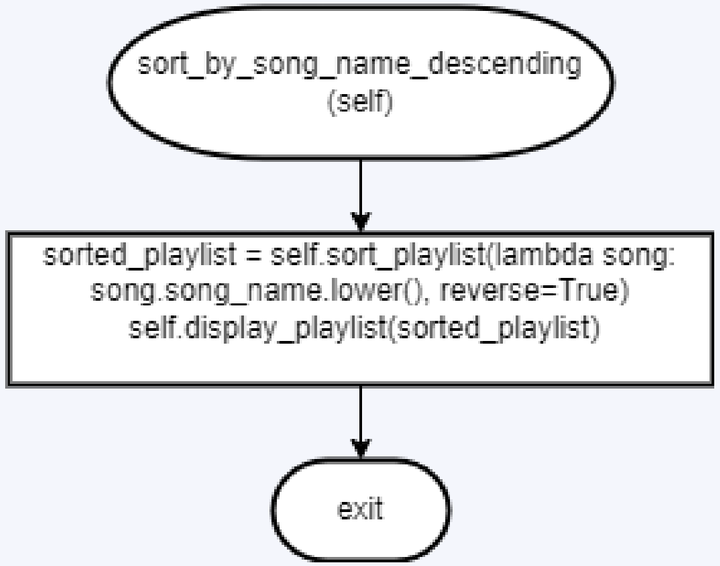
“def format_ duration”

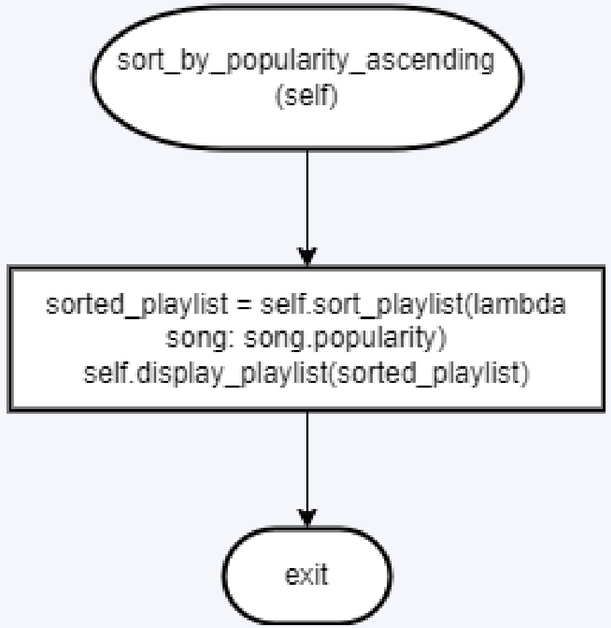


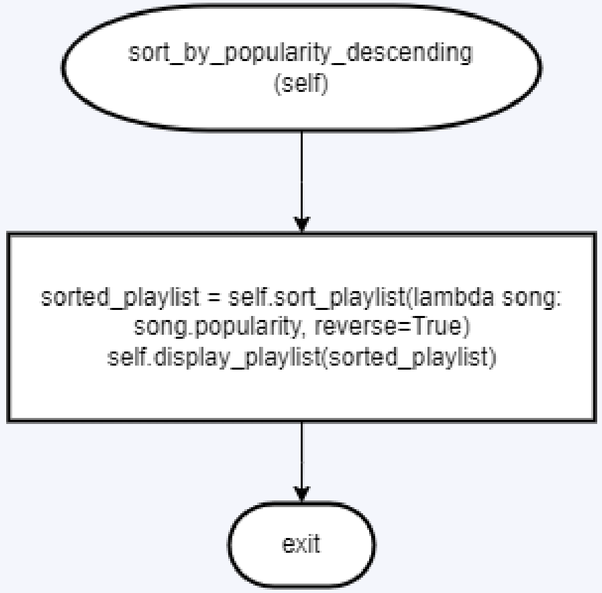
“ def sort_playlist”



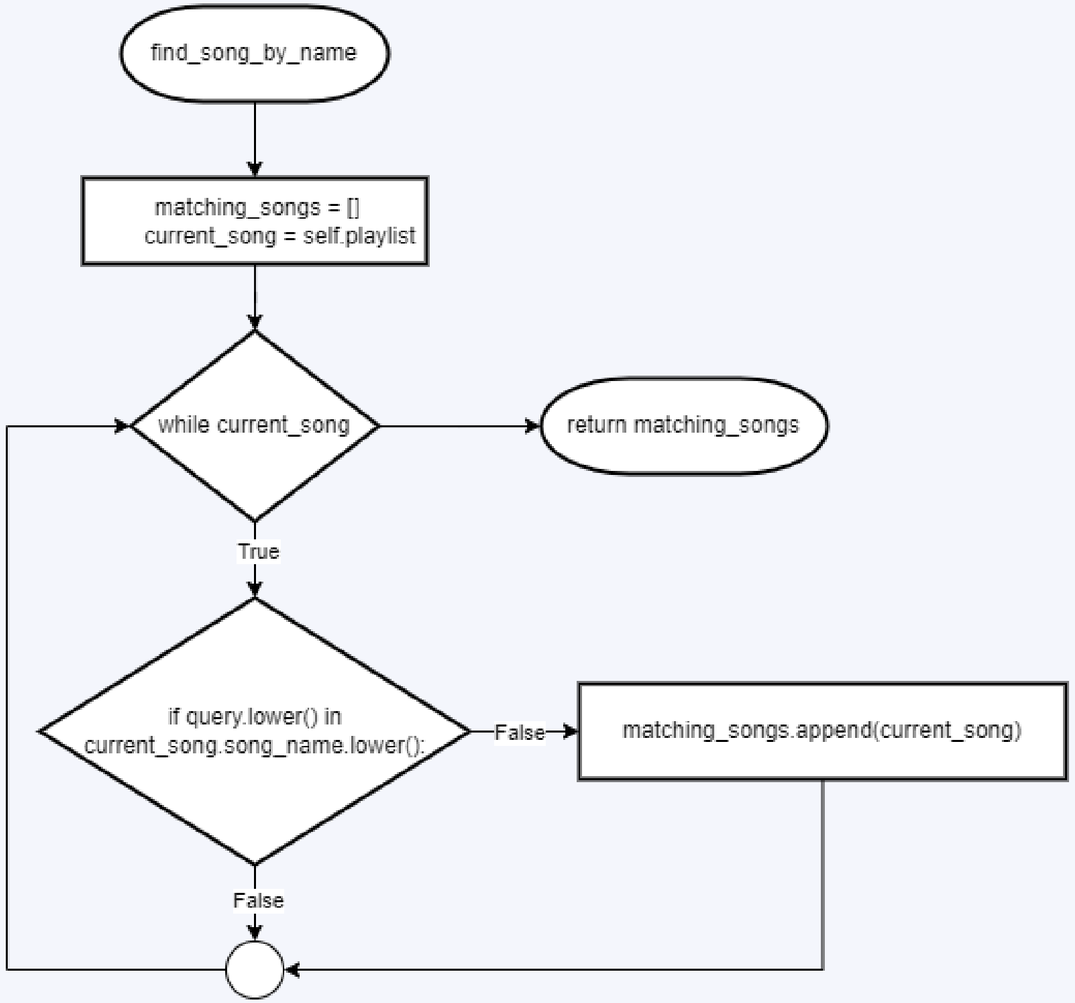




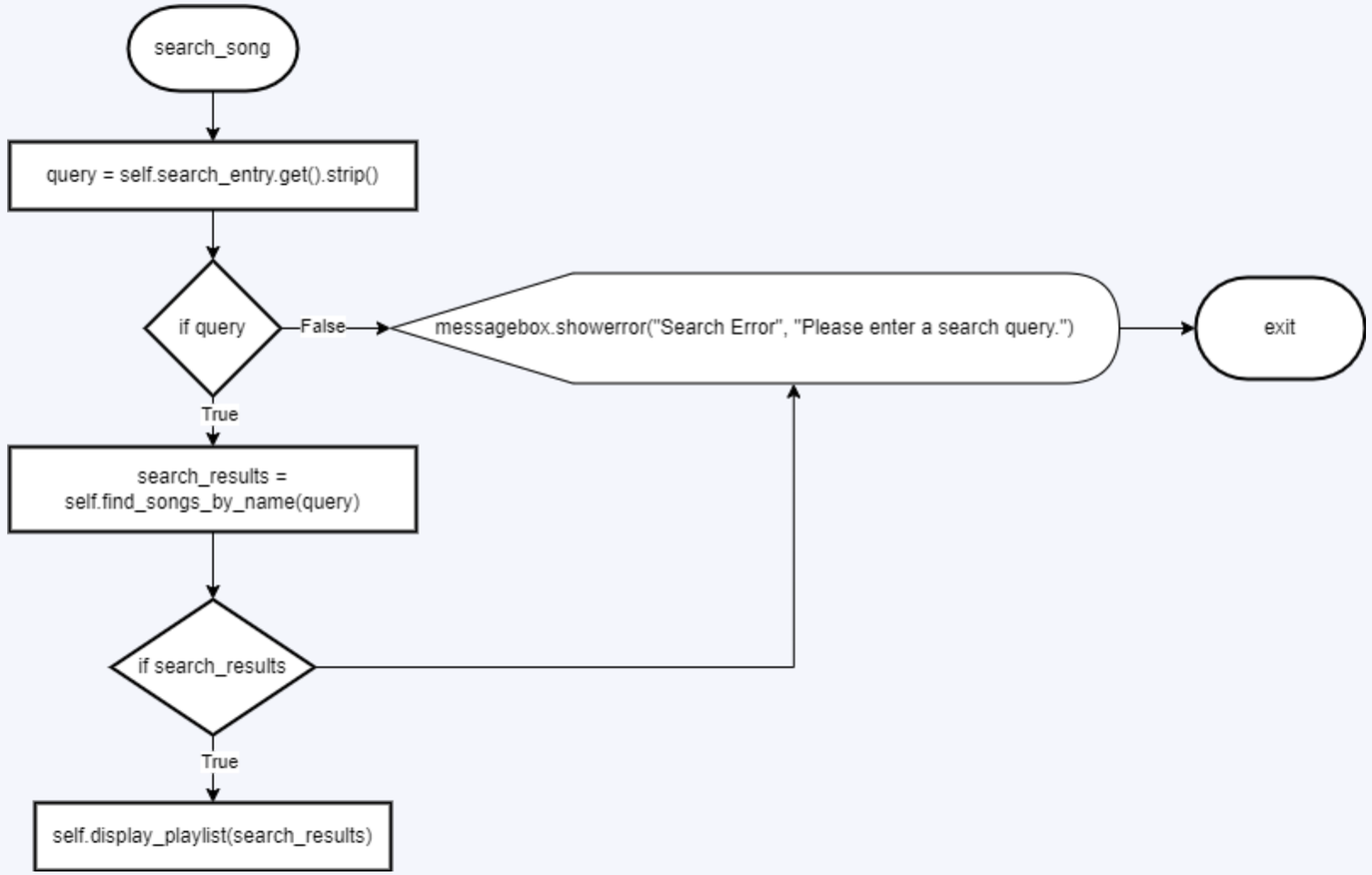




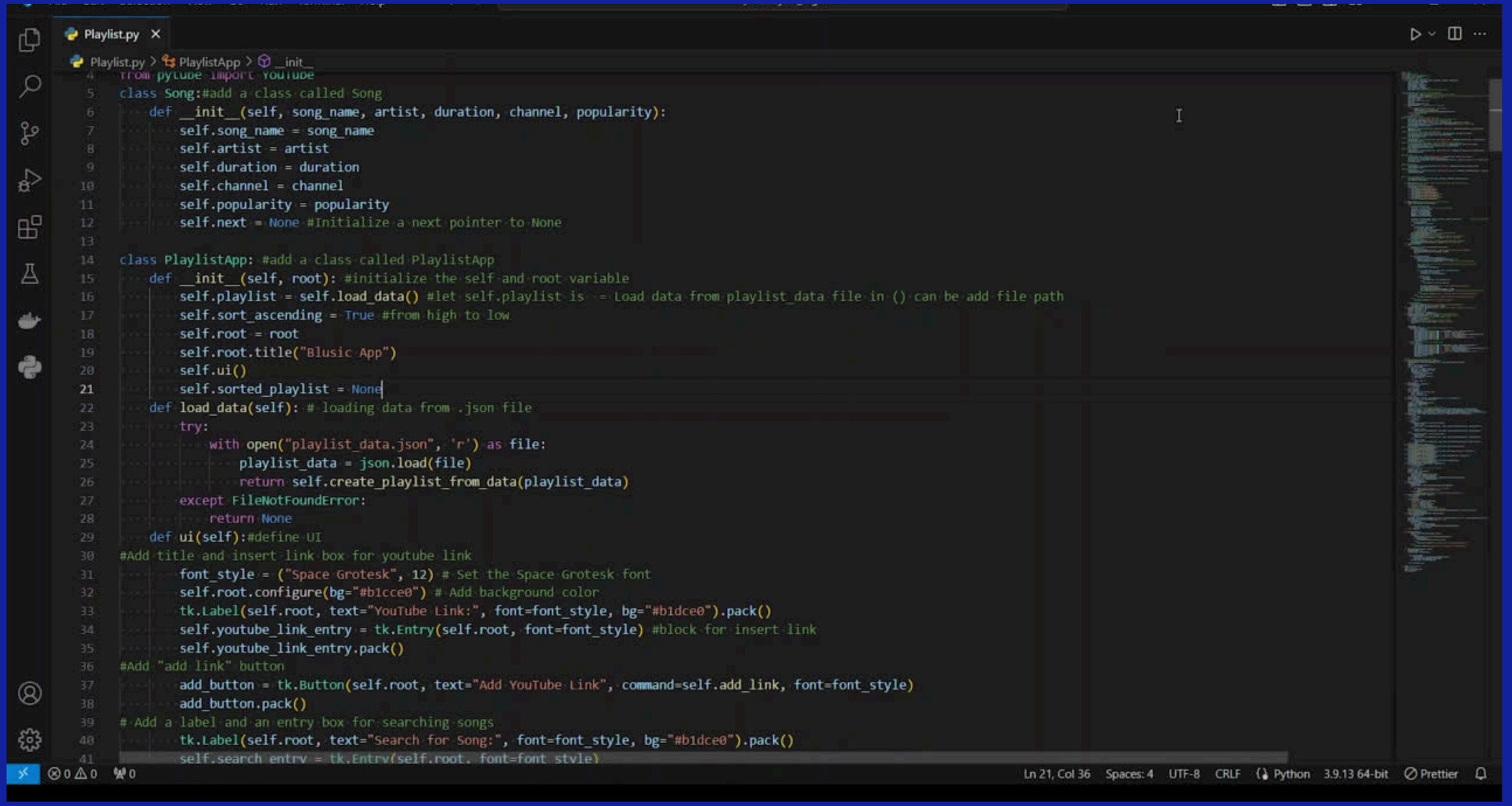
“ def find_song_name”



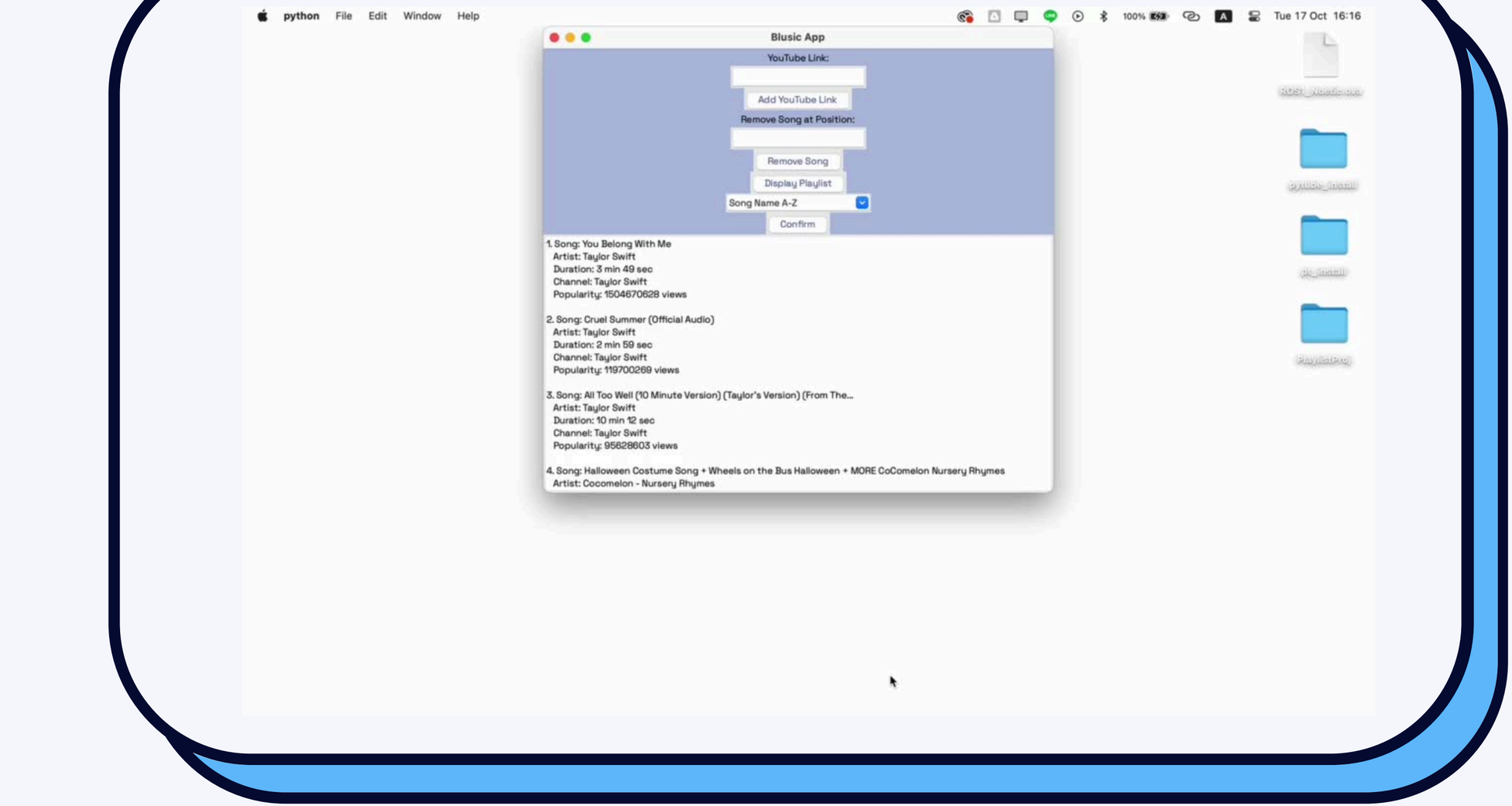
“ def search_song”



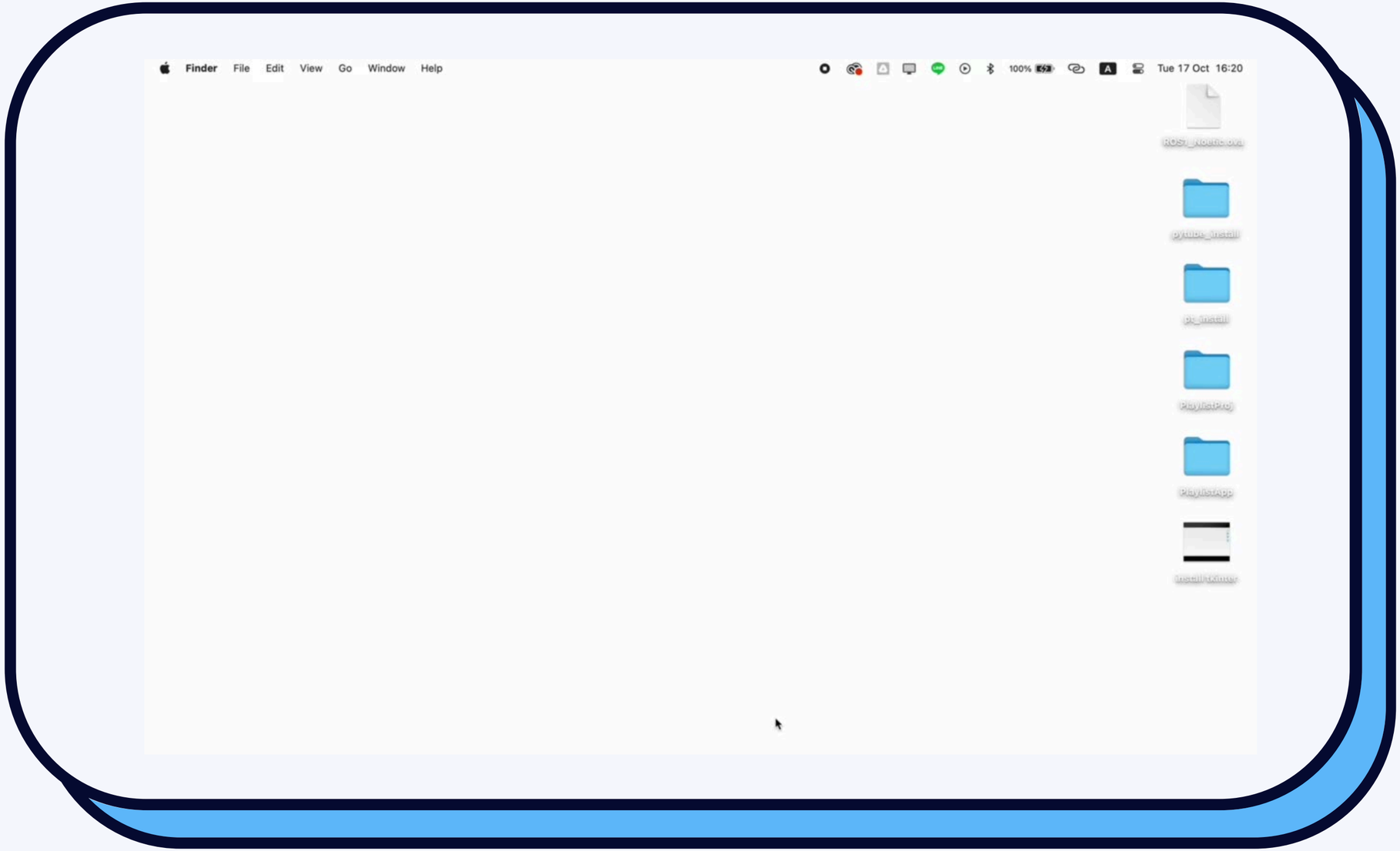
CODING


Playlist.py
Playlist.py > PlaylistApp > __init__
from pytube import YouTube
class Song:#add a class called Song
def __init__(self, song_name, artist, duration, channel, popularity):
self.song_name = song_name
self.artist = artist
self.duration = duration
self.channel = channel
self.popularity = popularity
self.next = None #Initialize a next pointer to None
class PlaylistApp: #add a class called PlaylistApp
def __init__(self, root): #initialize the self and root variable
self.playlist = self.load_data() #let self.playlist is = Load data from playlist_data file in () can be add file path
self.sort_ascending = True #from high to low
self.root = root
self.root.title("Blusic App")
self.ui()
self.sorted_playlist = None
def load_data(self): # loading data from .json file
try:
with open("playlist_data.json", 'r') as file:
playlist_data = json.load(file)
return self.create_playlist_from_data(playlist_data)
except FileNotFoundError:
return None
def ui(self):#define UI
#Add title and insert link box for youtube link
font_style = ("Space Grotesk", 12) # Set the Space Grotesk font
self.root.configure(bg="#b1cce0") # Add background color
tk.Label(self.root, text="YouTube Link:", font=font_style, bg="#b1dce0").pack()
self.youtube_link_entry = tk.Entry(self.root, font=font_style) #block for insert link
self.youtube_link_entry.pack()
#Add "add link" button
add_button = tk.Button(self.root, text="Add YouTube Link", command=self.add_link, font=font_style)
add_button.pack()
# Add a label and an entry box for searching songs
tk.Label(self.root, text="Search for Song:", font=font_style, bg="#b1dce0").pack()
self.search_entry = tk.Entry(self.root, font=font_style)
Ln 21, Col 36
Spaces: 4
UTF-8
CRLF
Python
3.9.13 64-bit
Prettier

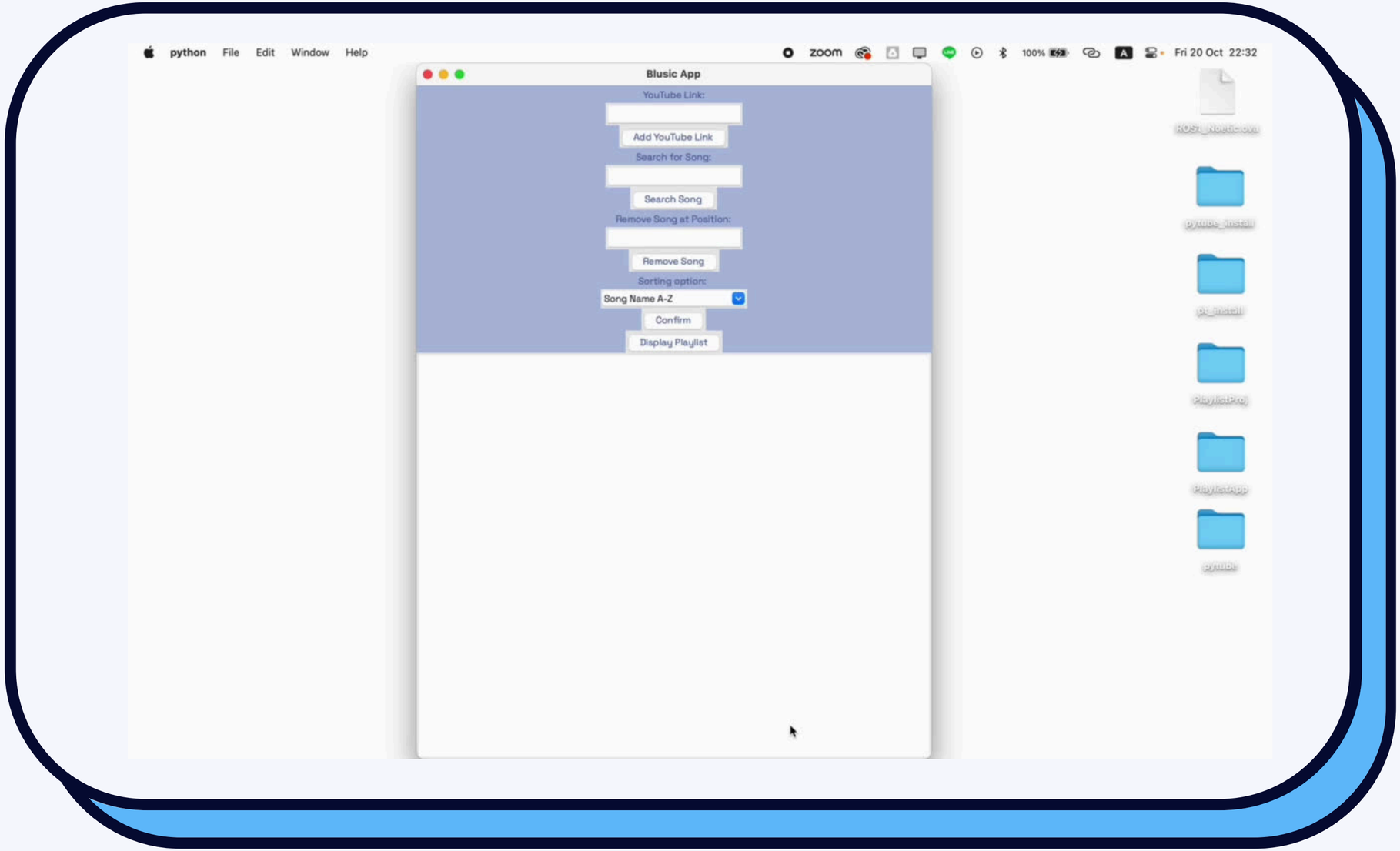
Guidelines for install Tkinter



Guidelines for install Pytube



Guidelines for Playlist App



Thank You

Music is the color for the world and everything in it

www.blusic.com