**Project Report**

**On**

**Adavnced Music Player**

**Subject-:**

**Presented By**

|  |  |  |
| --- | --- | --- |
| **Roll No** | **Batch-Sr. No** | **Name** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Signature of Internal Examiner**

**ABSTRACT**

it is advanced Music Player GUI application. It is used to play the all songs whatever in the your system

main adavantage of this application is automatically scan the all directories and find the .mp3 files

in this application I create 6 buttons like

1.play – it is used to play the songs

2.stop – it is used to stop the current song If you want to continue that song click play button

3.pause – it is used to pause the your song

4.un pause – it Is used to unpause the whatever you pause the song

5.previous – it is used to play the previous song after clicking this button

6.exit – it is used to exit this application with asking yes or no permisssion.

And also I create a listbox to display the all the songs line by line

if song is playing it contain some color like red remaining songs are contain green color.

If complete all songs then show the some pop up message to continue to listen or not

it is very useful to listen music without adding any songs because automatically take songs form your system.

note: I using os module to get my home path it is working for all systems but if you are in root directory to run this application you must add the your home path address

like,/home/rkvcseopencv/

**INDEX**

|  |  |  |
| --- | --- | --- |
| **Sr.No.** | **TITLE** | **PAGE NO** |
| **1** | **CHAPTER 1-INTRODUCTION**  **PYTHON**  Python Language Introduction  History of Python  Python Features  Python graphical user interfaces (GUIs)  **PYTHON TKINTER GUI**  Tkinter Widgets  Geometry Management | 4 - 10 |
| **2** | **CHAPTER-2 IMPLEMENTATION**  **Technologies used**  **Language used**  **CODE OF PROJECT** | 11 - 22 |
| **3** | **CHAPTER-3 SCREENSHOTS** | 23 - 25 |
| **4** | **CHAPTER-4 CONCLUSION** | 26 |

**CHAPTER 1-INTRODUCTION**

**PYTHON**

**Python Language Introduction**

[Python](https://www.geeksforgeeks.org/python-programming-language/) is a widely used general-purpose, high level programming language. It was initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation. It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code.

Python is a programming language that lets you work quickly and integrate systems more efficiently.

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

* **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
* **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
* **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

**History of Python**

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

**Python Features**

Python's features include −

* **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
* **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
* **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
* **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
* **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
* **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
* **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
* **Databases** − Python provides interfaces to all major commercial databases.
* **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
* **Scalable** − Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below −

* It supports functional and structured programming methods as well as OOP.
* It can be used as a scripting language or can be compiled to byte-code for building large applications.
* It provides very high-level dynamic data types and supports dynamic type checking.
* IT supports automatic garbage collection.
* It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

**Python graphical user interfaces (GUIs)**

* **Tkinter** − Tkinter is the Python interface to the Tk GUI toolkit shipped with Python. We would look this option in this chapter.
* **wxPython** − This is an open-source Python interface for wxWindows [http://wxpython.org](http://wxpython.org/).
* **JPython** − JPython is a Python port for Java which gives Python scripts seamless access to Java class libraries on the local machine [http://www.jython.org](http://www.jython.org/).

There are many other interfaces available, which you can find them on the net.

**PYTHON TKINTER GUI**

Tkinter Programming



Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps −

* Import the *Tkinter* module.
* Create the GUI application main window.
* Add one or more of the above-mentioned widgets to the GUI application.
* Enter the main event loop to take action against each event triggered by the user.

Example

#!/usr/bin/python

import tkinter

top = tkinter.Tk()

# Code to add widgets will go here...

top.mainloop()

This would create a following window −



**Tkinter Widgets**

Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets.

There are currently 15 types of widgets in Tkinter. We present these widgets as well as a brief description in the following table −

|  |  |
| --- | --- |
| **Sr.No.** | **Operator & Description** |
| 1 | [**Button**](https://www.tutorialspoint.com/python/tk_button.htm)  The Button widget is used to display buttons in your application. |
| 2 | [**Canvas**](https://www.tutorialspoint.com/python/tk_canvas.htm)  The Canvas widget is used to draw shapes, such as lines, ovals, polygons and rectangles, in your application. |
| 3 | [**Checkbutton**](https://www.tutorialspoint.com/python/tk_checkbutton.htm)  The Checkbutton widget is used to display a number of options as checkboxes. The user can select multiple options at a time. |
| 4 | [**Entry**](https://www.tutorialspoint.com/python/tk_entry.htm)  The Entry widget is used to display a single-line text field for accepting values from a user. |
| 5 | [**Frame**](https://www.tutorialspoint.com/python/tk_frame.htm)  The Frame widget is used as a container widget to organize other widgets. |
| 6 | [**Label**](https://www.tutorialspoint.com/python/tk_label.htm)  The Label widget is used to provide a single-line caption for other widgets. It can also contain images. |
| 7 | [**Listbox**](https://www.tutorialspoint.com/python/tk_listbox.htm)  The Listbox widget is used to provide a list of options to a user. |
| 8 | [**Menubutton**](https://www.tutorialspoint.com/python/tk_menubutton.htm)  The Menubutton widget is used to display menus in your application. |
| 9 | [**Menu**](https://www.tutorialspoint.com/python/tk_menu.htm)  The Menu widget is used to provide various commands to a user. These commands are contained inside Menubutton. |
| 10 | [**Message**](https://www.tutorialspoint.com/python/tk_message.htm)  The Message widget is used to display multiline text fields for accepting values from a user. |
| 11 | [**Radiobutton**](https://www.tutorialspoint.com/python/tk_radiobutton.htm)  The Radiobutton widget is used to display a number of options as radio buttons. The user can select only one option at a time. |
| 12 | [**Scale**](https://www.tutorialspoint.com/python/tk_scale.htm)  The Scale widget is used to provide a slider widget. |
| 13 | [**Scrollbar**](https://www.tutorialspoint.com/python/tk_scrollbar.htm)  The Scrollbar widget is used to add scrolling capability to various widgets, such as list boxes. |
| 14 | [**Text**](https://www.tutorialspoint.com/python/tk_text.htm)  The Text widget is used to display text in multiple lines. |
| 15 | [**Toplevel**](https://www.tutorialspoint.com/python/tk_toplevel.htm)  The Toplevel widget is used to provide a separate window container. |
| 16 | [**Spinbox**](https://www.tutorialspoint.com/python/tk_spinbox.htm)  The Spinbox widget is a variant of the standard Tkinter Entry widget, which can be used to select from a fixed number of values. |
| 17 | [**PanedWindow**](https://www.tutorialspoint.com/python/tk_panedwindow.htm)  A PanedWindow is a container widget that may contain any number of panes, arranged horizontally or vertically. |
| 18 | [**LabelFrame**](https://www.tutorialspoint.com/python/tk_labelframe.htm)  A labelframe is a simple container widget. Its primary purpose is to act as a spacer or container for complex window layouts. |
| 19 | [**tkMessageBox**](https://www.tutorialspoint.com/python/tk_messagebox.htm)  This module is used to display message boxes in your applications. |

**Geometry Management**

All Tkinter widgets have access to specific geometry management methods, which have the purpose of organizing widgets throughout the parent widget area. Tkinter exposes the following geometry manager classes: pack, grid, and place.

* [The *pack()* Method](https://www.tutorialspoint.com/python/tk_pack.htm) − This geometry manager organizes widgets in blocks before placing them in the parent widget.
* [The *grid()* Method](https://www.tutorialspoint.com/python/tk_grid.htm) − This geometry manager organizes widgets in a table-like structure in the parent widget.
* [The *place()* Method](https://www.tutorialspoint.com/python/tk_place.htm) − This geometry manager organizes widgets by placing them in a specific position in the parent widget.

**CHAPTER-2 IMPLEMENTATION**

**Technologies used** - Python 2.7 or 3.6

Python Tkinter GUI

**Language used** - Python

**CODE OF PROJECT**

import pygame

from tkinter import \*

import pygame.mixer

import os

import ntpath

from tkMessageBox import \*

pygame.init()

pygame.mixer.init()

root=Tk()

root.geometry("1600x800+0+0")

root.configure(background="powder blue")

Tops=Frame(root,width=1350,height=50,bd=8,bg="powder blue")

Tops.pack(side=TOP)

lblinfo=Label(Tops,font=('arial',40,'bold'),text="Advanced Music Player ",bd=10,fg="green")

lblinfo.grid(row=0,column=0)

list=[]

list5=[]

st=os.environ['HOME']+'/'

for rt, dirs, files in os.walk(st):#if u r in root change this path and replace with your home path

for file in files:

if file.endswith('.mp3'):

list.append(os.path.join(rt,file))

txtdisplay=Listbox(root,height=23,width=38,bd=16,font=('arial',15,'bold'),fg="green",bg="powder blue")

txtdisplay.place(relx=0.7,rely=0.5,anchor=CENTER)

i=0

for name in list:

list5.append(ntpath.basename(name))

txtdisplay.insert(END,list5[i])

i+=1

index=0

def next():

global index

index+=1

if index==len(list):

btn5['state']='disabled'

pygame.mixer.music.stop()

txtdisplay.itemconfig(index-1, {'fg': 'green'})

index=0

showinfo("FYI","your songs are over if u want listen once again press play button to continue?")

else:

txtdisplay.itemconfig(index-1, {'fg': 'green'})

txtdisplay.itemconfig(index, {'fg': 'red'})

pygame.mixer.music.load(list[index])

pygame.mixer.music.play()

def stop():

pygame.mixer.music.stop()

def pause():

pygame.mixer.music.pause()

def unpause():

pygame.mixer.music.unpause()

def previous():

global index

index-=1

if index==-1:

pygame.mixer.music.stop()

txtdisplay.itemconfig(index+1, {'fg': 'green'})

showinfo("FYI","your previous songs are over if u want play from starting press play button continue?")

index=0

else:

txtdisplay.itemconfig(index, {'fg': 'red'})

txtdisplay.itemconfig(index+1, {'fg': 'green'})

pygame.mixer.music.load(list[index])

pygame.mixer.music.play()

def play():

global index

btn5['state']='normal'

txtdisplay.itemconfig(index, {'fg': 'red'})

pygame.mixer.music.load(list[index])

pygame.mixer.music.play()

def exit():

exit=askyesno("music player","are u want exit click(yes/no)?")

if exit>0:

root.destroy()

return

btn6=Button(root,text="play",fg="powder blue",padx=10,pady=10,bd=4,width=10,bg="blue",font=('arial',15,'bold'),command=play)

btn6.place(relx=0.3,rely=0.3,anchor=CENTER)

btn5=Button(root,text="next",fg="powder blue",bg="blue",padx=10,pady=10,bd=4,width=10,state="normal",font=('arial',15,'bold'),command=next)

btn5.place(relx=0.3,rely=0.4,anchor=CENTER)

btn1=Button(root,text="stop",fg="powder blue",bg="red",padx=10,pady=10,bd=4,width=10,font=('arial',15,'bold'),command=stop)

btn1.place(relx=0.3,rely=0.5,anchor=CENTER)

btn2=Button(root,text="pause",fg="powder blue",bg="yellow",padx=10,pady=10,bd=4,width=10,font=('arial',15,'bold'),command=pause)

btn2.place(relx=0.3,rely=0.6,anchor=CENTER)

btn3=Button(root,text="unpause",fg="powder blue",bg="blue",padx=10,pady=10,bd=4,width=10,font=('arial',15,'bold'),command=unpause)

btn3.place(relx=0.3,rely=0.7,anchor=CENTER)

btn4=Button(root,text="previous",fg="powder blue",bg="blue",padx=10,pady=10,bd=4,width=10,font=('arial',15,'bold'),command=previous)

btn4.place(relx=0.3,rely=0.8,anchor=CENTER)

btn7=Button(root,text="Exit",fg="powder blue",bg="blue",padx=10,pady=10,bd=4,width=10,font=('arial',15,'bold'),command=exit)

btn7.place(relx=0.3,rely=0.8,anchor=CENTER)

root.mainloop()

**CHAPTER-3 SCREENSHOTS**

GUI – Main display window with name of Advanced Music Player

1.GUI of Advanced Music Player

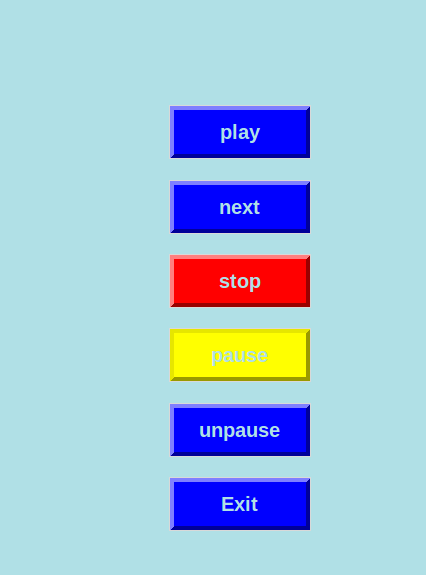


**it is a front end of the Advanced Music Player**

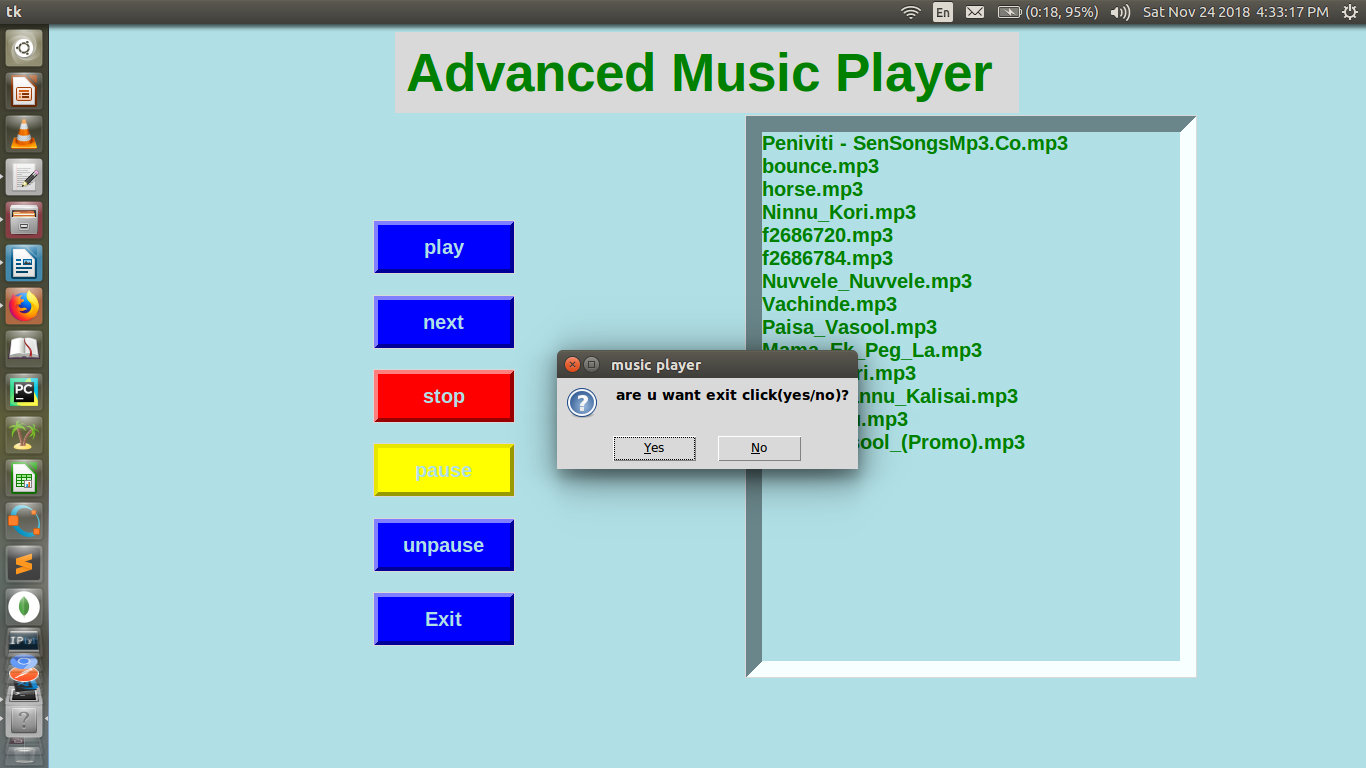
**2.show the current song with red color:-**



**3.Add Buttons**

****

**4.exit system**

****

**CHAPTER-4 CONCLUSION**

This project has really been faithful and informative. It has made us learn and understand the many trivial concepts of Python Language. As we have used python Tkinter as a GUI it provides various controls, such as buttons, labels and text boxes to build a user friendly application.

The fast growing use of internet confirms the good future and scope of the proposed project.

Finally it has taught us a valuable lifelong lesson about the improvements and working and interacting in a group.