MIHIR BIPIN CHAVAN 201080021 IT

Lab Project

Aim : Write a program to record the details of any employee working in your organization.

Theory :

Tkinter is the most commonly used library for developing GUI (Graphical User Interface) in Python. It is a standard Python interface to the Tk GUI toolkit shipped with Python. As Tk and Tkinter are available on most of the Unix platforms as well as on the Windows system, developing GUI applications with Tkinter becomes the fastest and easiest.

Tkinter widgets

There are various widgets like button, canvas, checkbutton, entry, etc. that are used to build the python GUI applications.

SN	Widget	Description
1	Button	The Button is used to add various kinds of buttons to the python application.
2	Canvas	The canvas widget is used to draw the canvas on the window.
3	Checkbutton	The Checkbutton is used to display the CheckButton on the window.
4	Entry	The entry widget is used to display the single-line text field to the user. It is commonly used to accept user values.
5	Frame	It can be defined as a container to which, another widget can be added and organized.
6	Label	A label is a text used to display some message or information about the other widgets.

7	ListBox	The ListBox widget is used to display a list of options to the user.
8	Menubutton	The Menubutton is used to display the menu items to the user.
9	Menu	It is used to add menu items to the user.
10	Message	The Message widget is used to display the message-box to the user.
11	Radiobutton	The Radiobutton is different from a checkbutton. Here, the user is provided with various options and the user can select only one option among them.
12	Scale	It is used to provide the slider to the user.
13	Scrollbar	It provides the scrollbar to the user so that the user can scroll the window up and down.
14	Text	It is different from Entry because it provides a multi-line text field to the user so that the user can write the text and edit the text inside it.
14	Toplevel	It is used to create a separate window container.
15	Spinbox	It is an entry widget used to select from options of values.
16	PanedWindow	It is like a container widget that contains horizontal or vertical panes.
17	LabelFrame	A LabelFrame is a container widget that acts as the container

18	MessageBox	This module is used to display the message-box in the desktop based applications.

Python Tkinter Geometry

The Tkinter geometry specifies the method by using which, the widgets are represented on display. The python Tkinter provides the following geometry methods.

1. The pack() method
2. The grid() method
3. The place() method

Let's discuss each one of them in detail.

Python Tkinter pack() method

The pack() widget is used to organize widget in the block. The positions widgets added to the python application using the pack() method can be controlled by using the various options specified in the method call.

However, the controls are less and widgets are generally added in the less organized manner.

The syntax to use the pack() is given below.

syntax

1. widget.pack(options)

A list of possible options that can be passed in pack() is given below.

- **expand:** If the expand is set to true, the widget expands to fill any space.
- **Fill:** By default, the fill is set to NONE. However, we can set it to X or Y to determine whether the widget contains any extra space.
- **size:** it represents the side of the parent to which the widget is to be placed on the window.

Python Tkinter grid() method

The grid() geometry manager organizes the widgets in the tabular form. We can specify the rows and columns as the options in the method call. We can also specify the column span (width) or rowspan(height) of a widget.

This is a more organized way to place the widgets to the python application. The syntax to use the grid() is given below.

Syntax

1. widget.grid(options)

A list of possible options that can be passed inside the grid() method is given below.

- **Column**
  The column number in which the widget is to be placed. The leftmost column is represented by 0.
- **Columnspan**
  The width of the widget. It represents the number of columns up to which, the column is expanded.
- **ipadx, ipady**
  It represents the number of pixels to pad the widget inside the widget's border.
- **padx, pady**
  It represents the number of pixels to pad the widget outside the widget's border.
- **row**
  The row number in which the widget is to be placed. The topmost row is represented by 0.
- **rowspan**
  The height of the widget, i.e. the number of the row up to which the widget is expanded.
- **Sticky**
  If the cell is larger than a widget, then sticky is used to specify the position of the widget inside the cell. It may be the concatenation of the sticky letters representing the position of the widget. It may be N, E, W, S, NE, NW, NS, EW, ES.

Python Tkinter place() method

The place() geometry manager organizes the widgets to the specific x and y coordinates.

Syntax

1. widget.place(options)

The os Module

The **os** module has many uses.

os.startfile()

The **os.startfile()** method allows us to "start" a file with its associated program. In other words, we can open a file with it's associated program, just like when you double-click a PDF and it opens in Adobe Reader. Let's give it a try!

```
>>> os.startfile(r'C:\Users\mike\Documents\labels.pdf')
```

In the example above, I pass a fully qualified path to **os.startfile** that tells it to open a file called **labels.pdf**. On my machine, this will open the PDF in Adobe Reader. You should try opening your own PDFs, MP3s, and photos using this method to see how it works.

Python tkinter messagebox

- Messagebox is used to display pop-up messages.
- To get started with message box import a library messagebox in Python.
- Messagebox provides mainly 6 types of message prompts like showinfo(), showerror(), showwarning(), askquestion(), askokcancel(), askyesno(), askretyrcancel().

showinfo()	used when any confirmation/information needs to display. like login successful, message sent, etc.
showerror()	used to display error prompt with sound. The ideal time for its appearance is when the user makes any mistakes or skips the necessary step.
showwarning()	Shows warning prompt with exclamation sign. It warns the user about the upcoming errors.

askquestion()	It asks the user for **Yes** or **No**. also it returns '**Yes**' or '**No**'
askokcancel()	It prompts for '**Ok**' or '**Cancel**', returns '**True**' or '**False**'
askyesno()	It prompts for '**Yes**' or '**No**'. Returns **True** for 'Yes' and **False** for 'No'
askyesnocancel()	It prompts for '**Yes**', '**No**', or '**Cancel**'. Yes returns True, No returns False, and Cancel returns None
askretrycancel()	It prompts with **Retry** and **Cancel** options. Retry returns **True** and Cancel returns **False.**

ttk in Tkinter

tkinter.ttk is a module that is used to style the tkinter widgets. Just like CSS is used to style an HTML element, we use tkinter.ttk to style tkinter widgets.

Here are the major differences between **tkinter widget and tkinter.ttk** −

- Tkinter widgets are used to add Buttons, Labels, Text, ScrollBar, etc., however, tkinter.ttk supports a variety of widgets as compared to tkinter widgets.
- Tkinter.ttk doesn't support Place, Pack() and Grid(), thus it is recommended to use tkinter widget with ttk.
- Ttk has many features and configurations that extend the functionality of a native application and make it look more modern.
- Tkinter widget is a native widget in the tkinter library, however ttk is a themed module.
- To override the basic Tk widget in tkinter, use "from tkinter.ttk import *"

Code :

```
#MIHIR CHAVAN 201080021 IT
from tkinter import *
from tkinter import ttk,messagebox
```

```
import os #To use startfile() function
root=Tk() #Tk class is used to create a root window
root.title('Employee management system')#Title of tkinter window
root.geometry('1280x720')
bg_color='#3333FF'#background Color in Hexcode format
#======================Variable====================#
ref_var=IntVar() #variable for employee ref number
name_var=StringVar()#variable for employee name
gender_var=StringVar()
email_var=StringVar()
salary_var=IntVar()
phone_var=IntVar()
desi_var=StringVar()
#======================Functions====================#
def add(): #definition of add button/function
    if ref_var.get()==0 or name_var.get()=='' or salary_var.get()=='' or gender_var=='' or email_var=='' or desi_var=='' or phone_var='':
        messagebox.showerror('Error','All fields are required') #Will show error if all fields are not filled
    else:
        textarea.delete(1.0,END)
        textarea.insert(END, '\n===========================================')
        textarea.insert(END,f'Employee Ref\t\t\t\t{ref_var.get()}')
        textarea.insert(END, '\n===========================================')
        textarea.insert(END,f'\nEmployee Name\t\t\t\t{name_var.get()}')
        textarea.insert(END,f'\nEmail Id\t\t\t\t{email_var.get()}')
        textarea.insert(END,f'\nGender\t\t\t\t{gender_var.get()}')
        textarea.insert(END,f'\nDesignation\t\t\t\t{desi_var.get()}')
        textarea.insert(END,f'\nSalary\t\t\t\t{salary_var.get()}')
        textarea.insert(END,f'\nAddress\t\t\t\t{txt_add.get(1.0,END)}')

def save(): #definition of save button/function
    data=textarea.get(1.0,END)
    f1=open('D:\Mihir docs C to d Drive\Desktop Files\Python vs code\Pdl_Proj\\'+str(ref_var.get())+'.txt','w')
```

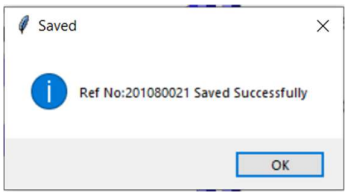
```
    f1.write(data) #File will be saved only if correct address of folder in which it is to be saved is provide which will be different for all PCs
    f1.close()
    messagebox.showinfo('Saved',f'Ref No:{ref_var.get()} Saved Successfully')


def print(): #definition of Print button/function
    data=textarea.get(1.0,END)
    f='D:\\Mihir docs C to d Drive\\Desktop Files\\Python vs code\\PdI_Proj\\'+str(ref_var.get())+'.txt'
    os.startfile(f,'Print') #File will be printed only if correct address of folder in which it is to be printed is provide which will be different for all PCs
    #The os.startfile() method allows us to “start” a file with its associated program.
def reset(): #definition of reset button/function
    textarea.delete(1.0,END)
    txt_add.delete(1.0,END)
    ref_var.set(0)
    name_var.set('')
    gender_var.set('')
    desi_var.set('')
    phone_var.set('')
    email_var.set('')
    salary_var.set('')


def Exit(): #definition of exit button/function
    if messagebox.askyesno('Exit','Do you really want to exit'):
        root.destroy()
#======================Heading====================#
title=Label(root,text='Employee Record System',bg=bg_color,fg='white',font=('times new rommon',35,'bold'),relief=GROOVE,bd=12)
title.pack(fill=X) #Accomaodates itself on x axis
#Title of inner window/output page
#======================Left Frame==================#
F1=Frame(root,bg=bg_color,relief=RIDGE,bd=15) #Information of 1st frame on left side
F1.place(x=10,y=80,width=650,height=530)
#Creating Labels and text boxes/drop down menus to fill details in left frame
lbl_ref=Label(F1,text='Employee Reference',font=('times new rommon',20,'bold'),fg='black',bg=bg_color)
```

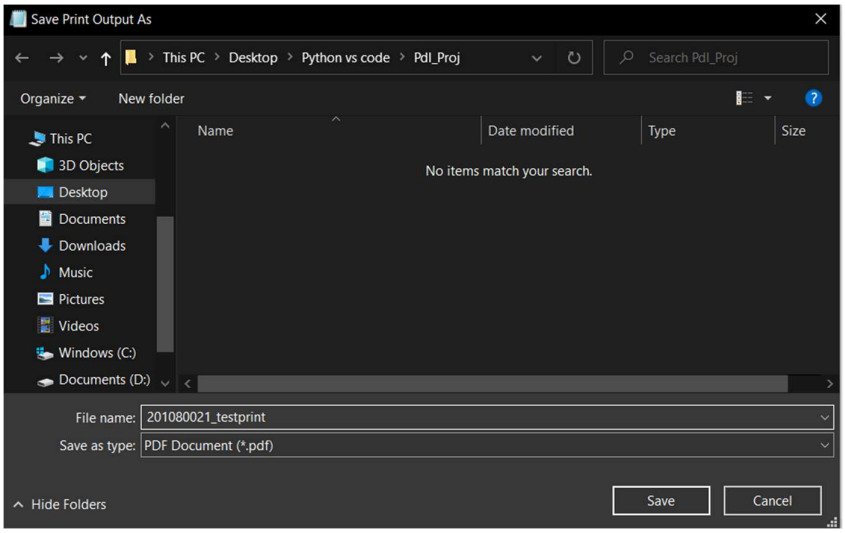
```
lbl_ref.grid(row=0,column=0,padx=30,pady=10)

txt_ref=Entry(F1,font=('times new
rommon',18,'bold'),relief=RIDGE,bd=7,textvariable=ref_var)#Creating text boxes

txt_ref.grid(row=0,column=1,pady=10,sticky='w')


lbl_name=Label(F1,text='Employee Name',font=('times new
rommon',20,'bold'),fg='black',bg=bg_color)

lbl_name.grid(row=1,column=0,padx=30,pady=10)

txt_name=Entry(F1,font=('times new rommon',18,'bold'),relief=RIDGE,bd=7,textvariable=name_var)

txt_name.grid(row=1,column=1,pady=10,sticky='w')


lbl_email=Label(F1,text='Employee Email',font=('times new rommon',20,'bold'),fg='black',bg=bg_color)

lbl_email.grid(row=2,column=0,padx=30,pady=10)

txt_email=Entry(F1,font=('times new rommon',18,'bold'),relief=RIDGE,bd=7,textvariable=email_var)

txt_email.grid(row=2,column=1,pady=10,sticky='w')


lbl_gender=Label(F1,text='Gender',font=('times new rommon',20,'bold'),fg='black',bg=bg_color)

lbl_gender.grid(row=3,column=0,padx=30,pady=10)

combo_gender=ttk.Combobox(F1,font=('times new
rommon',18),state='readonly',textvariable=gender_var)#state is 'readonly' to avoid random user input

combo_gender['value']=('Male','Female') #Creating drop down menu to choose selected options only

combo_gender.grid(row=3,column=1,pady=10)


lbl_des=Label(F1,text='Designation',font=('times new rommon',20,'bold'),fg='black',bg=bg_color)

lbl_des.grid(row=4,column=0,padx=30,pady=10)

combo_des=ttk.Combobox(F1,font=('times new rommon',18),state='readonly',textvariable=desi_var)

combo_des['value']=('Hr','IT','Sales','Marketing','Finanace','Quality Check','R&D')

combo_des.grid(row=4,column=1,pady=10)


lbl_no=Label(F1,text='Mobile Number',font=('times new rommon',20,'bold'),fg='black',bg=bg_color)

lbl_no.grid(row=5,column=0,padx=30,pady=10)

txt_no=Entry(F1,font=('times new rommon',18,'bold'),relief=RIDGE,bd=7,textvariable=phone_var)

txt_no.grid(row=5,column=1,pady=10,sticky='w')


lbl_s=Label(F1,text='Salary',font=('times new rommon',20,'bold'),fg='black',bg=bg_color)
```

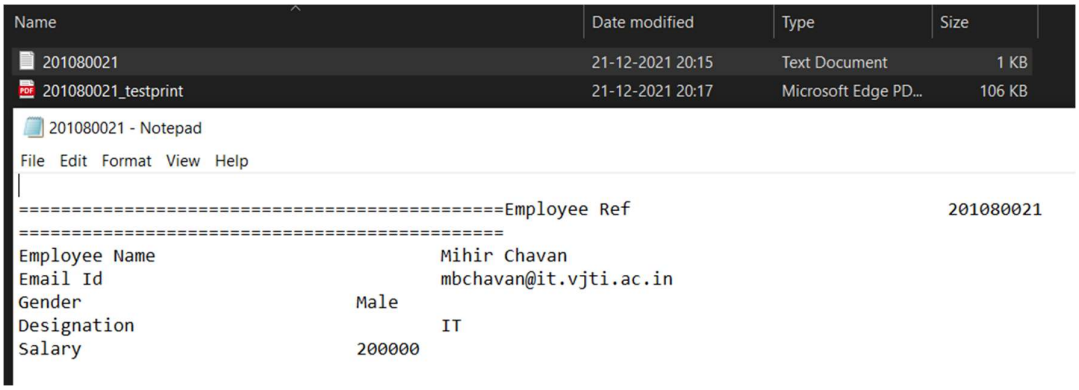
```
lbl_s.grid(row=6,column=0,padx=30,pady=10)

txt_s=Entry(F1,font=('times new rommon',18,'bold'),relief=RIDGE,bd=7,textvariable=salary_var)

txt_s.grid(row=6,column=1,pady=10,sticky='w')


lbl_add=Label(F1,text='Address',font=('times new rommon',20,'bold'),fg='black',bg=bg_color)

lbl_add.grid(row=7,column=0,padx=30,pady=10)

txt_add=Entry(F1,font=('times new rommon',18,'bold'),relief=RIDGE,bd=7)

txt_add.grid(row=7,column=1,pady=10,sticky='w')

#======================Right Frame==================#

F2=Frame(root,bg=bg_color,relief=RIDGE,bd=15) #Information of 2nd frame on right side

F2.place(x=665,y=80,width=610,height=530)


lbl_t=Label(F2,text='Employee Details',font=('arial',20,'bold'),fg='black',bd=7,relief=GROOVE)#Label of right frame

lbl_t.pack(fill=X)

scroll=Scrollbar(F2,orient=VERTICAL)#Creating a scroll bar for right frame

scroll.pack(side=RIGHT,fill=Y)

textarea=Text(F2,font='arial 15',yscrollcommand=scroll.set)

textarea.pack(fill=BOTH)

scroll.config(command=textarea.yview)

#======================Buttons==================#

F3=Frame(root,bg=bg_color,relief=RIDGE,bd=15) #creating 3rd frame at the bottom

F3.place(x=10,y=615,width=1260,height=100)

btn1=Button(F3,text='Add Record',font='aril 20 bold',bg='yellow',fg='crimson',command=add)#Add Record button created

btn1.grid(row=0,column=0,padx=50,pady=7)

btn2=Button(F3,text='Save',font='aril 20 bold',bg='yellow',fg='crimson',command=save) #Save button created

btn2.grid(row=0,column=1,padx=50,pady=7)

btn3=Button(F3,text='Print',font='aril 20 bold',bg='yellow',fg='crimson',command=print)#Print button created

btn3.grid(row=0,column=2,padx=50,pady=7)

btn4=Button(F3,text='Reset',font='aril 20 bold',bg='yellow',fg='crimson',command=reset) #Reset button created

btn4.grid(row=0,column=3,padx=50,pady=7)
```

```
btn5=Button(F3,text='Exit',font='aril 20 bold',bg='yellow',fg='crimson',command=Exit) #Exit button created

btn5.grid(row=0,column=4,padx=50,pady=7)

root.mainloop()
```

Output:







Conclusion :

1. All the functions of module tkinter are imported.
2. Also, the ttk and messagebox functions are imported from module tkinter separately.
3. The module 'os' is imported to use the function startfile() which allows to open a file associated with program.
4. Tk class is used to create a root window.
5. Background color of window is mentioned in hexcode format.
6. Variables like ref_var,name_var,gender_var etc are created to store the input values of ref no.,name,gender etc.
7. Then,Functions like add,save,print,reset and exit are created and defined to use the respective buttons when clicked by user.
8. In add function,if all the fields are not filled, error message is shown using messagebox function, else the user input info is stored in respective text boxes.
9. In save function, the whole textarea of right frame is read and saved in specified location.
10. In print function, the saved file is converted to pdf format and saved at mention location.
11. In reset function the info added in right frame is deleted and in exit function, the window is closed/destroyed.
12. In left frame(F1), various labels and textboxes are created to take the user input info.
13. Also, dropdown menus are created using combo widget of tkinter.
14. In right frame(F2), a textarea and scrollbar is created using Scrollbar and text widgets.
15. Then, various buttons like add record, save etc are created to use the respective functions.
16. Then, the window is closed usimg root.mainloop().
17. Thus, we have successfully created an employee management/record system in python.