Python Programming

Day 14: Graphical User Interface (2)

Graphical User Interface

GUI Library

Widget

Menu

Toolbar

Color and symbol meaning

❗	Hint
✓	Preferred
❓	Student's activity
☐	Practice code

🟨	Keyword
🟪	In-built functions
🟩	Strings
🟦	Output

Tkinter – Spinbox Widget

The **Spinbox** widget is a variant of the standard Tkinter Entry widget, which can be used to select from a fixed number of values.

Syntax

Here is the simple syntax to create this widget

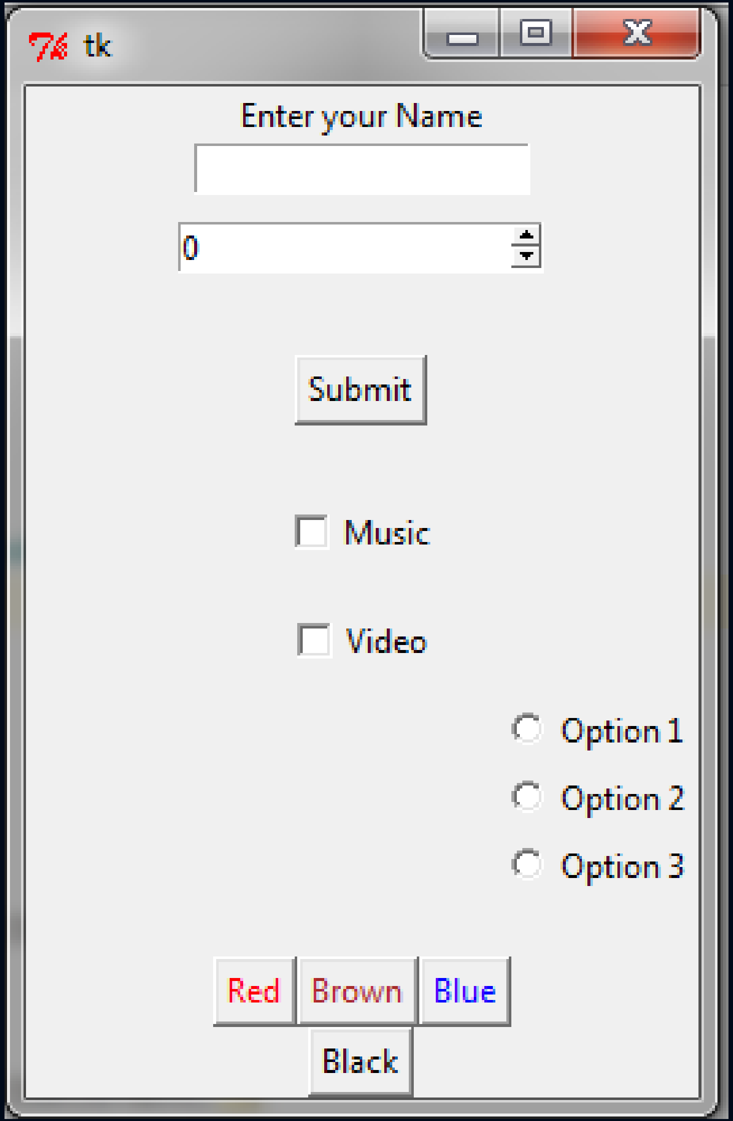
w = Spinbox(master, option, ...)

Tkinter – Spinbox Widget

Place the code in the existing GUI script.

```
# create spinbox widget

w = tkinter.Spinbox(window, from_=0, to=10)

w.pack({'side':'top', 'pady':10})
```

Output



Tkinter – Spinbox Options

It has similar options as entry widget, below are peculiar options

Option	Description
from_	The minimum value. Used together with “to” to limit the spinbox range.
to	limit the spinbox range.
validate	Validation mode. Default is NONE.
validatecommand	Validation callback. No default value.
values	A tuple containing valid values for this widget. Overrides from/to/increment.
repeatdelay	Together with repeatinterval, this option controls button auto-repeat. Both values are given in milliseconds.
repeatinterval	See repeatdelay.

Tkinter – MessageBox Widget

The **tkMessageBox** module is used to display message boxes in your applications. This module provides a number of functions that you can use to display an appropriate message.

Some of these functions are **showinfo**, **showwarning**, **showerror**, **askquestion**, **askokcancel**, **askyesno**, and **askretryignore**.

Syntax

Here is the simple syntax to create this widget

from tkinter import messagebox
messageBox.FunctionName(title, message [, options])

Tkinter – MessageBox Widget

Parameters

- ❑ **FunctionName** – This is the name of the appropriate message box function.

- ❑ **title** – This is the text to be displayed in the title bar of a message box.

- ❑ **message** – This is the text to be displayed as a message.

- ❑ **options** – options are **alternative choices** that you may use to tailor a standard message box.

Tkinter – MessageBox Widget

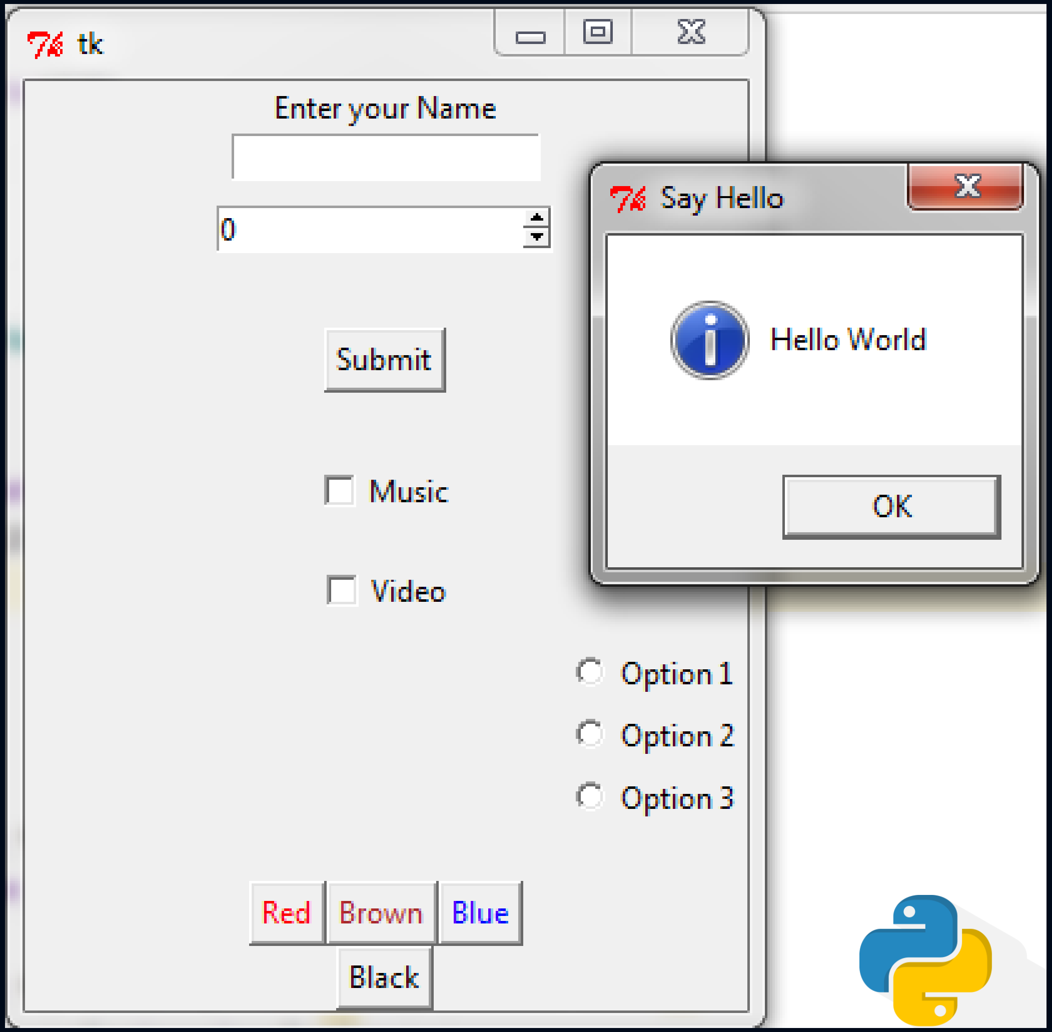
You could use one of the following functions with dialogue box :

- ❑ showinfo()
- ❑ showwarning()
- ❑ showerror ()
- ❑ askquestion()
- ❑ askokcancel()
- ❑ askyesno ()
- ❑ askretrycancel ()

Tkinter – MessageBox Widget

```
# create messageBox widget
def submitForm():
  messagebox.showinfo("Say Hello",
"Hello World")

#create button widget
btnSubmit = tkinter.Button(window,
text='Submit', command= submitForm)
btnSubmit.pack({'side':'top', 'pady':20})
```

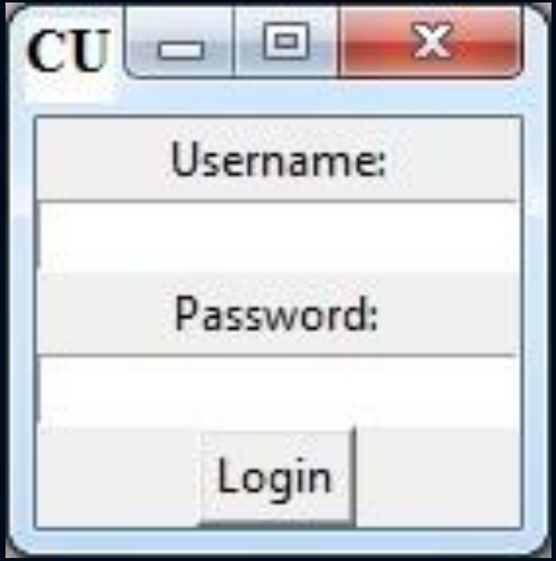
Output

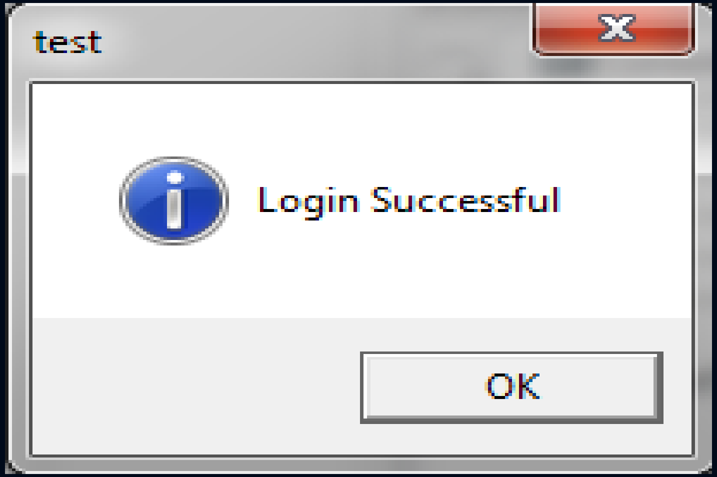


Hello World

Class Activity 1

Use widget discussed above to create login page to an application. Similar to the figure below





Login Successful

Tkinter – ListBoxes Widget

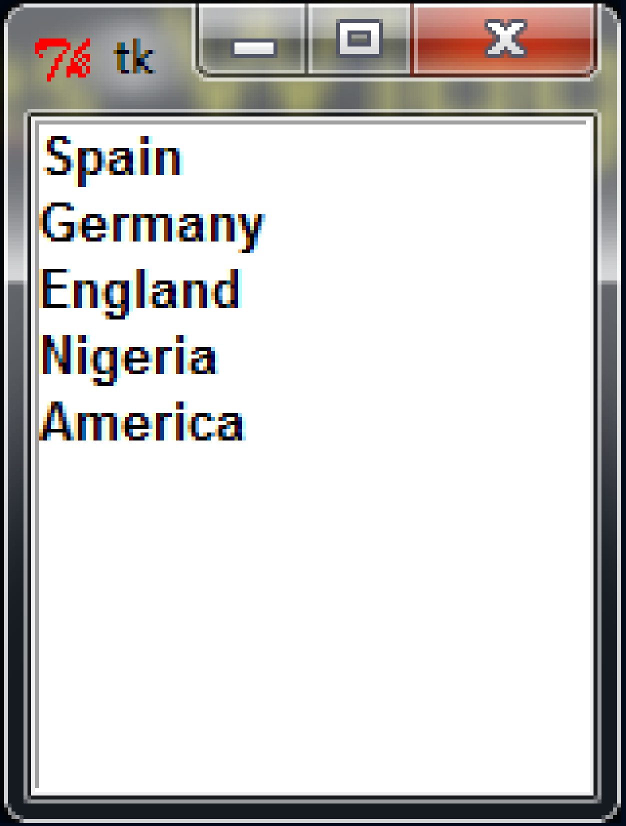
The **Listbox** is a control that allows single and multiple selections between various items.

The Following script defines a Listbox and reads country names from an array, and then inserts these names into the listbox.

Tkinter – ListBoxes Widget

```
from tkinter import *
window = Tk()
lstBox1 = Listbox(window, width = 20,
font = 'Arial 10 bold')
countries =
['Spain','Germany','England','Nigeria','Ame
rica']
lstBox1.pack()
for i in countries:
   lstBox1.insert(lstBox1.size(), i)

window.mainloop()
```

Output



```
Spain
Germany
England
Nigeria
America
```

Tkinter – Scale Widget

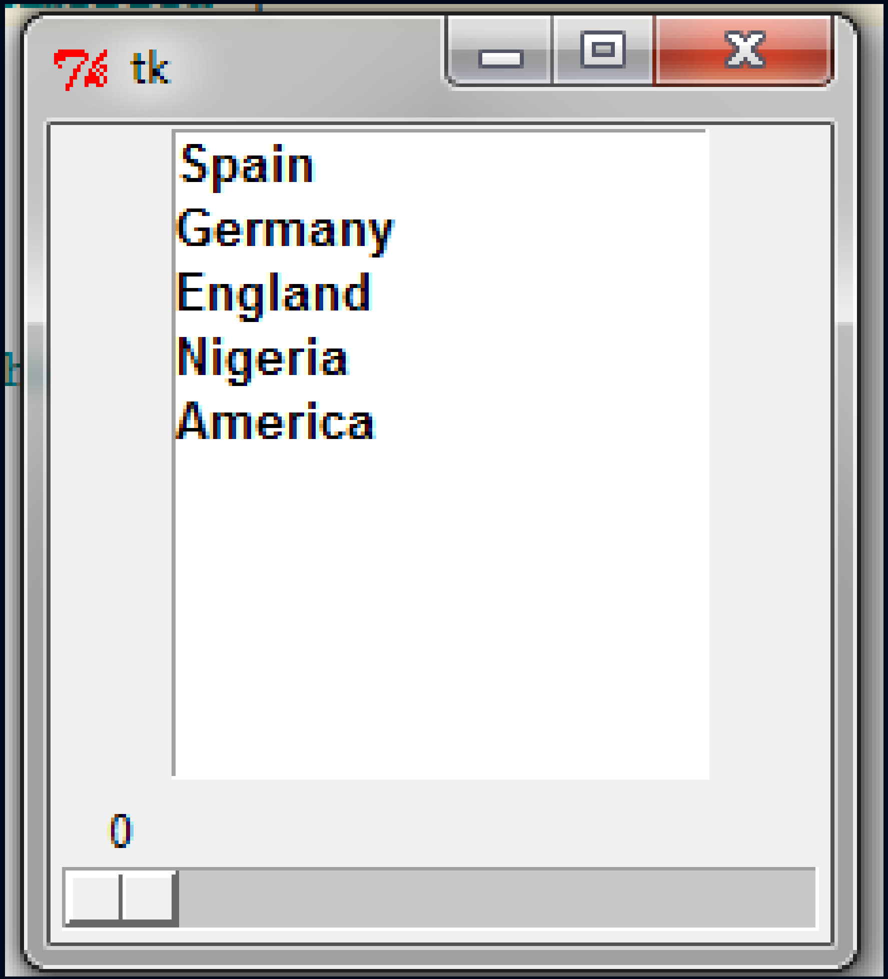
Scale (Slide Bar) control allows the user to graphically choose a value from scale by sliding the bar.

The following script creates a horizontal slide bar that ranges from 0 to 100.

Tkinter – Scale Widget

```
sca1 = Scale(window, from_
= 0, to = 100, orient =
'horizontal', length = 200)

sca1.pack()
```

Output



```
Spain
Germany
England
Nigeria
America
```

```
0
```

Tkinter – Combobox Widget

The **Combobox** is a dropdown list that provides a graphical way for the user to select one value of the list.

To use the Combobox control, we need to import the **tkinter.ttk** module.

The following script creates a combobox that contains the names of the weekdays.

Sample Code

```
# code includes combobox widget
from tkinter import *
from tkinter import ttk
window = Tk()
weekdays =
('Sunday','Monday','Tuesday','Wednesday','T
hursday','Friday','Saturday')
cmbWeekdays = ttk.Combobox(window,
values=weekdays)
cmbWeekdays.pack()
window.mainloop()
```

Class Activity 2

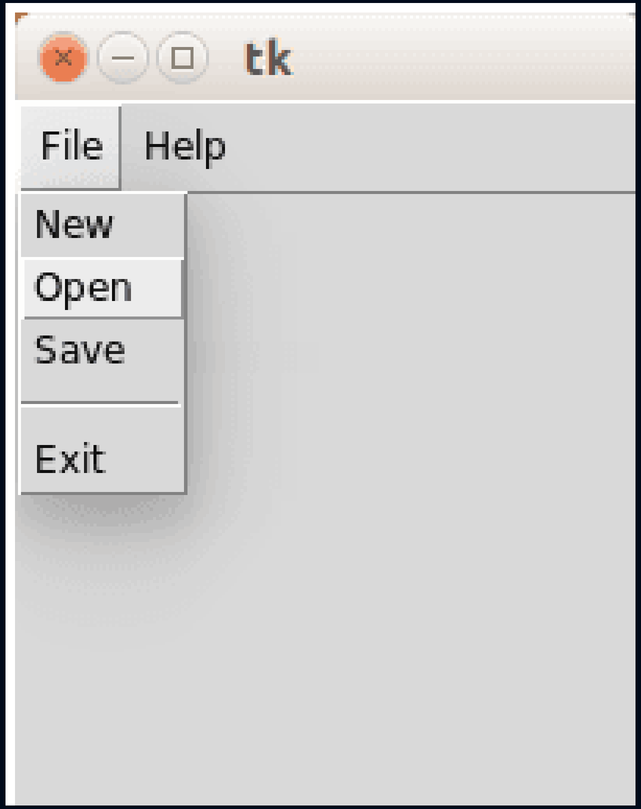
Create a python GUI program that populates a Listbox widget using an entry widget.

Tkinter – Menu Widget

A **menubar** is one of the most visible parts of the GUI application. It is a group of commands located in various menus.

Menus group commands that we can use in an application. **Toolbars** provide a quick access to the most frequently used commands.

The screenshot below demonstrates a Tkinter based menu.



Tkinter – Menu Widget

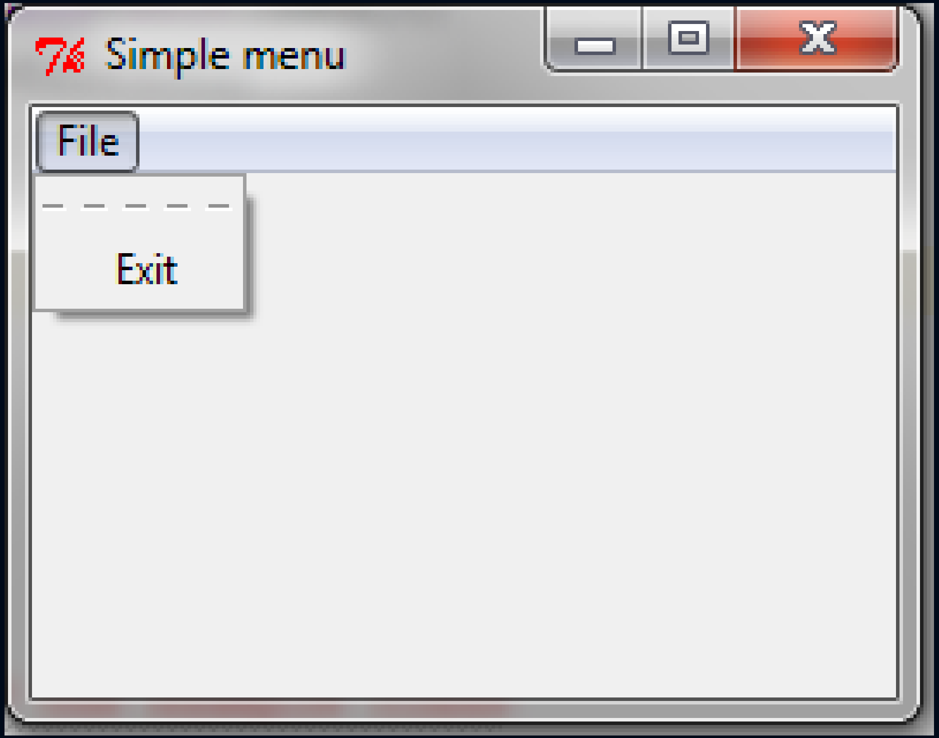
```
from tkinter import Tk, Frame, Menu

#the Example class inherits from the Frame class which is a container
#the class Constructs a frame widget with the parent MASTER.
class Example(Frame):
   def __init__(self, parent):
      Frame.__init__(self, parent)

      self.parent = parent
      self.initUI() #initializes the menubar

   def initUI(self):
      self.parent.title("Simple menu")

      menubar = Menu(self.parent)
      self.parent.config(menu=menubar)

      fileMenu = Menu(menubar)
      fileMenu.add_command(label="Exit", command=self.onExit)
      menubar.add_cascade(label="File", menu=fileMenu)

   def onExit(self):
      self.quit()
```

The **initUI()** method binds all other widgets to the parent widget.

Tkinter – Menu Widget

```
def main():
   root = Tk()
   root.geometry("250x150+300+300")
   app = Example(root) #creates an
instance of the Example class
   root.mainloop()



if __name__ == '__main__':
   main()
```

Output



Tkinter – Menu Widget

Code Description

```
menubar = Menu(self.parent)
self.parent.config(menu=menubar)
```

Here we create a menubar. It is a regular Menu widget configured to be the menubar of the root window.

```
fileMenu = Menu(menubar)
```

We create a file menu object. A menu is a drop-down window containing commands.

```
fileMenu.add_command(label="Exit",
command=self.onExit)
```

We add a command to the file menu. The command will call the onExit() method.

```
menubar.add_cascade(label="File",
menu=fileMenu)
```

The file menu is added to the menubar using the add_cascade() method.

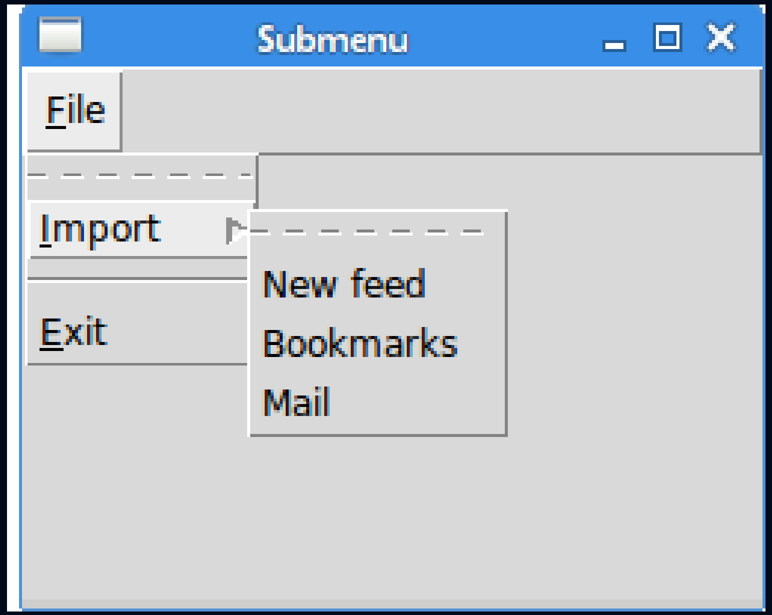
Class Activity 3

Create a python GUI program with 4 menus (File, Sales, Purchases and Help).

Tkinter – Submenu Widget

A **submenu** is a menu plugged into another menu object. The next example demonstrates this.

The screenshot below demonstrates a Tkinter based submenu.



Tkinter – Submenu Widget

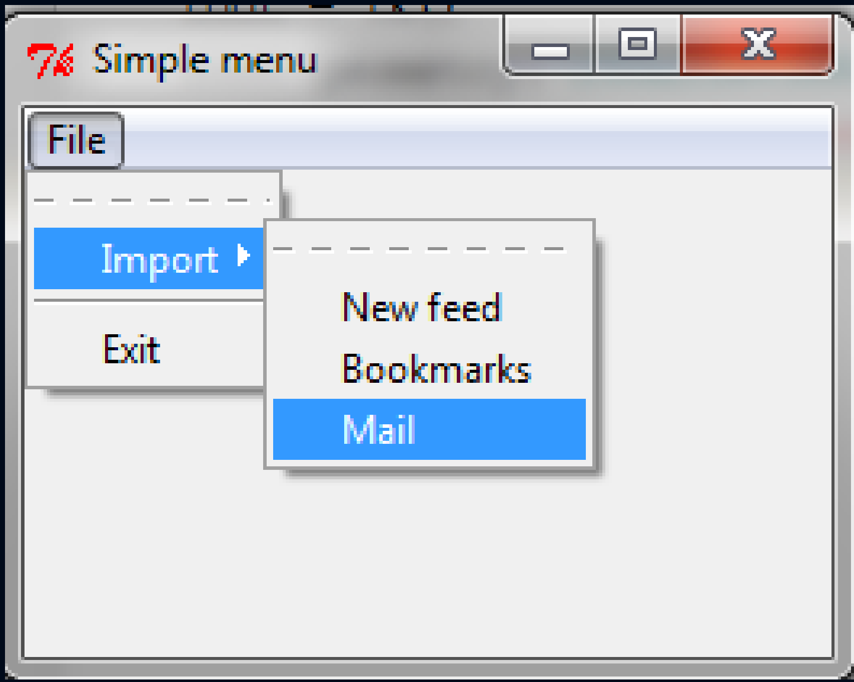
```
from tkinter import Tk, Frame, Menu
#the Example class inherits from the Frame class which is a container
#the class Constructs a frame widget with the parent MASTER.
class Example(Frame):
  def __init__(self, parent):
    Frame.__init__(self, parent)

    self.parent = parent
    self.initUI() #initializes the menubar

  def initUI(self):
    self.parent.title("Simple menu")

    menubar = Menu(self.parent)
    self.parent.config(menu=menubar)

    fileMenu = Menu(menubar)
```

In the sample code, we have three options in a submenu of a file menu. We create a separator and keyboard shortcuts.

Tkinter – Submenu Widget

```
        submenu = Menu(fileMenu)
        submenu.add_command(label="New feed")
        submenu.add_command(label="Bookmarks")
        submenu.add_command(label="Mail")
        fileMenu.add_cascade(label='Import', menu=submenu, underline=0)

        fileMenu.add_separator()

        fileMenu.add_command(label="Exit", command=self.onExit)
        menubar.add_cascade(label="File", menu=fileMenu)

    def onExit(self):
        self.quit()

def main():
    root = Tk()
    root.geometry("250x150+300+300")
    app = Example(root) #creates an instance of the Example class
    root.mainloop()

if __name__ == '__main__':
    main()
```

Output



Tkinter – Submenu Widget *Code Description*

```
submenu = Menu(fileMenu)
submenu.add_command(label="New feed")
submenu.add_command(label="Bookmarks")
submenu.add_command(label="Mail")
```

We have a submenu with three commands. The submenu is a regular menu.

```
fileMenu.add_separator()
```

A separator is a horizontal line that visually separates menu commands. This way we can group items into some logical places.

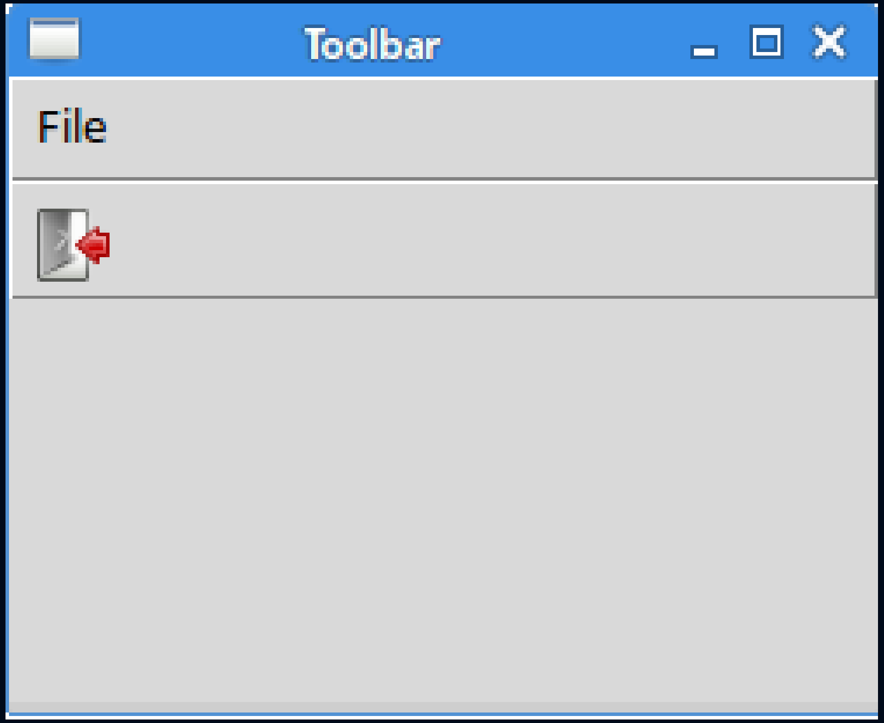
```
fileMenu.add_cascade(label='Import', menu=submenu, underline=0)
```

By adding the menu to the fileMenu and not to the menubar, we create a submenu. The underline parameter creates a keyboard shortcut. It provides the character position which should be underlined. In our case it is the first. Positions start from zero. When we click on the File menu, a popup window is shown. The Import menu has one character underlined. We can select it either with the mouse pointer or with the Alt+I shortcut.

Toolbar Widget

The screenshot below demonstrates a Tkinter based menu with Toolbar.

Toolbars provide a quick access to the most frequently used commands. There is no toolbar widget in Tkinter.



Toolbar Widget

```
from PIL import Image, ImageTk
from tkinter import Tk, Frame, Menu
from tkinter import Button, LEFT, TOP, X, FLAT, RAISED
#the Example class is modified to include toolbar widget

class Example(Frame):
    def __init__(self, parent):
        Frame.__init__(self, parent)

        self.parent = parent
        self.initUI() #initializes the menubar

    def initUI(self):
        self.parent.title("Toolbar")

        menubar = Menu(self.parent)
        self.fileMenu = Menu(self.parent, tearoff=0)
        self.fileMenu.add_command(label="Exit", command=self.onExit)
        menubar.add_cascade(label="File", menu=self.fileMenu)

        toolbar = Frame(self.parent, bd=1, relief=RAISED)
```

You must install the "pillow" library to work with images in python.

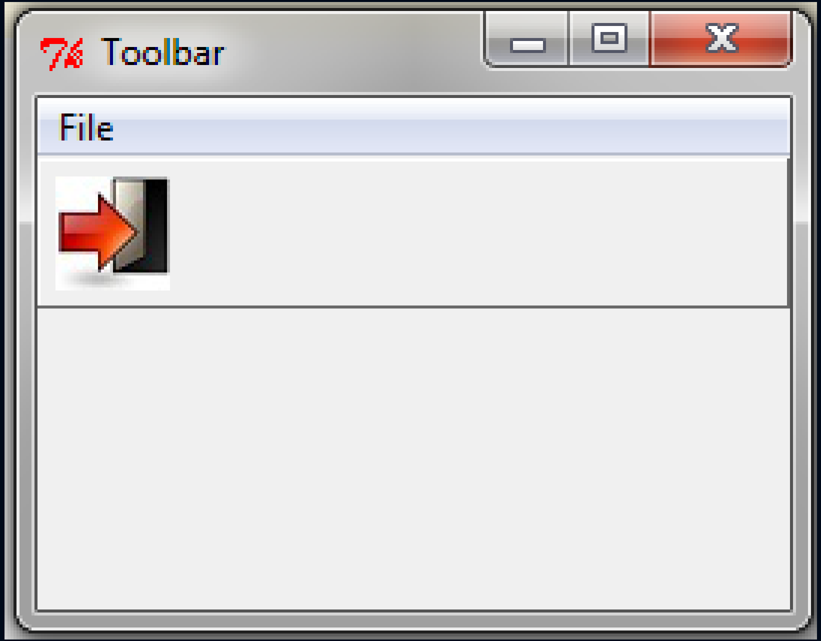
Procedure:

- **From settings-> Project-> project interpreter**
- **Click on the + symbol by the right**
- **Search for pillow in the available package form and install**

Toolbar Widget

```
    self.img = Image.open("exit1.jpg")
    eimg = ImageTk.PhotoImage(self.img)

    exitButton = Button(toolbar, image=eimg, relief=FLAT,
              command=self.quit)
    exitButton.image = eimg
    exitButton.pack(side=LEFT, padx=2, pady=2)

    toolbar.pack(side=TOP, fill=X)
    self.parent.config(menu=menubar)
    self.pack()

  def onExit(self):
    self.quit()

def main():
  root = Tk()
  root.geometry("250x150+300+300")
  app = Example(root) #creates an instance of the Example class
  root.mainloop()

if __name__ == '__main__':
  main()
```

Output



Class Activity 3

Create a python GUI program that store sales order (Quantity, Item name & price) in a listbox and calculate the total amount of items entered into the list box

Handle all possible exceptions.

Next Lecture ...

Python Programming

Day 15: *Introduction to Database*