Python: GUI Design

-There are many GUI design libraries available in Python such as Tkinter, Qt, Pyglet, Kivy etc.
-One of the popular and easy to use is Tkinter → it provides easily configurable classes for creating buttons, labels, text boxes, list boxes, drop downs, radio buttons, check boxes, menus, modal and modeless dialogs etc.
-To create a label in Tkinter, the typical code looks as:

```
l1 = Label(self, text="hello", foreground = "#ff0000", background = "light blue",
font = "Arial 9")  # Arial 12 bold italic
l1.place(x=100, y=200)
```

-The above code will create a label with text color of red and background color of light blue, and display hello in it
-Note that the origin in a GUI container starts in the top left hand corner →Positive X-axis moves to the right and positive Y-axis move vertically down

-Similarly, to create a textbox, the class is called Entry e.g.

```
        self.txtAmount = Entry()
        self.txtAmount.place(x=200, y = 200, width=70)
```

-To read the data from an Entry and convert it to a float, we use the following code:

```
        amt = float(self.txtAmount.get())
```

-We can also define styles for buttons, labels etc., and then apply these to the appropriate control:

```
        style = Style()
        style.configure("Exit.TButton", foreground="red", background="white")
        #T.Checkbutton for checboxes
        style.configure("MainButton.TButton", foreground="yellow", background="red")

        exitButton = Button(self, text="Exit", command=self.exitButtonClick)
        xitButton.configure(style="Exit.TButton")
        exitButton.place(x=xpos, y=ypos)
```

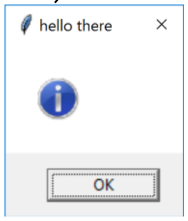
-The above code creates an exit button will style of "Exit.TButton" which has a text color of red and background color of "white"
-If someone clicks on the exit button, the exitButtonClick function will be called

-For a given UI, the different controls needed are placed a class derived from the Tkinter Frame class
-The typical code for a frame (equivalent of a form in a windows application) appears as:

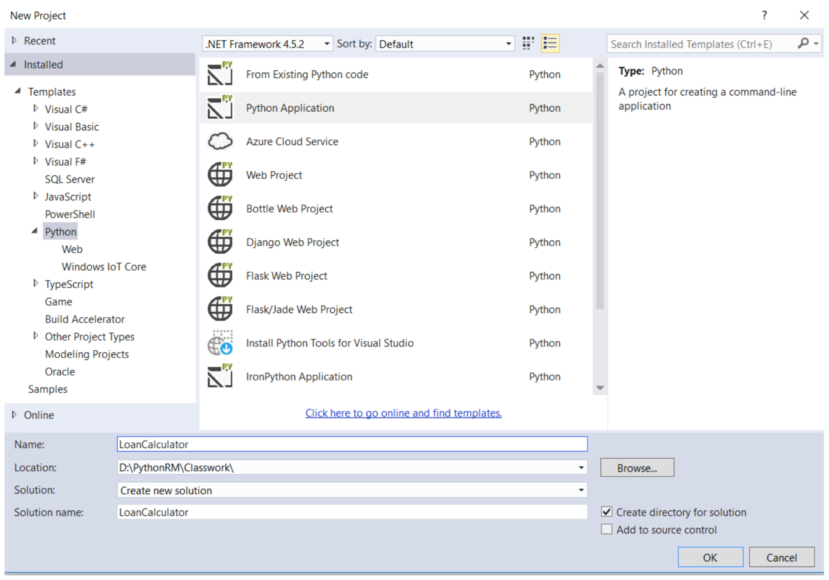
```
class MyFrame(Frame):

    def __init__(self, parent):
        Frame.__init__(self, parent)
        self.parent = parent
        self.initUI()
    def initUI(self):
        self.parent.title("Name of your Frame")
        self.style = Style()
        self.style.theme_use("default")

        # code for creating controls and their event handlers
```

-In a GUI application, we often need a message box (or a prompt)
-Tkinter provides a class called messagebox for this purpose → it has a showinfo method that can be called to show the messagebox e.g.,

```
    messagebox.showinfo("hello there")
```



-We will demonstrate some of the above GUI concepts through a simple loan calculation application
-Create a new Python application project called LoanCalculator as shown below.

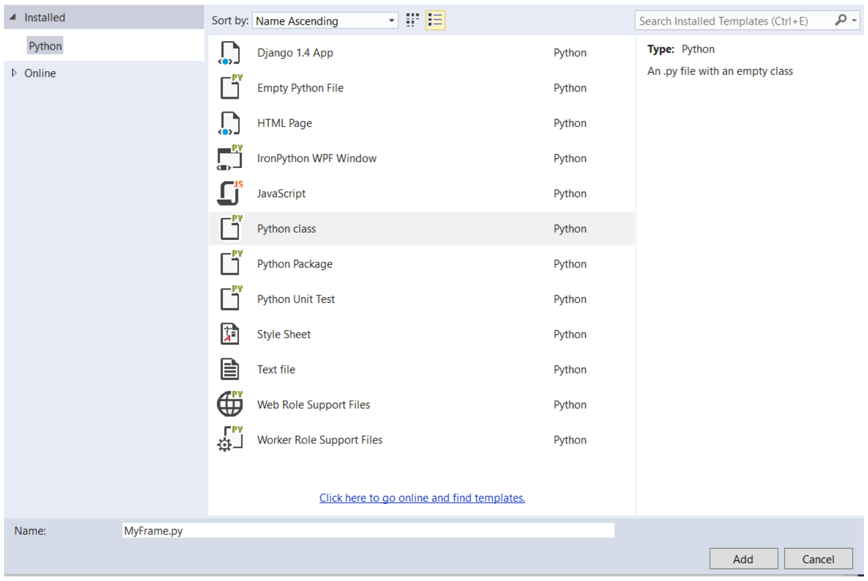


From Existing Python code	Python
Python Application	Python
Azure Cloud Service	Python
Web Project	Python
Bottle Web Project	Python
Django Web Project	Python
Flask Web Project	Python
Flask/Jade Web Project	Python
Install Python Tools for Visual Studio	Python
IronPython Application	Python

Type: Python
A project for creating a command-line application

Name: LoanCalculator
Location: D:\PythonRM\Classwork\
Solution: Create new solution
Solution name: LoanCalculator
[✓] Create directory for solution
[ ] Add to source control

-By right clicking on the name of the project in the solution explorer, add a class called MyFrame.py as shown below:


Installed
Python
Online
Sort by: Name Ascending
Search Installed Templates (Ctrl+E)
Django 1.4 App — Python
Empty Python File — Python
HTML Page — Python
IronPython WPF Window — Python
JavaScript — Python
Python class — Python
Python Package — Python
Python Unit Test — Python
Style Sheet — Python
Text file — Python
Web Role Support Files — Python
Worker Role Support Files — Python
Click here to go online and find templates.
Add
Cancel

Type: Python
An .py file with an empty class

Name: MyFrame.py

-Type the following code for creating the GUI for the loan calculator in the MyFrame.py as shown below:

```
from tkinter.ttk import Frame, Button,Label,Entry, Style
from tkinter import BOTH,END, messagebox
import sys

class MyFrame(Frame):

    def __init__(self, parent):
        Frame.__init__(self, parent)
        self.parent = parent
        self.initUI()


    def initUI(self):
        self.parent.title("Loan Calculator")
        self.style = Style()
        self.style.theme_use("default")  # default

        self.pack(fill=BOTH, expand=1)

        xpos = 40
        ypos = 30
        xpos2 = xpos+100
        l1 = Label(self, text="Amount", foreground = "#ff0000", background = "light blue",
font = "Arial 9")  # Arial 12 bold italic
        l1.place(x=xpos, y=ypos)
        self.txtAmount = Entry()
        self.txtAmount.place(x=xpos2, y = ypos, width=70)

        ypos += 30
        l2 = Label(self, text="Rate(%)", foreground = "#ff0000", background = "light blue",
font = "Arial 9")  # Arial 12 bold italic
        l2.place(x=xpos, y=ypos)
        self.txtRate = Entry()
        self.txtRate.place(x=xpos2, y = ypos)
        ypos += 30
        l3 = Label(self, text="Duration(months)", foreground = "#ff0000", background = "light
blue", font = "Arial 9")  # Arial 12 bold italic
        l3.place(x=xpos, y=ypos)
        self.txtDuration = Entry()
        self.txtDuration.place(x=xpos2, y = ypos)

        ypos += 30
        l4 = Label(self, text="Monthly Payment", foreground = "#ff0000", background =
"yellow", font = "Arial 9")  # Arial 12 bold italic
        l4.place(x=xpos, y=ypos)
        self.txtMonthlyPayment = Entry()
        self.txtMonthlyPayment.configure(state="readonly")
        self.txtMonthlyPayment.place(x=xpos2, y = ypos)

        ypos += 30
        l5 = Label(self, text="Total Payments", foreground = "#ff0000", background = "yellow",
font = "Arial 9")  # Arial 12 bold italic
        l5.place(x=xpos, y=ypos)
        self.txtTotalPayment = Entry();
        self.txtTotalPayment.configure(state="readonly")
        self.txtTotalPayment.place(x=xpos2, y = ypos)

        ypos += 30
        style = Style()
        style.configure("Exit.TButton", foreground="red", background="white")  #T.Checkbutton
for checboxes
        style.configure("MainButton.TButton", foreground="yellow", background="red")
        exitButton = Button(self, text="Exit", command=self.exitButtonClick)
        exitButton.configure(style="Exit.TButton")
        exitButton.place(x=xpos, y=ypos)

        calcButton = Button(self, text="Calculate", command=self.calcButtonClick)
        calcButton.configure(style="MainButton.TButton")
        calcButton.place(x=xpos2, y=ypos)

    def exitButtonClick(self):
        if (messagebox.askokcancel("OK to close?","Close application?")):
            self.parent.destroy
            exit()  # needed to close the main frame

    def calcButtonClick(self):
        amt = float(self.txtAmount.get())
        rate = float(self.txtRate.get())
        dur = float(self.txtDuration.get())
        monthlyPayment = amt * (rate / 1200.0) * ((rate / 1200 + 1)** dur) / (((rate / 1200 +
1)** dur) - 1)
        totalPayment = amt * ((1 + rate / 1200) ** dur)

        self.txtMonthlyPayment.configure(state="normal")  # has to be turned back to normal
otherwise, data is not modified, as text box is declared readonly
        self.txtMonthlyPayment.delete(0,END)
        self.txtMonthlyPayment.insert(0,format(monthlyPayment, "0.2f"))
        self.txtMonthlyPayment.configure(state="readonly")

        self.txtTotalPayment.configure(state="normal")
        self.txtTotalPayment.delete(0,END)
        self.txtTotalPayment.insert(0,format(totalPayment, "0.2f"))
        self.txtTotalPayment.configure(state="readonly")
```

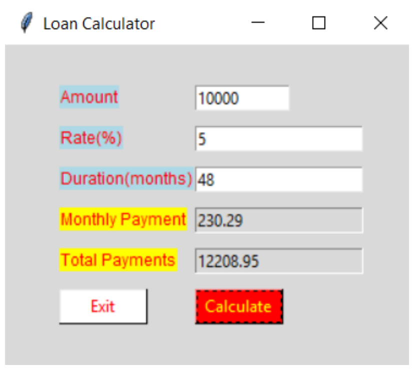
-The above code uses two variables xpos, and ypos to control the exact X and Y location coordinates for the different controls
-We increment the value of xpos or ypos so that all other controls' placement is relative to xpos and ypos, thus moving an entire set of controls to a different location becomes easier
-Tkinter does provide grid based layouts also, but for an advanced GUI, the precise placement through X and Y coordinate specifications is a better approach

-Note that the Entry objects representing the monthly payments (txtMonthlyPayment), and total payments (txtTotalPayement) are declared with a state of "readonly" so that the user cannot modify these
-In the button handler for the calculations (calcButtonClick), the state of these text boxes has to be switched back to the normal mode, otherwise we can display the calculation results in these text boxes

-Type the following code in the LoanCalculator.py file to declare the main method and to invoke the MyFrame class

```
import sys
from tkinter import Tk
from MyFrame import MyFrame

def main():
    root = Tk()
    root.geometry("300x300")
    app = MyFrame(root)   #creates a frame of 300x300 pixels
    root.mainloop()

if __name__ == "__main__":
    sys.exit(int(main() or 0))
```

-Build and test the application:



Amount	10000
Rate(%)	5
Duration(months)	48
Monthly Payment	230.29
Total Payments	12208.95

Creating GUI Applications with Multiple Frames

-When multiple frames (or sometimes called forms) are needed in an application, some of these are used to collect information from the user such as login information, or customer feedback etc.

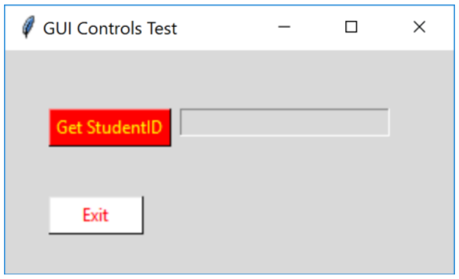
-In some cases, we want the frame to have full focus and not allow the user to do anything else unless they have completed the information in the form and submitted it, or cancelled the form → this is referred to as "modal" behavior
-Similarly, we may need to display multiple frames at the same time and allow the user to switch between one or the other
-For example, one form may be displaying user's contact info, where as another may display recent transaction history → such forms that coexist with each other are referred to as "modeless"

-The general procedure for creating another frame is to create a class and inherit it from TopLevel which further inherits from the Frame class
-Then in the main form, through a menu or a button, invoke the frame as a modal or modeless display
-We will demonstrate this through another Python application. Creation of other user interface elements such as menus, checkboxes, radio buttons and listboxes etc.. will also be covered

Create a new Python application called GUIControls. Type the following code in the GUIControls.py file:

```
import sys
from tkinter import *
from MyFrame import MyFrame

def main():
    root = Tk()
    root.geometry("700x600")
    app = MyFrame(root)
    root.mainloop()

if __name__ == "__main__":
    sys.exit(int(main() or 0))
```

Our goal is simply to place two buttons and a label in our frame as shown below. One button is for collecting student ID from a modal frame and displaying it in the label, and the other is the exit button to close the application:

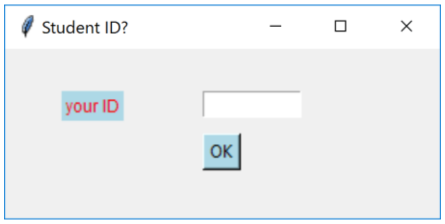


Add a MyFrame class to the project with the following code:

```
from tkinter.ttk import Frame, Button,Label,Entry, Style
from tkinter import BOTH,END, messagebox
import sys
from StudentIDDlg import StudentIDDlg
```

```
class MyFrame(Frame):

    def __init__(self, parent):
        Frame.__init__(self, parent)
        self.parent = parent
        self.initUI()


    def initUI(self):
        self.parent.title("GUI Controls Test")
        self.style = Style()
        self.style.theme_use("default")
        self.pack(fill=BOTH, expand=1)
        #--------------------------------------------

        xpos = 30
        ypos = 40
        xpos2 = xpos + 90

        #------------styling----------------------------------
        style = Style()
        style.configure("Exit.TButton", foreground="red", background="white")
        style.configure("MainButton.TButton", foreground="yellow", background="red")
        #----------------------------------------------------

        testButton = Button(self, text="Get StudentID", command=self.btnGetStudentIDClick)
        testButton.configure(style="MainButton.TButton")
        testButton.place(x=xpos, y=ypos)

        self.txtID = Entry(self, text="", foreground = "#ff0000", background = "light blue",
font = "Arial 9")  # Arial 12 bold italic
        self.txtID.place(x=xpos2, y=ypos)
        self.txtID.configure(state="readonly")

        ypos += 30

        ypos += 30
        exitButton = Button(self, text="Exit", command=self.exitButtonClick)
        exitButton.configure(style="Exit.TButton")
        exitButton.place(x=xpos, y=ypos)

    def exitButtonClick(self):
        if (messagebox.askokcancel("OK to close?","Close application?")):
            self.parent.destroy
            exit()  # needed to close the main frame

    def btnGetStudentIDClick(self):
        #show modal dialog and collect student ID
        dlg = StudentIDDlg("your ID", "Student ID?", "Please Enter your Student ID:")
        dlg.grab_set()  #events only go to the modal dialog
        self.wait_window(dlg)

        self.txtID.configure(state="normal")
        self.txtID.delete(0,END)
        self.txtID.insert(0,dlg.getID())
        self.txtID.configure(state="readonly")
        print(dlg.getID())
```

-Don't run the program yet - when someone clicks on the "Get Student ID" button, we want to show a modal frame as shown below:

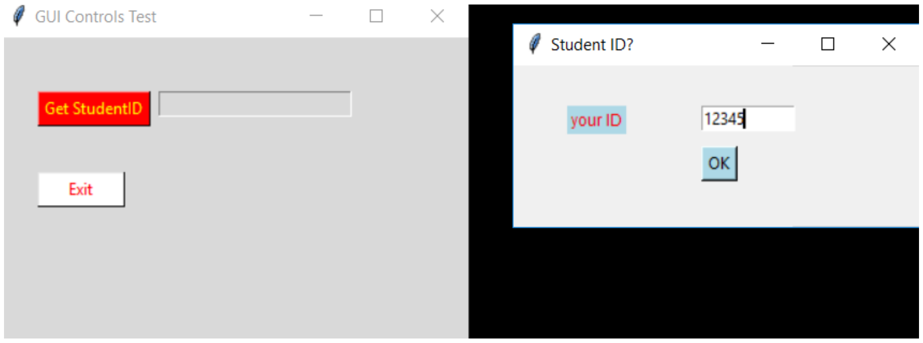


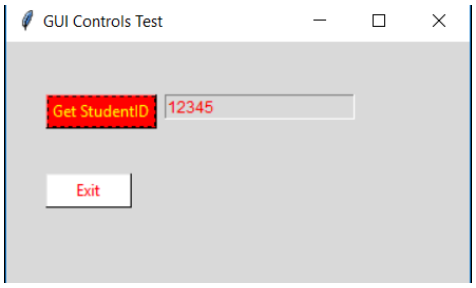
For this purpose, add another Python class to the project called StudentIDDlg with the following code:

```
from tkinter.ttk import Frame, Button,Label,Entry, Style
from tkinter import *

class StudentIDDlg(Toplevel):
    def __init__(self, initialText, title, labeltext = '' ):
        Toplevel.__init__(self)
        self.initUI(initialText,title,labeltext)

    def initUI(self,initialText,title,labeltext=''):
        self.STID = initialText
        self.geometry("200x120")
        if len(title) > 0: self.title(title)
        self.style = Style()
        self.style.theme_use("default")  # default

        style = Style()
        style.configure("Exit.TButton", foreground="red", background="white")
        style.configure("MainButton.TButton", foreground="yellow", background="red")

        if len(labeltext) == 0: labeltext = 'Please enter your ID..'
        self.bind("<Return>", self.ok)

        xpos = 40
        ypos = 30
        xpos2 = xpos+100
        l1 = Label(self, text=initialText, foreground = "#ff0000", background = "light 
blue", font = "Arial 9")  # Arial 12 bold italic
        l1.place(x=xpos, y=ypos)

        self.txtID = Entry(self)
        self.txtID.place(x=xpos2, y = ypos, width=70)
        self.txtID.bind("<Return>", self.ok)
        self.txtID.bind("<Escape>", self.cancel)
        self.txtID.focus_set()

        ypos += 30

        okButton = Button(self, text="OK", background = "light blue", command=self.ok)
        #okButton.configure(style="ExitButton.TButton") # does not work
        okButton.place(x=xpos2, y=ypos)

    def getID(self):
        return self.STID

    def ok(self, event=None):
        self.STID = self.txtID.get()
        self.destroy()

    def cancel(self, event=None):
        self.destroy()
```

-Note that the above class is derived from TopLevel as we want it to be created in a modal manner

-Build and test the application. Once you click on the "Get Student ID" button, a dialog to enter the student ID will appear
-Once you enter an ID and close the Student ID Dlg, the ID entered will appear in the label in the main form as shown below.





Displaying Modeless Frames

-We wish to display the loan calculator that we developed earlier as a modeless frame → this way, both the main frame and the calculator are available to the user
-In the same GUIControls project, add another button to the MyFrame class
-Add the following code (shown in bold) to the `def initUI(self):` method in the MyFrame class just below the code for declaring the test button:

```
self.txtID = Entry(self, text="", foreground = "#ff0000", background = "light blue", font
= "Arial 9")  # Arial 12 bold italic
        self.txtID.place(x=xpos2, y=ypos)
        self.txtID.configure(state="readonly")

        ypos += 30

        btnLoanCalc = Button(self, text="Loan Calculator",
command=self.loanCalcButtonClick)
```

```
        btnLoanCalc.configure(style="Exit.TButton")
        btnLoanCalc.place(x=xpos, y=ypos)
```

-Towards the end of the MyFrame.py class, add the event hander for displaying the loan calculator in a modeless manner as shown below:

```
    def loanCalcButtonClick(self):
        if (self.LCD is None) :
            self.LCD = LoanCalculator()
        else:
            if (self.LCD.winfo_exists()):
                self.LCD.focus()
            else:
                self.LCD = LoanCalculator()
```

-The above code checks to see if the loan calculator is already displayed. If so, it brings the focus to it; if not, it creates a new loan calculator object if it has been closed
-Declare the LCD object in te def__init__ method in the MyFrame class as:

```
def __init__(self, parent):
        Frame.__init__(self, parent)
        self.parent = parent
        self.LCD = None   #Loan calculator object
        self.initUI()       …..
```

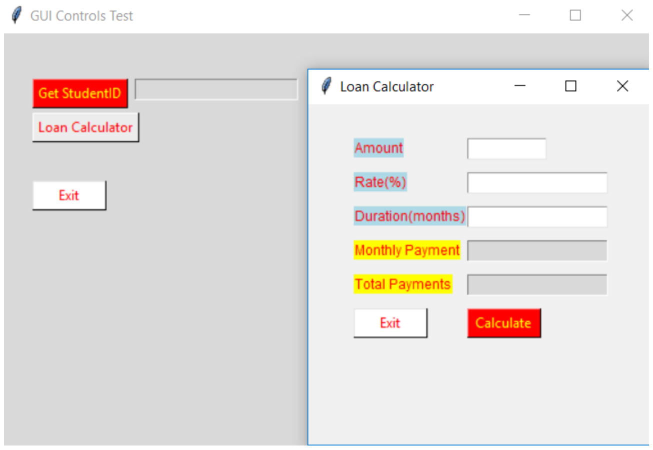
-Add a class LoanCalculator to the project with the following code:

```
from tkinter import *
from tkinter.ttk import Frame, Button,Label,Entry, Style
from tkinter import BOTH,END, messagebox
import sys
class LoanCalculator(Toplevel):

    def __init__(self):
        Toplevel.__init__(self)
        self.initUI()


    def initUI(self):
        self.title("Loan Calculator")
        self.geometry("300x300")
        self.style = Style()
        self.style.theme_use("default")  # default

        #self.pack(fill=BOTH, expand=1)

        xpos = 40
        ypos = 30
        xpos2 = xpos+100
        l1 = Label(self, text="Amount", foreground = "#ff0000", background = "light
blue", font = "Arial 9")  # Arial 12 bold italic
        l1.place(x=xpos, y=ypos)
        self.txtAmount = Entry(self)
        self.txtAmount.place(x=xpos2, y = ypos, width=70)

        ypos += 30
        l2 = Label(self, text="Rate(%)", foreground = "#ff0000", background = "light 
blue", font = "Arial 9")  # Arial 12 bold italic
        l2.place(x=xpos, y=ypos)
        self.txtRate = Entry(self)
        self.txtRate.place(x=xpos2, y = ypos)

        ypos += 30
        l3 = Label(self, text="Duration(months)", foreground = "#ff0000", background = 
"light blue", font = "Arial 9")  # Arial 12 bold italic
        l3.place(x=xpos, y=ypos)
        self.txtDuration = Entry(self)
        self.txtDuration.place(x=xpos2, y = ypos)

        ypos += 30
        l4 = Label(self, text="Monthly Payment", foreground = "#ff0000", background = 
"yellow", font = "Arial 9")  # Arial 12 bold italic
        l4.place(x=xpos, y=ypos)
        self.txtMonthlyPayment = Entry(self)
        self.txtMonthlyPayment.configure(state="readonly")
        self.txtMonthlyPayment.place(x=xpos2, y = ypos)

        ypos += 30
        l5 = Label(self, text="Total Payments", foreground = "#ff0000", background = 
"yellow", font = "Arial 9")  # Arial 12 bold italic
        l5.place(x=xpos, y=ypos)
        self.txtTotalPayment = Entry(self);
        self.txtTotalPayment.configure(state="readonly")
        self.txtTotalPayment.place(x=xpos2, y = ypos)

        ypos += 30
        style = Style()
        style.configure("Exit.TButton", foreground="red", background="white") 
#T.Checkbutton for checboxes
        style.configure("MainButton.TButton", foreground="yellow", background="red")
        exitButton = Button(self, text="Exit", command=self.exitButtonClick)
        exitButton.configure(style="Exit.TButton")
        exitButton.place(x=xpos, y=ypos)

        calcButton = Button(self, text="Calculate", command=self.calcButtonClick)
        calcButton.configure(style="MainButton.TButton")
        calcButton.place(x=xpos2, y=ypos)

    def exitButtonClick(self):
        if (messagebox.askokcancel("OK to close?","Close aapplication?")):
            self.parent.destroy
            exit()  # needed to close the main frame

    def calcButtonClick(self):
        amt = float(self.txtAmount.get())
        rate = float(self.txtRate.get())
        dur = float(self.txtDuration.get())
        monthlyPayment = amt * (rate / 1200.0) * ((rate / 1200 + 1)** dur) / (((rate / 
1200 + 1)** dur) - 1)
        totalPayment = amt * ((1 + rate / 1200) ** dur);

        self.txtMonthlyPayment.configure(state="normal")  # has to be turned back to 
normal otherwise, data is not modified
        self.txtMonthlyPayment.delete(0,END)
        self.txtMonthlyPayment.insert(0,format(monthlyPayment, "0.2f"))
        self.txtMonthlyPayment.configure(state="readonly")

        self.txtTotalPayment.configure(state="normal")
        self.txtTotalPayment.delete(0,END)
        self.txtTotalPayment.insert(0,format(totalPayment, "0.2f"))
        self.txtTotalPayment.configure(state="readonly")
```

-Add an import for the LoanCalculator in the MyFrame class as:

```
from LoanCalculator import LoanCalculator
```

-Build and test the application:



Creating Menus

-The Menu class allows us to create menus in an easy manner. Add the following code to the MyFrame class in the def initUI method (the added part is shown in bold):

```
def initUI(self):
        self.parent.title("GUI Controls Test")
        self.style = Style()
        self.style.theme_use("default")
        self.pack(fill=BOTH, expand=1)
        #--------------------------------------------

        #--------------------create menus------------
        menuBar = Menu(self.parent)
        mnuFile = Menu(menuBar, tearoff=0)
        menuBar.add_cascade(label="File", menu=mnuFile)
        mnuFile.add_command(label="Open", command=self.mnuOpenFileClick)
        mnuFile.add_command(label="Save", command=self.mnuSaveFileClick)
        mnuFile.add_separator()
        mnuFile.add_command(label="Exit", command=self.exitButtonClick)


        mnuCustomers = Menu(menuBar, tearoff=0)
        menuBar.add_cascade(label="Loan Processing", menu=mnuCustomers)
        mnuCustomers.add_command(label="Loan Calculator",
command=self.loanCalcButtonClick)
```

```
        mnuCustomers.add_separator()
        mnuCustomers.add_command(label="Provide Feedback",
command=self.mnuShowFeedbackClick)

        self.parent.config(menu=menuBar)
        #--------------------------------------------…….
```

-Add the following line (shown in bold) to the MyFrame class

```
def __init__(self, parent):
        Frame.__init__(self, parent)
        self.parent = parent
        self.LCD = None   #Loan calculator object
        self.FBD = None   #Feedback Dlg object
        self.initUI()
```

-Add the event handlers for the menus by adding the following functions to the end of MyFrame class

```
def loanCalcButtonClick(self):
        if (self.LCD is None) :
            self.LCD = LoanCalculator()
        else:
            if (self.LCD.winfo_exists()):
                self.LCD.focus()
            else:
                self.LCD = LoanCalculator()

    def mnuOpenFileClick(self):
        options = opts = {}
        opts['initialdir'] = 'd:\\PythonRM'
        opts['filetypes'] = [('all files', '.*'), ('jpeg files', '.jpg')]
        fname = filedialog.askopenfilename(**options) # file in read mode
        img = misc.imread(fname)  # read the image file
        toimage(img).show()

    def mnuSaveFileClick(self):
        print("OK2")

    def mnuShowFeedbackClick(self):
        if (self.FBD is None):
            self.FBD = FeedBackDlg()
        else:
            if (self.FBD.winfo_exists()):
                self.LCD.focus()
            else:
                self.FBD = FeedBackDlg ()
```

-For the open file, the above code displays the open file dialog where the user can select an image file (e.g., with a jeg or jpg extension), then display it using the SciPy toimage method. The initial directory to search for a file can be specified as shown above.

-Add the following imports to the beginning of MyFrame class:

```
from scipy import misc
from scipy.misc import toimage    #for displaying an image in photoviewer
```

-Add a class called FeedBackDlg to the project with the following code:

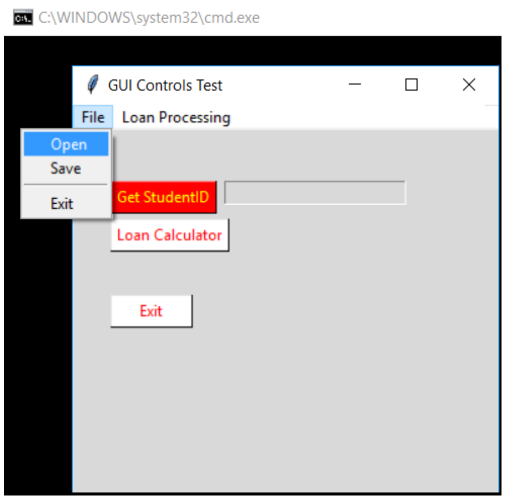
```
from tkinter import *
from tkinter.ttk import Frame, Button,Label,Entry, Style
from tkinter import BOTH,END, messagebox
import sys
class FeedBackDlg(Toplevel):

    def __init__(self):
        Toplevel.__init__(self)
        self.initUI()


    def initUI(self):
        self.title("Feedback..")
        self.geometry("600x400")
        self.style = Style()
        self.style.theme_use("default")  # default

        xpos = 40
        ypos = 30
        xpos2 = xpos+100
        l1 = Label(self, text="First Name", foreground = "#ff0000", background = "light 
blue", font = "Arial 9")  # Arial 12 bold italic
        l1.place(x=xpos, y=ypos)
        self.txtFirstName = Entry(self)
        self.txtFirstName.place(x=xpos2, y = ypos, width=70)

        ypos += 30
        l2 = Label(self, text="Email", foreground = "#ff0000", background = "light blue", 
font = "Arial 9")  # Arial 12 bold italic
        l2.place(x=xpos, y=ypos)
        self.txtEmail = Entry(self)
        self.txtEmail.place(x=xpos2, y = ypos)

        ypos += 30
        l3 = Label(self, text="Your interest in type of our products:", foreground = 
"#ff0000", background = "light blue", font = "Arial 9")  # Arial 12 bold italic
        l3.place(x=xpos, y=ypos)

        ypos += 30
        self.electronicsChoice = BooleanVar()
        self.electronicsChoice.set(True)
        self.chkElectronics = Checkbutton(self, text="Electronics", 
variable=self.electronicsChoice)
                                # ,command=self.chkElectronicsChanged)
        self.chkElectronics.place(x=xpos2, y = ypos)

        self.sportsChoice = BooleanVar()
        self.chkSports = Checkbutton(self, text="Sports", variable=self.sportsChoice)
        self.chkSports.place(x=xpos2+80, y = ypos)

        self.gardeningChoice = StringVar()
        self.gardeningChoice.set("YES")
        self.chkGardening = Checkbutton(self, text="Gardening", 
variable=self.gardeningChoice,
                                        onvalue="YES", offvalue="NO", )
        self.chkGardening.place(x=xpos2+160, y = ypos)

        ypos += 30
        #----------radio buttons---------------
        serviceChoices = [("Disappointed", "0"),("Satisfied", "1"),("Good",
"2"),("Excellent", "3")]
        self.serviceFeedback = StringVar()
        self.serviceFeedback.set("2") # initial value
        inc = 0
        for text, val in serviceChoices:
            radBtn = Radiobutton(self, text=text, variable=self.serviceFeedback,
value=val)
            radBtn.place(x = xpos2 + inc, y = ypos)
            inc += 100
       #--------------------------------------

        ypos += 30

        #--------listbox-----------
        states = [("Connecticut", "0"),("New York", "1"),("New Jersey",
"2"),("Massachussetts", "3")]
        self.lb = Listbox(self,selectmode=EXTENDED, height=len(states))
        self.lb.place(x=xpos2,y = ypos)
        self.lb.delete(0, END) # clear
        for key,val in states:
            self.lb.insert(END, key)
            self.data = val
        #---------------------------
        ypos += 80
        #--------drop down listbox-----------
        #departments = [("Sales","100"),("Marketing","200"), ("HR","300"),
("Technology","300")]
        departments = ["Sales","Marketing", "HR","Technology"]

        self.dept = StringVar()
        self.dept.set("HR") # initial value
        #self.ddlDept = OptionMenu(self, self.dept,*(dict(departments).keys()))
        self.ddlDept = OptionMenu(self, self.dept,*departments)
        self.ddlDept.place(x=xpos2,y = ypos)
        #---------------------------


        ypos += 30  * (len(states)-1)
        style = Style()
        style.configure("Exit.TButton", foreground="red", background="white")
#T.Checkbutton for checboxes
        style.configure("MainButton.TButton", foreground="yellow", background="red")

        btnSubmit = Button(self, text="Submit", command=self.btnSubmitClick)
        btnSubmit.configure(style="MainButton.TButton")
        btnSubmit.place(x=xpos2, y=ypos)

    def exitButtonClick(self):
        if (messagebox.askokcancel("OK to close?","Close aapplication?")):
            self.parent.destroy
            exit()  # needed to close the main frame
    def btnSubmitClick(self):
        print("electronics = " + str(self.electronicsChoice.get()) +
              " sports=" + str(self.sportsChoice.get()) + " gardening=" +
str(self.gardeningChoice.get()))
        print("service feedback = " + str(self.serviceFeedback.get()))
        print("state = " + str(self.lb.curselection()[0]))
        print("department = " + self.dept.get())
```

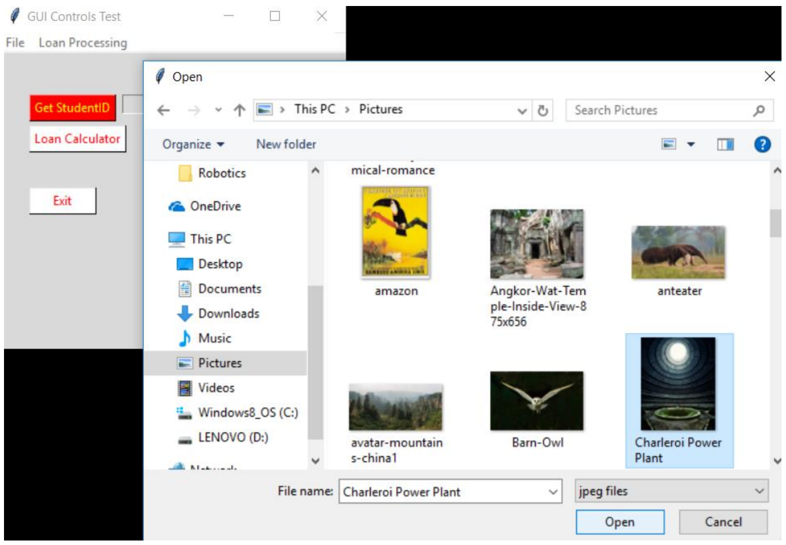
-The above FeedBackDlg shows the creation of radio buttons, checkboxes, list boxes, drop downs. Study the code in the above class to understand how these are created and how the defaults are set.

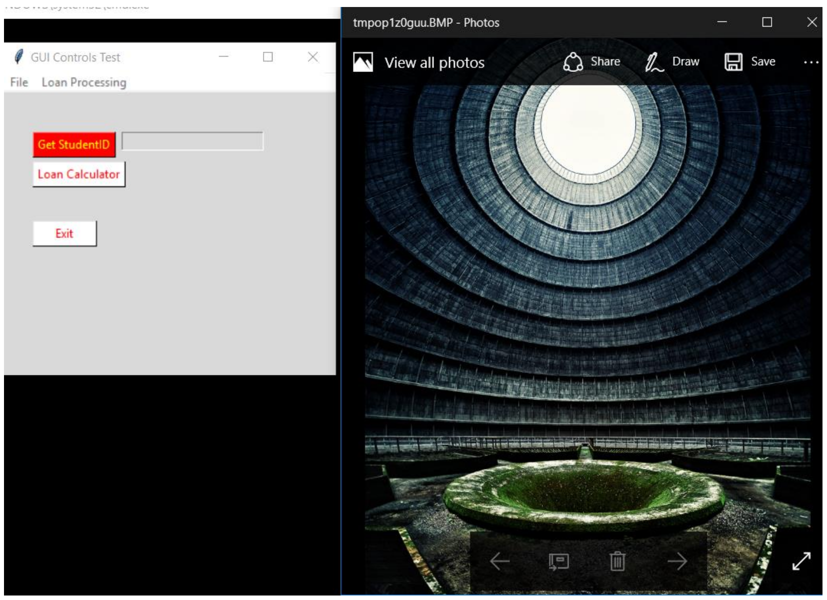
-Add the import for the FeedBackDlg in the MyFrame class. You will also need to import the Menu class:

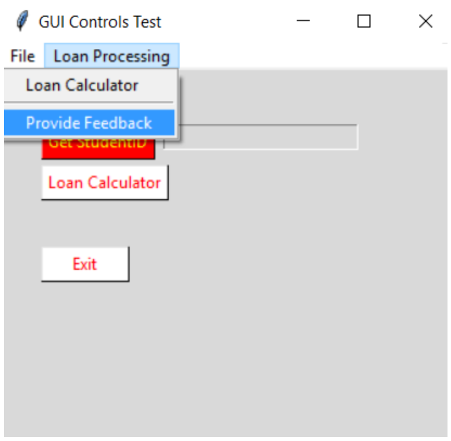
```
from FeedBackDlg import FeedBackDlg
from tkinter import Menu, BOTH,END,NONE, messagebox, filedialog
```

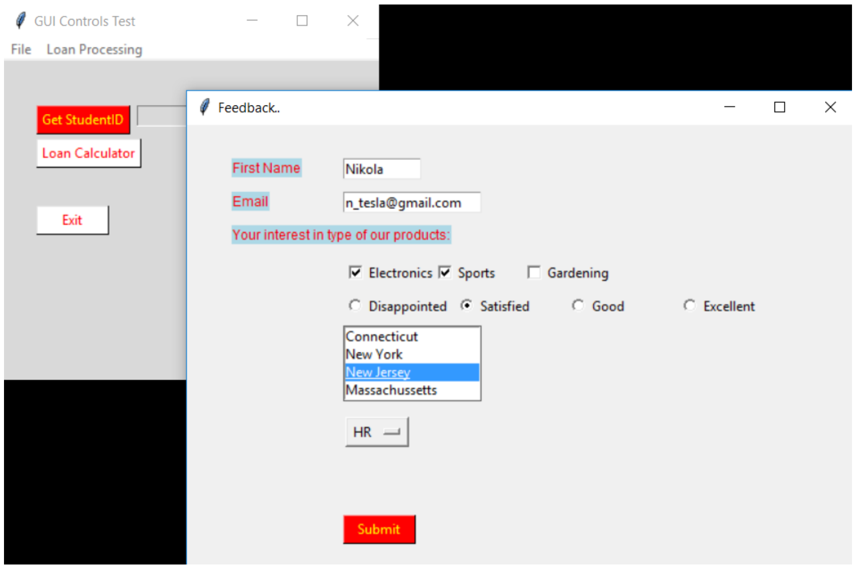
-Build and test the application:











First Name Nikola

Email n_tesla@gmail.com

Your interest in type of our products:

[✓ Electronics] [✓ Sports] Gardening

Disappointed [✓ Satisfied] Good Excellent

Connecticut
New York
[✓ New Jersey]
Massachussetts

HR