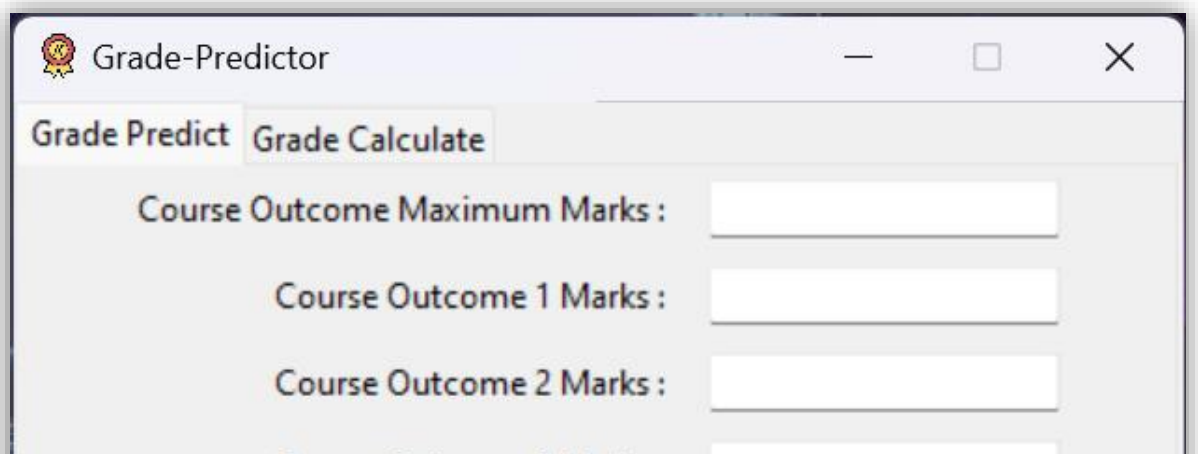
Console to Clicks

Learn how to convert your python programs to GUI programs using Tkinter

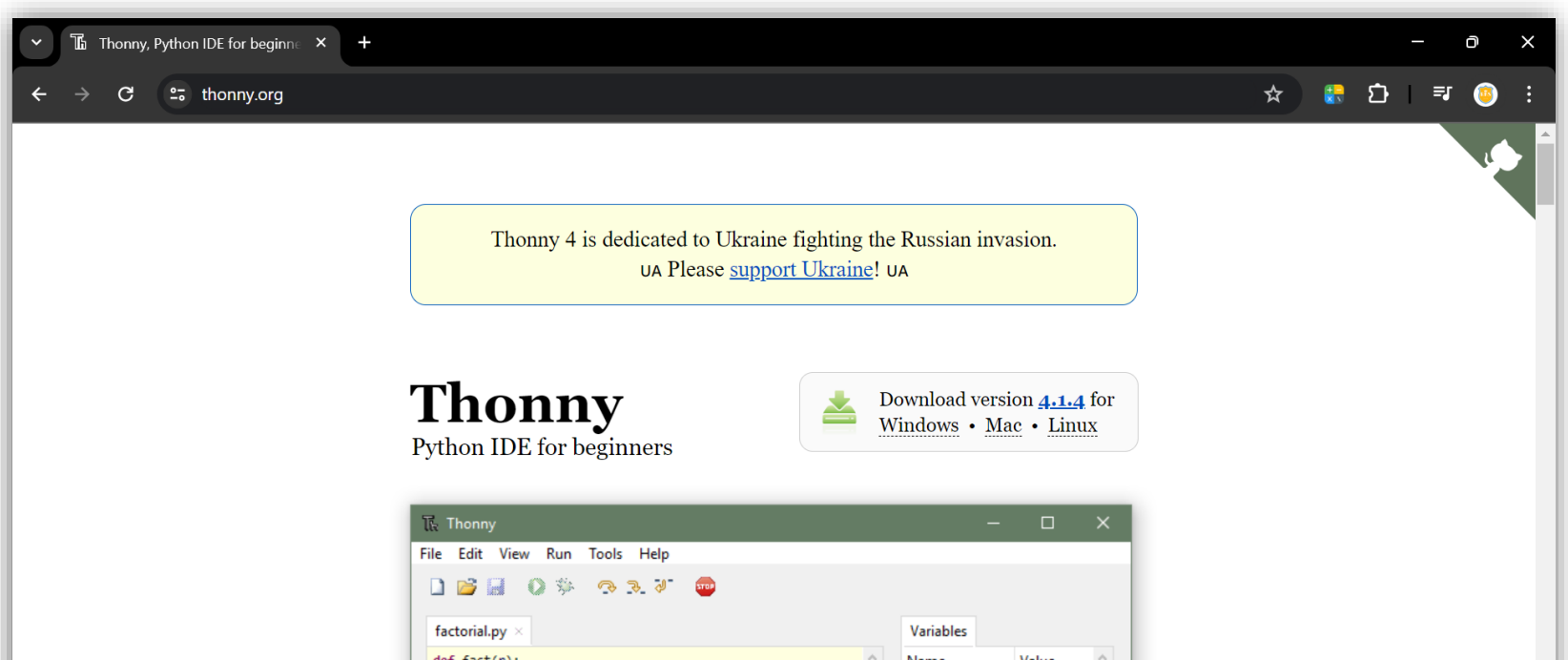


What is Tkinter

- Tkinter is a useful tool for creating a wide variety of graphical user interfaces, including windows, dialog boxes, and custom widgets. It is particularly well-suited for building desktop applications and adding a GUI to command-line programs.
- Full Form of Tkinter: The name “Tkinter” comes from “Tk interface“, referring to the Tk GUI toolkit that Tkinter is based on.

Installing Thonny IDE

- Thonny comes with Tkinter pre-configured by default.
- So it feels easy to begin with it.
- Download from thonny.org and install.



Summary of Existing Program

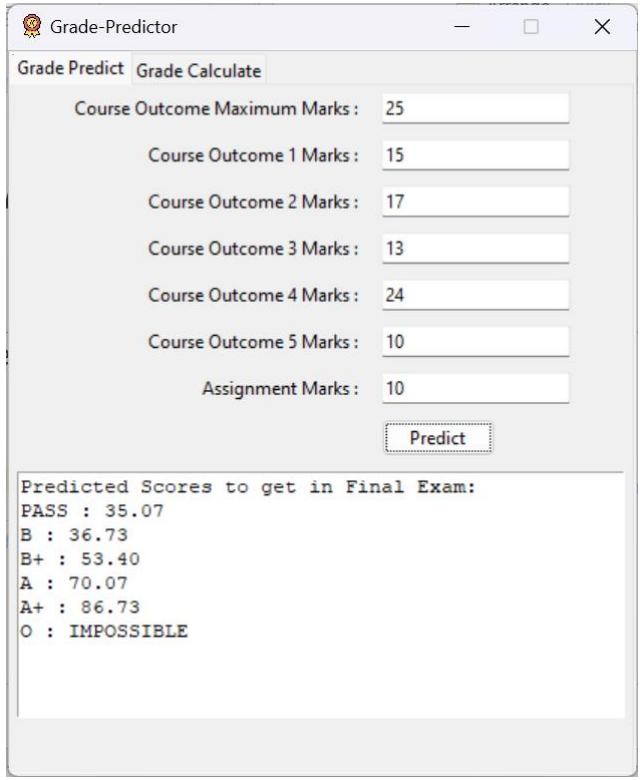
Explaining main.py

Overview of Conversion

Console Based

```
Shell
Grade-Predictor

Main Menu
1) Predict Grade
2) Calculate Grade
3) Exit

Enter your choice (1-3): 1
Enter the Course Outcome Maximum Marks: 25
Enter the Course Outcome 1 Marks      : 15
Enter the Course Outcome 2 Marks      : 17
Enter the Course Outcome 3 Marks      : 13
Enter the Course Outcome 4 Marks      : 24
Enter the Course Outcome 5 Marks      : 10
Enter the Assignment Marks            : 10

Predicted Scores to get in Final Exam:
PASS : 35.07
B : 36.73
B+ : 53.40
A : 70.07
A+ : 86.73
O : IMPOSSIBLE
```

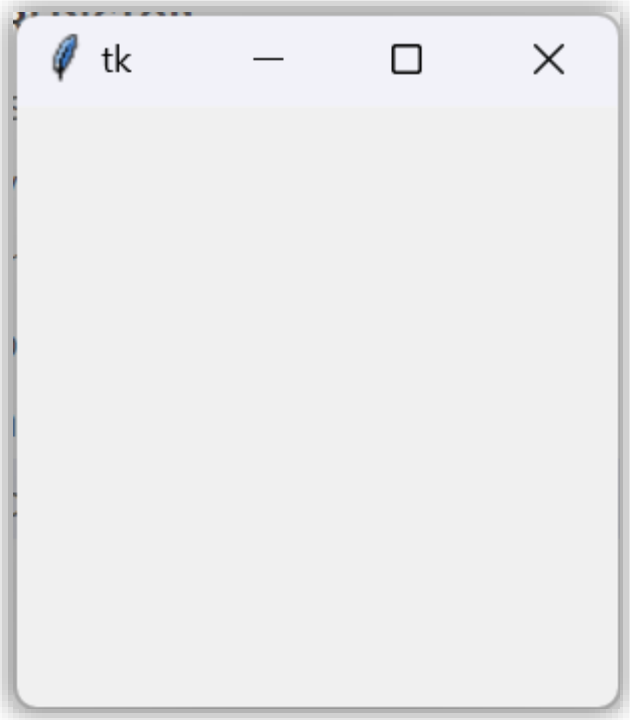
GUI Based



Creating a Blank Window

- The journey of GUI development with Tkinter starts with creating a blank window.

```
import tkinter as tk
root = tk.Tk()
root.mainloop()
```



Titles

Console

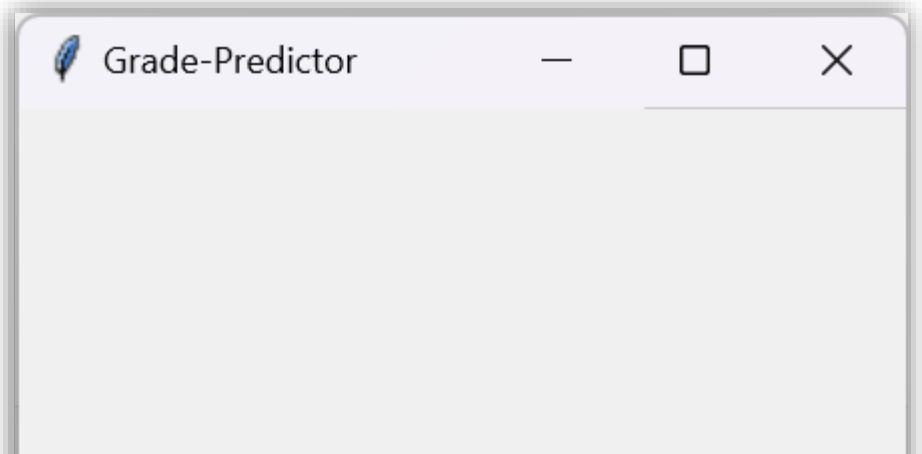
- In Console Title is just a print statement.

```
print("\n\nGrade-Predictor")
```

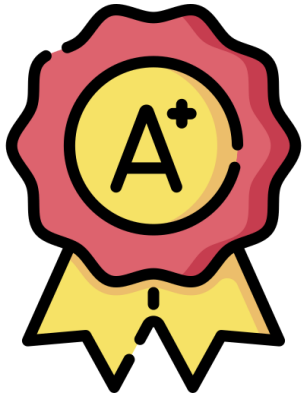
GUI

- In GUI we can set a title for the created blank window

```
root.title("Grade-Predictor")
```

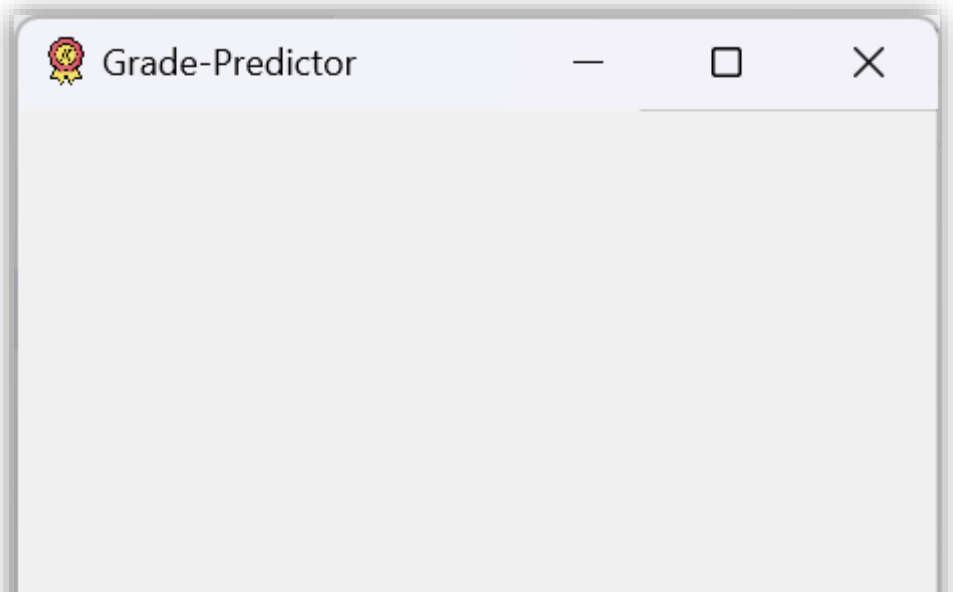


Adding Icon



- As you are developing GUI , you can add a PNG or any format of image as a icon in tkinter.

```
root.iconphoto(False, tk.PhotoImage(file="icon.png"))
```



Conversion of Main Menu : Tabs

Console

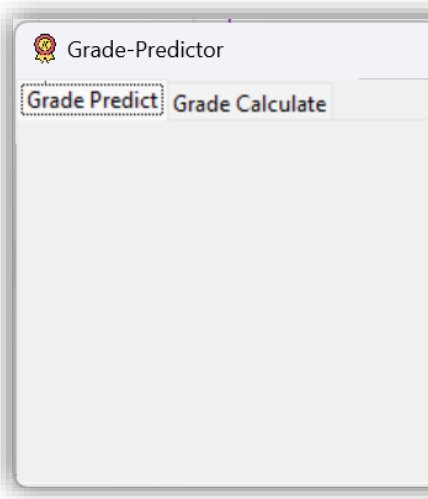
- In console the main menu is just printing a menu normally with numbered choice as input from user.

```
Main Menu
1) Predict Grade
2) Calculate Grade
3) Exit

Enter your choice (1-3):
```

GUI

- In GUI we have many ways to handle this.
- In this example I have used tabs as a GUI alternative for main menus.

```python
import tkinter.ttk as ttk
tabs = ttk.Notebook(root)

gp_frame = ttk.Frame(tabs)
gp_frame.pack(padx = 5 , pady = 5)
gc_frame = ttk.Frame(tabs)
gc_frame.pack(padx = 5 , pady = 5)

tabs.add(gp_frame, text = "Grade Predict")
tabs.add(gc_frame, text = "Grade Calculate")
tabs.pack(expand = 1, fill ="both")
```



Console : Getting an Input

- In console to get a input we use just a input() function.

```
co_max = int(input("Enter the Course Outcome Maximum Marks: "))
```

- We can also write the above code as,

```
print("Enter the Course Outcome Maximum Marks: ",end="")
co_max = int(input())
```

```
Enter your choice (1-3): 1
Enter the Course Outcome Maximum Marks:
```

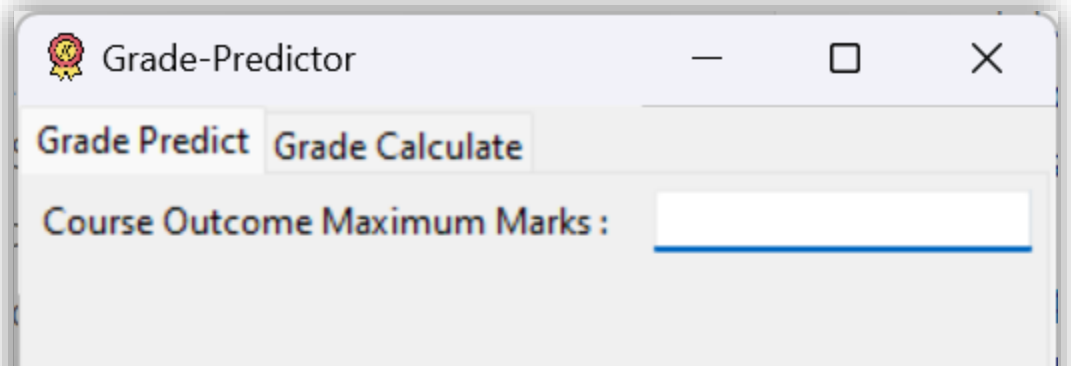
GUI : Creating a Form

- Generally in a form there will be a text along with a input area.
- From the last slide we can replace the print statement with Label.

```
pre_co_max_lbl = ttk.Label(gp_frame, text = "Course Outcome Maximum Marks : ")
pre_co_max_lbl.grid(row = 0, column = 0, padx = 5 , pady = 5, sticky = 'e’)
```

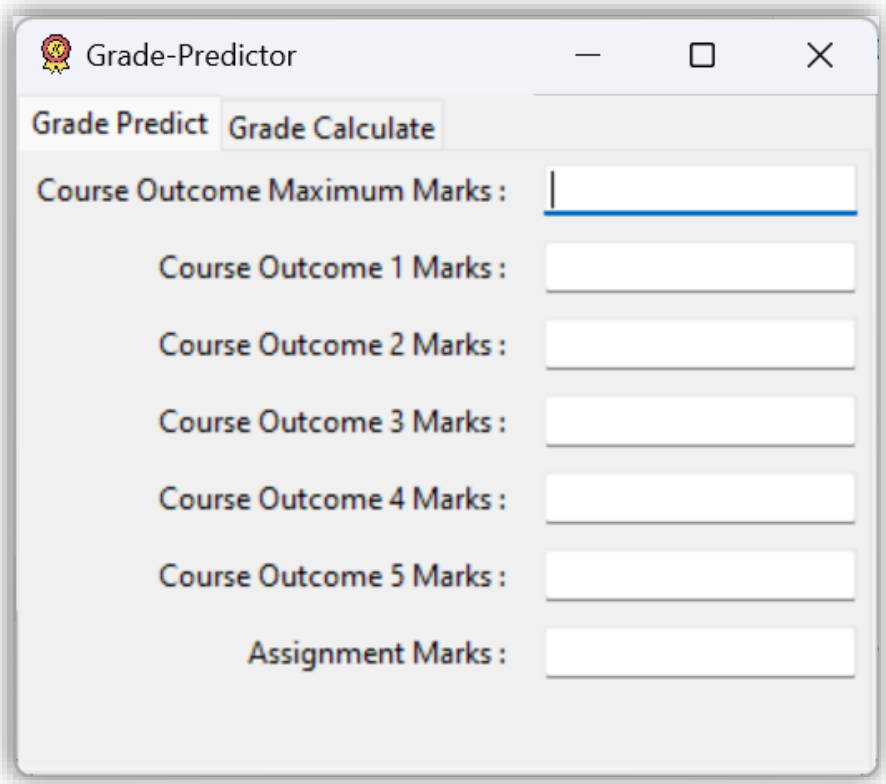
- From the last slide we can replace the input statement with Entry.

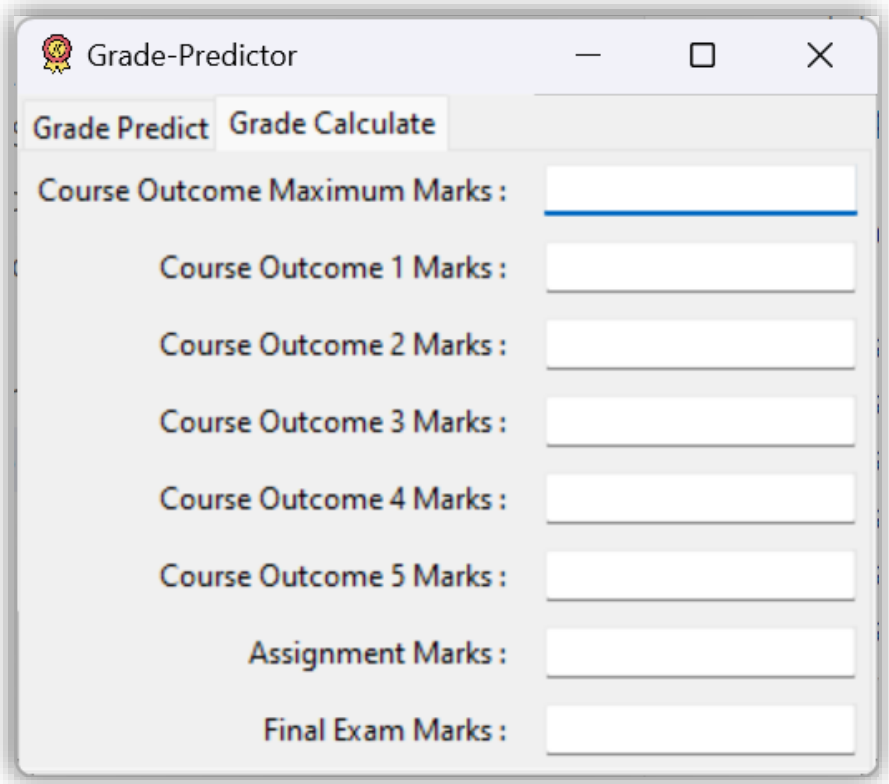
```
pre_co_max = tk.StringVar()
pre_co_max_ent = ttk.Entry(gp_frame, textvariable = pre_co_max)
pre_co_max_ent.grid(row = 0, column = 1, padx = 5 , pady = 5, sticky = 'w')
```



GUI : Creating a Form

- Do the last for all needed inputs for both tabs.
- Your final output should look like….



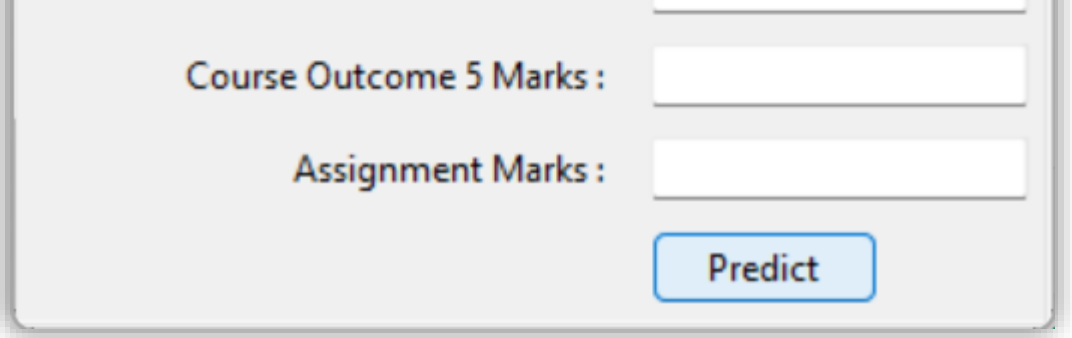


Console : After Getting Input

- After getting all inputs, by just pressing Enter the console based programs starts to do next lines of code sequentially. In this case calling appropriate functions.

```
co5 =    float(input("Enter the Course Outcome 5 Marks      :
assign = float(input("Enter the Assignment Marks           : "

res = grade_predict(co_max,co1,co2,co3,co4,co5,assign)

print("\nPredicted Scores to get in Final Exam:")
for i in range(6):
```

GUI : Buttons



- In case of GUI we need a input from user to invoke the functions.
- Usually we use Buttons to invoke such functions.

```
pre_btn = ttk.Button(gp_frame, text='Predict', command = predict)
pre_btn.grid(row = 7, column = 1, padx = 5 , pady = 5, sticky = 'w’)
```

- In the above code we invoke a function called predict passed in the command parameter.
- Note we cannot pass parameters in the button function like we do normally

Button Functions

- Button function do some process and change the GUI parameters for output.
- In this example we make use of the same function from the console program by importing it to GUI program.

```
from main import *
```

…

```
def predict():
    try:
        res = grade_predict(int(pre_co_max.get()),int(pre_co1.get()),int(pre
    except ValueError:
        #msgbox.showerror(title='Value Error', message="Invalid Input")
```

NOTE
We use get() method to get the value from tk.StringVar()
In this case, pre_co_max.get()c

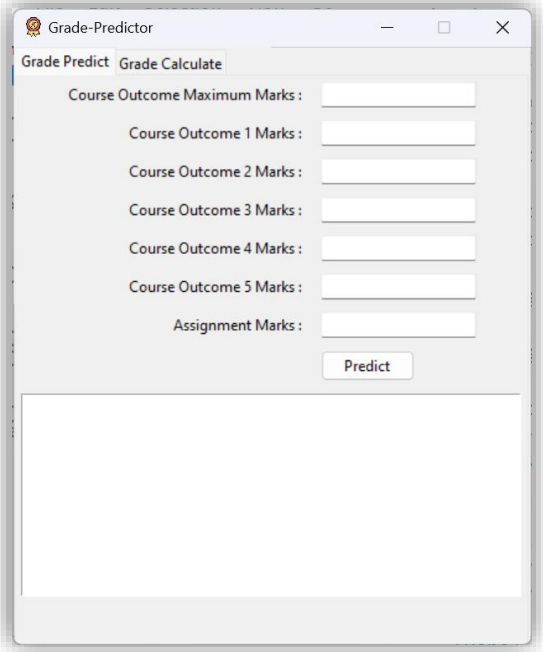
Console : Output

- In console we use print statement to display the output.

```
    print("\nPredicted Scores to get in Final Exam:")
    for i in range(6):
        try:
            print(['PASS','B','B+','A','A+','O'][i],f": {res[i]:.2f}")
        except ValueError:
            print(['PASS','B','B+','A','A+','O'][i],f": {res[i]}")
except ValueError:
    print("Invalid Input")
```

```
Predicted Scores to get in Final Exam:
PASS : 35.07
B : 36.73
B+ : 53.40
A : 70.07
A+ : 86.73
O : IMPOSSIBLE
```

GUI : Output



- We can use the same Label widget to display the output.
- As our output is quiet long we use Text widget to display.

```
pre_output = tk.Text(gp_frame, height = 10, width = 50, state = tk.DISABLED)
pre_output.grid(row = 8, column = 0, columnspan=2, padx = 5 , pady = 5)
```

- We use state = tk.DISABLED to disable a widget from user interaction. In this case is done to use the Text widget as output.

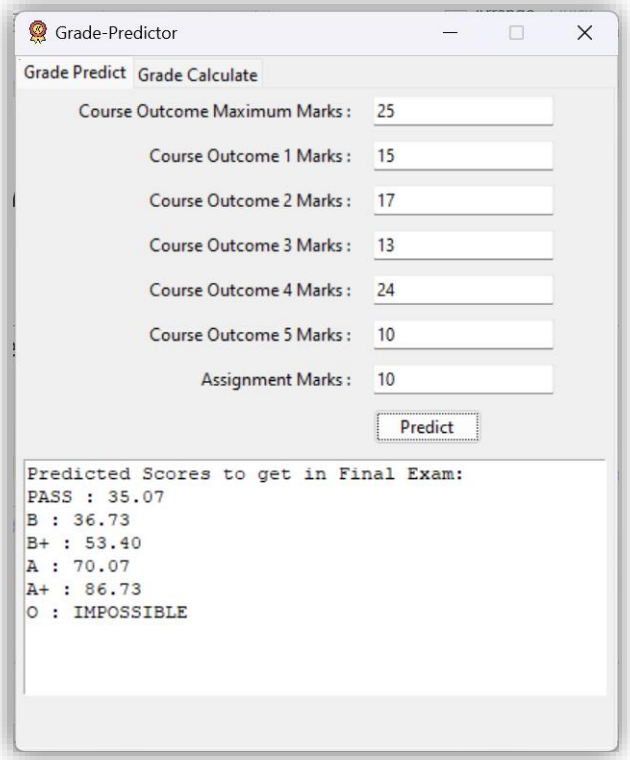
NOTE
We use columnspan parameter in grid() method to merge cells in a grid structure.

GUI : Printing Output

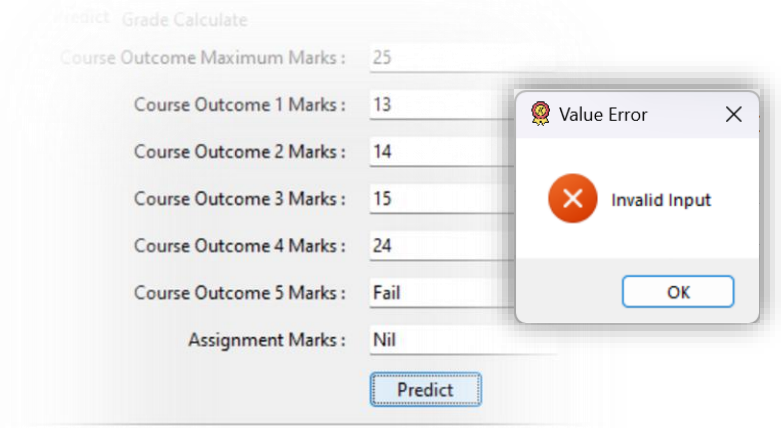
NOTE
We use delete() method to clear the contents present in Text widget.

- We use insert() method to print text into the Text widget.
- We cannot insert text to a disabled Text widget.
- So we change to normal state using configure method.

```
pre_output.configure(state=tk.NORMAL)
pre_output.delete(1.0,tk.END)
pre_output.insert(tk.END, res_txt)
pre_output.configure(state=tk.DISABLED)
```



Exceptional Handling



Console

- We print a suitable error message instead of raising error in this example.

```
except ValueError:
    print("Invalid Input")
```

```
Enter the Course Outcome 3 Marks      : 16
Enter the Course Outcome 4 Marks      : 24
Enter the Course Outcome 5 Marks      : fail
Invalid Input


Grade-Predictor
```

GUI

- We display the error as a popup window using messagebox class.

```
except ValueError:
    msgbox.showerror(title='Value Error', message="Invalid Input")
    return
res_txt = "Predicted Scores to get in Final Exam:\n"
for i in range(6):
```

```
import tkinter.messagebox as msgbox


msgbox.showerror(title='Value Error',
message="Invalid Input")
```

Summary of the GUI Program

Explaining app.py

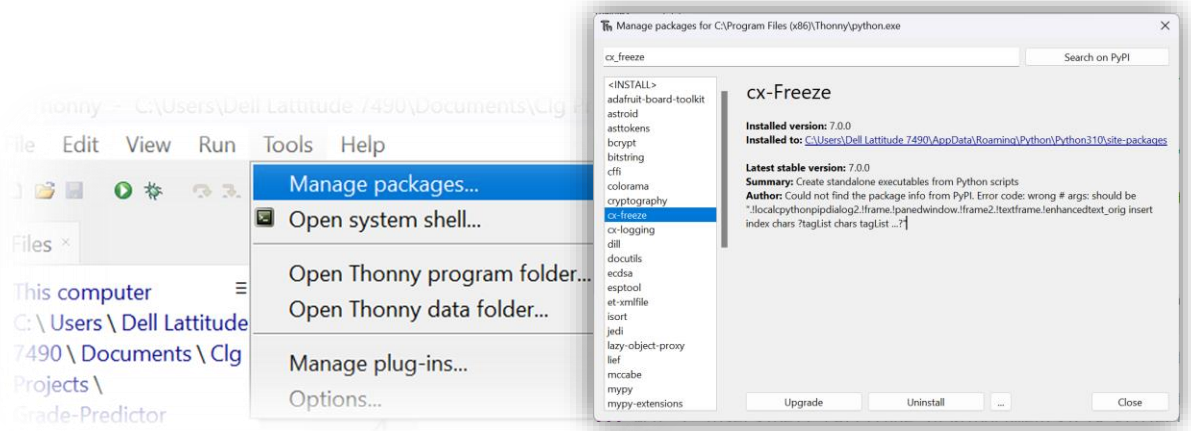
Creating Standalone Executables

- Using cx_freeze package we can make our GUI program into a standalone Executable.

pip install cx-Freeze

NOTE
For thonny users you can install packages from manage package window inside tools menu.

- Use the above command in terminal to install cx_freeze package.
- Next we have to create a setup.py…



Creating setup.py

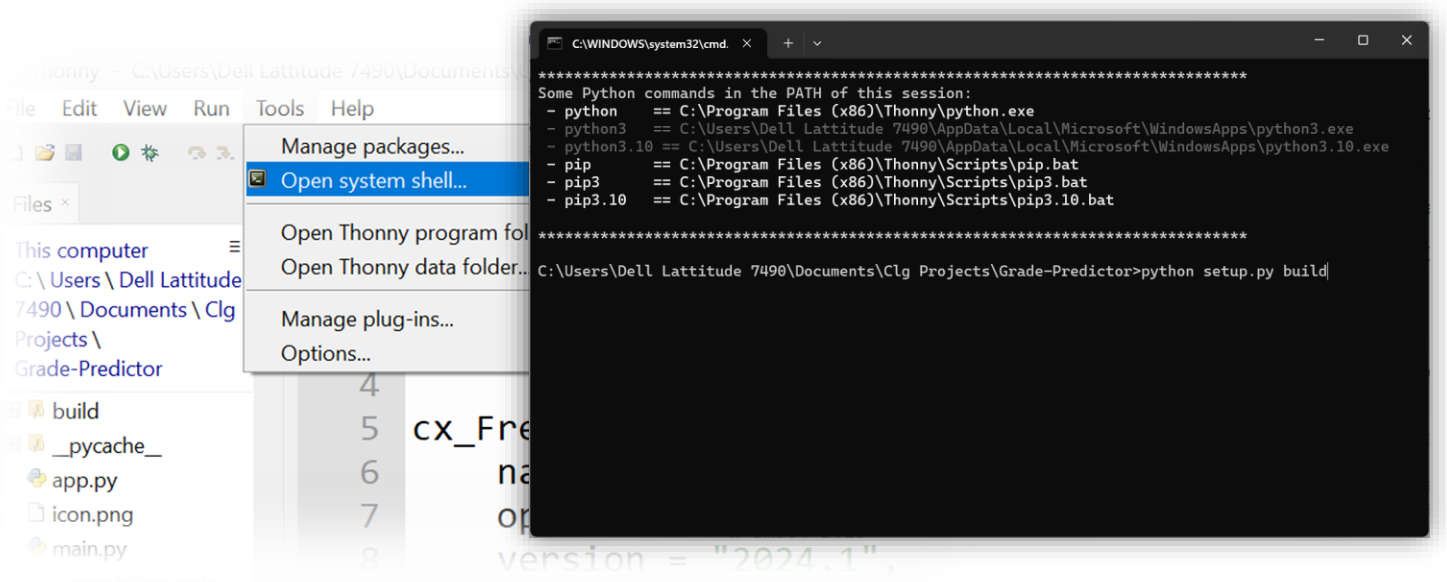
```
import cx_Freeze

executables = [cx_Freeze.Executable("app.py", base='gui', icon="icon.png")]

cx_Freeze.setup( name = "Grade-Predictor",
   options = {"build_exe": {"packages":["tkinter"],
              "include_files":["icon.png","main.py"]}},
    version = "2024.1",
    description = "Predicting and calculating grades for college courses.",
    executables = executables
    )
```

Building Executable

- Just run the code in terminal below you will be able to see a folder build created inside which the standalone executable is generated.

python setup.py build

NOTE
For thonny users you can access terminal from open system shell window inside tools menu.

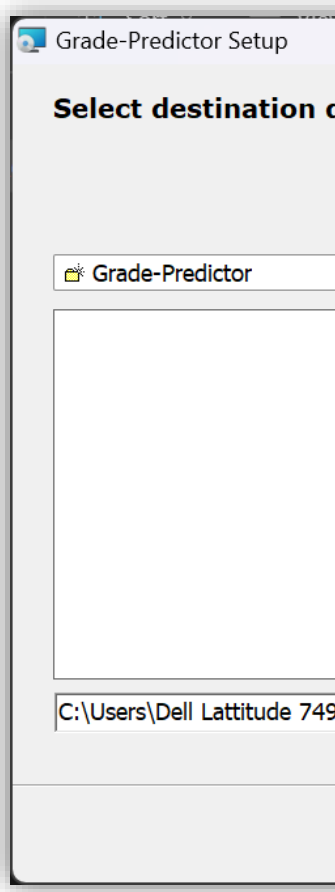


Creating Packages/Installers

- Instead of creating the executables directly we can use the below code to create packages or installers for distribution.
- It will created under dist folder.
- It makes the executable more portable.

python setup.py <command>

Command	Description
bdist_appimage	AppImage application bundle (`.AppImage`)
bdist_deb	DEB distribution (`.deb`)
bdist_dmg	DMG disk image (`.dmg`)
bdist_mac	Mac application bundle (`.app`)
bdist_msi	Windows installer (`.msi`)
bdist_rpm	RPM distribution (`.rpm`)

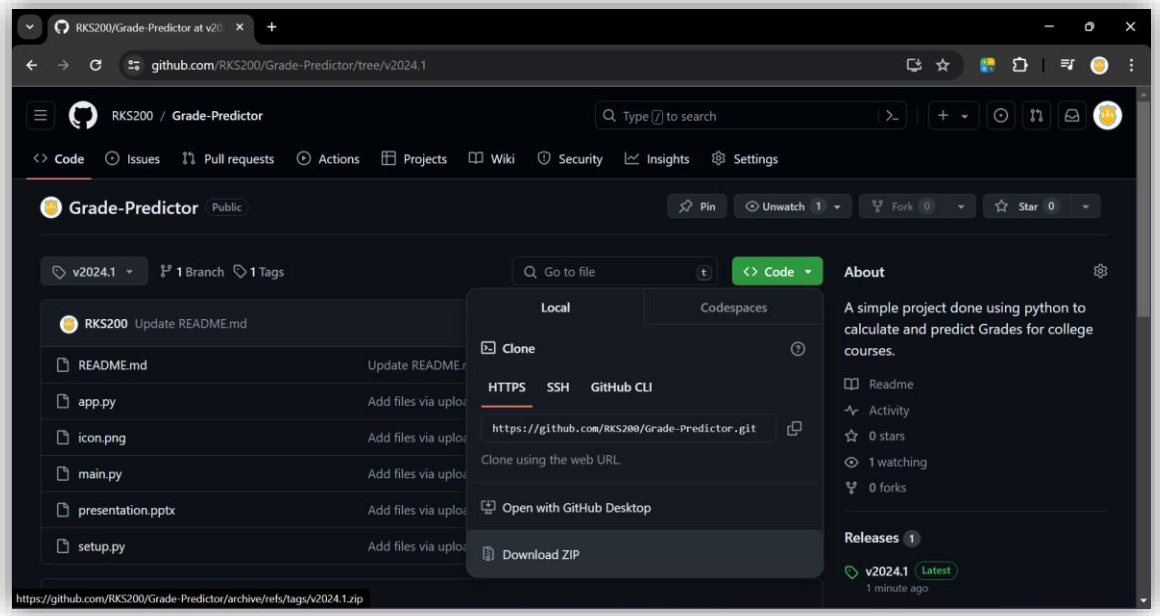


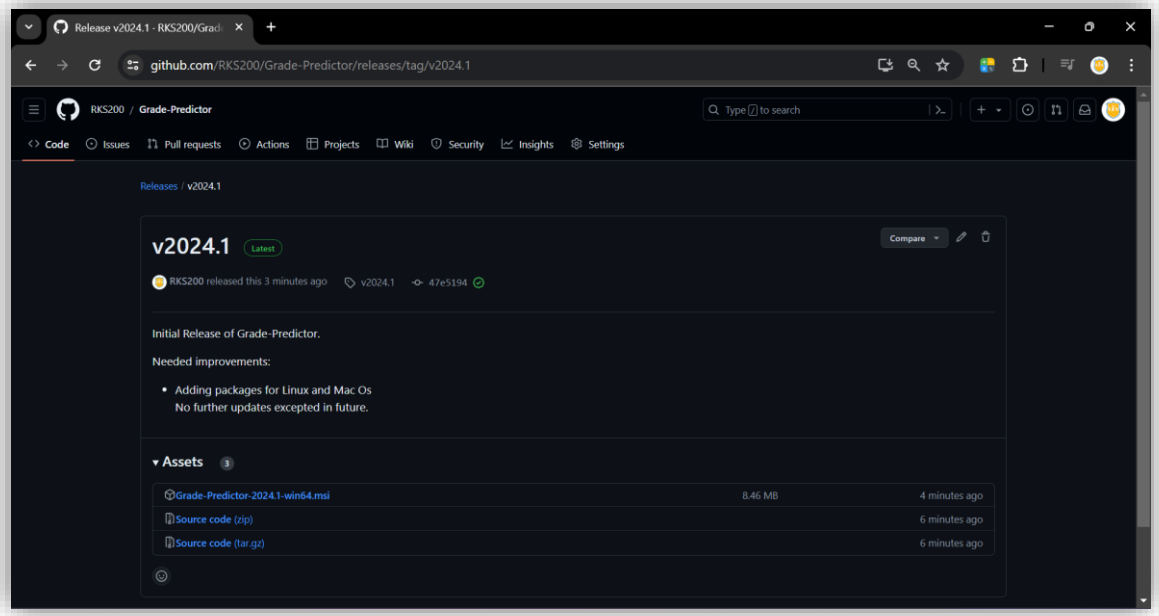
Still more …

- There are still more widgets like Menubar , RadioButton and ProgressBar that cannot be explained with this example.
- You can use websites like geeksforgeeks.org/python-tkinter-tutorial, tutorialspoint.com/python/python_gui_programming.htm to learn and explore more widgets and features of tkinter.

Source Code

- Goto github.com/RKS200/Grade-Predictor for downloading the source code.
- You can also head to Releases to download the packages / installer of the program.





Thank You