

Assignment 17 – GUI

Graphical User Interfaces

So far, we've only created console applications but in this assignment, we'll be creating a graphical user interface applications and we will start with an app that calculates miles per gallon.

Creating the Root Window

This will be the root window where the application will be displayed

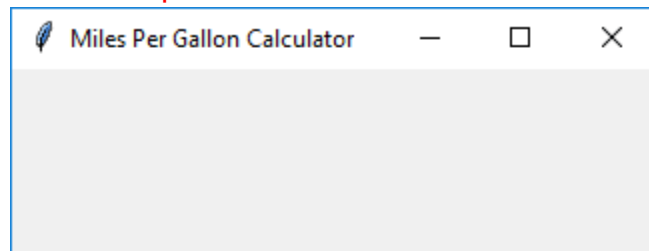
1. Create a root window as follows:

Screen Capture #1

```
1      #!/usr/bin/env python3
2
3      import tkinter as tk
4
5      if __name__ == "__main__":
6          # Create the root window
7          root = tk.Tk()
8
9          # Add a title to the root
10         root.title("Miles Per Gallon Calculator")
11
12         # Display the root window
13         root.mainloop()
14
```

2. Run the code to test

Screen Capture #2



Adding Frames and Components

To work with components like frames, labels, buttons, etc., we'll need to import the `ttk` module from `tkinter`. Once we add the frame, we can then add it to the root, and from there start adding the components.

We will create a module for the GUI setup

3. Code as follows:

Screen Capture #3

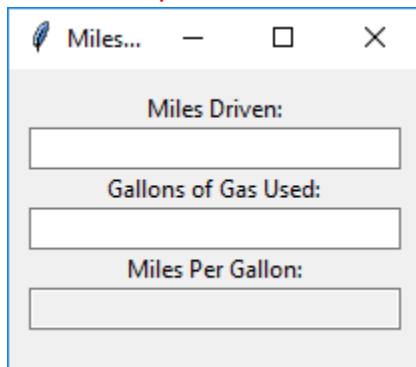
```

1  #!/usr/bin/env python3
2
3  import tkinter as tk
4  from tkinter import ttk
5
6
7  class MPGFrame(ttk.Frame):
8      def __init__(self, parent):
9          ttk.Frame.__init__(self, parent, padding="10 10 10 10")
10         self.pack()
11
12         # Define string variables for text entry fields
13         self.milesDriven = tk.StringVar()
14         self.gallonsUsed = tk.StringVar()
15         self.milesPerGallon = tk.StringVar()
16
17         # Display the components
18         ttk.Label(self, text="Miles Driven:").pack()
19         ttk.Entry(self, width=30, textvariable=self.milesDriven).pack()
20
21         ttk.Label(self, text="Gallons of Gas Used:").pack()
22         ttk.Entry(self, width=30, textvariable=self.gallonsUsed).pack()
23
24         ttk.Label(self, text="Miles Per Gallon:").pack()
25         ttk.Entry(self, width=30, textvariable=self.milesPerGallon,
26                   state="readonly").pack()
27
28
29  if __name__ == "__main__":
30      root = tk.Tk()
31      root.title("Miles Per Gallon Calculator")
32      MPGFrame(root)
33      root.mainloop()

```

- Once again, run the code to test

Screen Capture #4



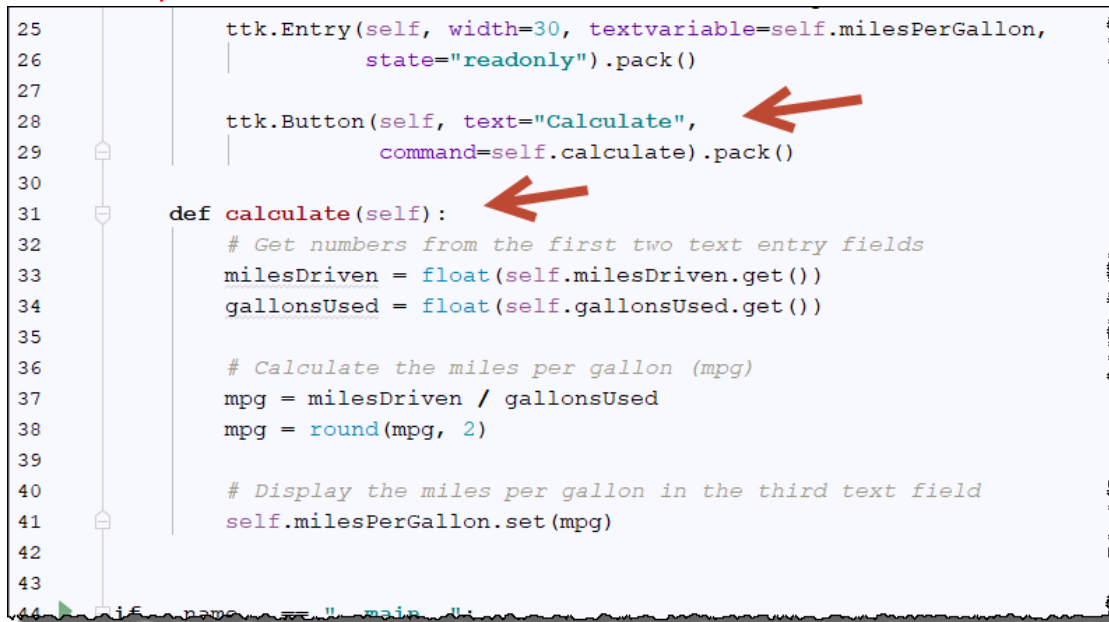
A screenshot of a Tkinter window titled "Miles...". The window contains three vertically stacked input fields. The first field is labeled "Miles Driven:", the second is labeled "Gallons of Gas Used:", and the third is labeled "Miles Per Gallon:". Each label is positioned above its corresponding text entry widget.

Add the Button and Handling the Click Event

When the button is clicked, we want to be able to perform some kind of action. To trigger the action, we use the command parameter.

- Code as follows:

Screen Capture #5



```

25         ttk.Entry(self, width=30, textvariable=self.milesPerGallon,
26                    state="readonly").pack()
27
28         ttk.Button(self, text="Calculate",
29                    command=self.calculate).pack()
30
31     def calculate(self):
32         # Get numbers from the first two text entry fields
33         milesDriven = float(self.milesDriven.get())
34         gallonsUsed = float(self.gallonsUsed.get())
35
36         # Calculate the miles per gallon (mpg)
37         mpg = milesDriven / gallonsUsed
38         mpg = round(mpg, 2)
39
40         # Display the miles per gallon in the third text field
41         self.milesPerGallon.set(mpg)
42
43
44 if __name__ == "__main__":

```

The screenshot shows a code editor with Python code. Two red arrows point to the `command=self.calculate` parameter in the `ttk.Button` call (line 29) and the `def calculate(self):` method definition (line 31). The code implements a `calculate` method that retrieves values from the first two text entry fields, calculates the miles per gallon (mpg), rounds it to two decimal places, and updates the third text field.

6. Once again, run the code to test. Make sure you add the values and click calculate

Screen Capture #6

A screenshot of a Tkinter window titled "Miles...". The window contains three text entry fields stacked vertically. The first field is labeled "Miles Driven:" and contains the text "100". The second field is labeled "Gallons of Gas Used:" and contains the text "10". The third field is labeled "Miles Per Gallon:" and contains the text "10.0". Below the third field is a button labeled "Calculate".

Working with a Grid Layout

7. Code as follows:

Screen Capture #7

```

15         self.milesPerGallon = tk.StringVar()
16
17         # Display the grid of components
18         ttk.Label(self, text="Miles Driven:").grid(
19             column=0, row=0, sticky=tk.E)
20         ttk.Entry(self, width=30, textvariable=self.milesDriven).grid(
21             column=1, row=0)
22
23         ttk.Label(self, text="Gallons of Gas Used:").grid(
24             column=0, row=1, sticky=tk.E)
25         ttk.Entry(self, width=30, textvariable=self.gallonsUsed).grid(
26             column=1, row=1)
27
28         ttk.Label(self, text="Miles Per Gallon:").grid(
29             column=0, row=2, sticky=tk.E)
30         ttk.Entry(self, width=30, textvariable=self.milesPerGallon,
31             state="readonly").grid(
32             column=1, row=2)
33
34         ttk.Button(self, text="Calculate",
35             command=self.calculate).grid(
36             column=1, row=3, sticky=tk.E)
37
38         # Add padding to all components
39         for child in self.winfo_children():
40             child.grid_configure(padx=5, pady=3)
41
42         def calculate(self):
43             # Get numbers from the first two text entry fields

```

8. Once again, run the code to test

Screen Capture #8

Future Value Program

Next, we'll create an application that will calculate the future value.

Business Class

We'll first create a class which will contain all the business logic for our application.

9. Add a new python file named **business.py**.
 10. Code as follows:

Screen Capture #9

```

1  class Investment:
2      def __init__(self):
3          self.monthly_investment = 0
4          self.yearly_interest_rate = 0
5          self.years = 0
6
7      def calculate_future_value(self):
8          monthly_interest_rate = self.yearly_interest_rate / 12 / 100
9          months = self.years * 12
10
11         future_value = 0
12         for i in range(months):
13             future_value += self.monthly_investment
14             monthly_interest_amount = future_value * monthly_interest_rate
15             future_value += monthly_interest_amount
16
17         return future_value
18

```

11. Next, create a ui.py file. This is where we will build our user interface.

12. Start by creating the structure.

```

1  #!/usr/bin/env python3
2
3  import tkinter as tk
4  from tkinter import ttk
5  import locale
6
7  from business import Investment
8
9
10 if __name__ == "__main__":
11     root = tk.Tk()
12     root.title("Future Value Calculator")
13     root.mainloop()
14

```

13. Now add the class for the frame layout

```

9
10 class FutureValueFrame(ttk.Frame):
11     def __init__(self, parent):
12         ttk.Frame.__init__(self, parent, padding="10 10 10 10")
13         self.parent = parent
14         self.investment = Investment()
15
16         # Set Locale
17         result = locale.setlocale(locale.LC_ALL, '')
18         if result[0] == 'C':
19             locale.setlocale(locale.LC_ALL, 'en_US')
20
21         # Define string variables for text entry fields
22         self.monthly_investment = tk.StringVar()
23         self.yearly_interest_rate = tk.StringVar()
24         self.years = tk.StringVar()
25         self.futureValue = tk.StringVar()
26
27
28 if __name__ == "__main__":
29     root = tk.Tk()
30     root.title("Future Value Calculator")
31     FutureValueFrame(root)
32     root.mainloop()
33

```



14. Add the components for our layout

```

25         self.futureValue = tk.StringVar()
26
27     def init_components(self):
28
29     def init_components(self):
30         self.pack()
31
32         # Display the grid of labels and text entry fields
33         ttk.Label(self, text="Monthly Investment:").grid(
34             column=0, row=0, sticky=tk.E)
35         ttk.Entry(self, width=25, textvariable=self.monthly_investment).grid(
36             column=1, row=0)
37
38         ttk.Label(self, text="Yearly Interest Rate:").grid(
39             column=0, row=1, sticky=tk.E)
40         ttk.Entry(self, width=25, textvariable=self.yearly_interest_rate).grid(
41             column=1, row=1)
42
43         ttk.Label(self, text="Years:").grid(
44             column=0, row=2, sticky=tk.E)
45         ttk.Entry(self, width=25, textvariable=self.years).grid(
46             column=1, row=2)
47
48         ttk.Label(self, text="Future Value:").grid(
49             column=0, row=3, sticky=tk.E)
50         ttk.Entry(self, width=25, textvariable=self.futureValue,
51                 state="readonly").grid(column=1, row=3)
52
53
54 if __name__ == "__main__":
55     root = tk.Tk()

```

15. Next add the buttons

a. Note: We will add the calculate method next

```

50         ttk.Entry(self, width=25, textvariable=self.futureValue,
51                 state="readonly").grid(column=1, row=3)
52
53     def make_buttons(self):
54
55     def make_buttons(self):
56         # Create a frame to store the two buttons
57         button_frame = ttk.Frame(self)
58
59         # Add the button frame to the bottom row of the main grid
60         button_frame.grid(column=0, row=4, columnspan=2, sticky=tk.E)
61
62         # Add two buttons to the button frame
63         ttk.Button(button_frame, text="Calculate", command=self.calculate) \
64             .grid(column=0, row=0, padx=5)
65         ttk.Button(button_frame, text="Exit", command=self.parent.destroy) \
66             .grid(column=1, row=0)
67
68
69 if __name__ == "__main__":
70     root = tk.Tk()
71     root.title("Future Value Calculator")

```

16. Add the calculate method

```

65         ttk.Button(button_frame, text="Exit", command=self.parent.destroy) \
66             .grid(column=1, row=0)
67
68     def calculate(self):
69         self.investment.monthly_investment = float(
70             self.monthly_investment.get())
71         self.investment.yearly_interest_rate = float(
72             self.yearly_interest_rate.get())
73         self.investment.years = int(
74             self.years.get())
75
76         self.futureValue.set(locale.currency(
77             self.investment.calculate_future_value(), grouping=True))
78
79
80     if __name__ == "__main__":
81         root = tk.Tk()

```

17. And we'll finish off by adding some padding to the components

```

52
53         self.make_buttons()
54
55         for child in self.winfo_children():
56             child.grid_configure(padx=5, pady=3)
57
58     def make_buttons(self):
59         # Create a frame to store the two buttons
60         button_frame = ttk.Frame(self)

```

18. Now run and test:

Screen Capture #10

Future Value Calculator

Monthly Investment: 100

Yearly Interest Rate: 5.0

Years: 5

Future Value: \$6,828.94

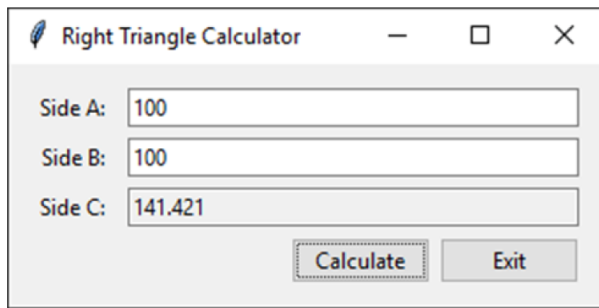
Calculate Exit

Extra Credit

To get full points for each extra credit, you must include screen captures of the running output as well as the python (.py) code files.

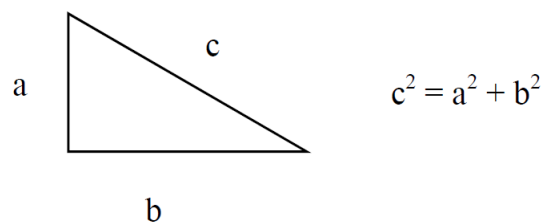
Extra Credit #1 – Right Triangle Calculator (+2 Extra Credit)

Create a GUI program that calculates the hypotenuse of a right triangle after the user enters the lengths of the two short sides and clicks the Calculate button.



Specifications:

- Use the Pythagorean Theorem to calculate the length of the third side. The Pythagorean Theorem states that the square of the hypotenuse of a right-triangle is equal to the sum of the squares of the opposite sides:



- As a result, you can calculate side C like this:
`c = square_root(a2 + b2)`
- Side C should be rounded to a maximum of 3 decimal places.