Assignment 17 – GUI

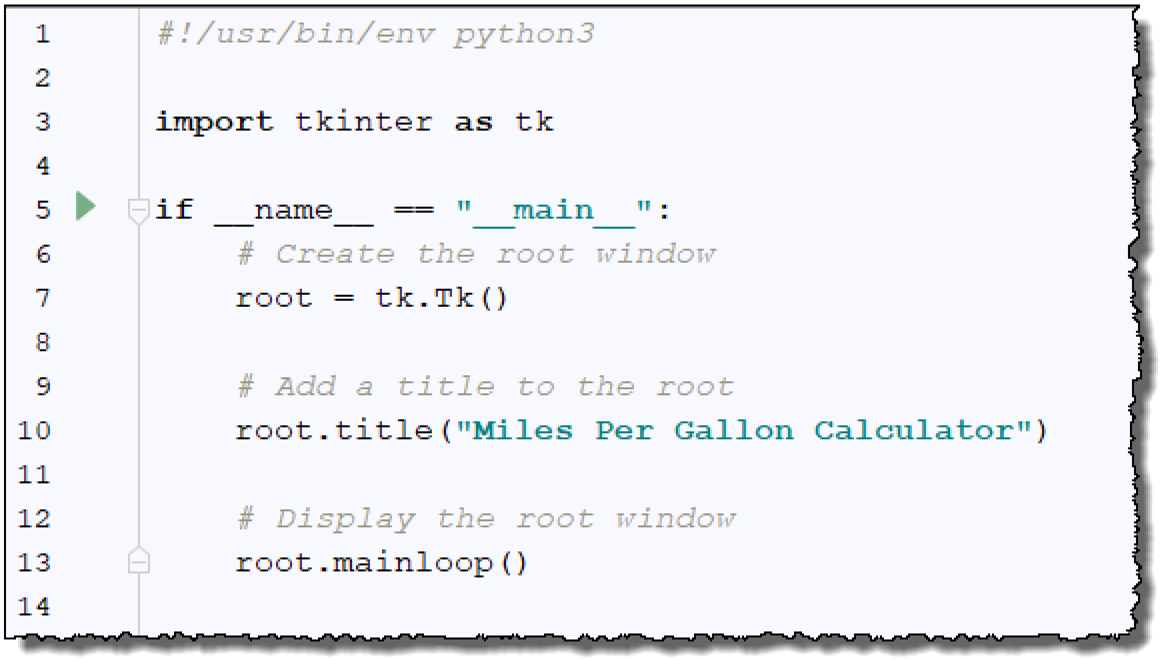
Graphical User Interfaces

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

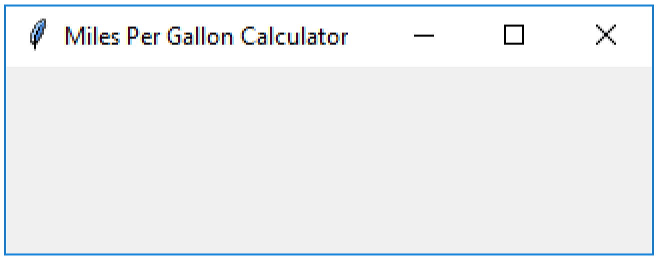
This will be the root window where the application will be displayed

1. Create a root window as follows:
   Screen Capture #1



```
#!/usr/bin/env python3

import tkinter as tk

if __name__ == "__main__":
    # Create the root window
    root = tk.Tk()

    # Add a title to the root
    root.title("Miles Per Gallon Calculator")

    # Display the root window
    root.mainloop()

```

2. Run the code to test
   Screen Capture #2

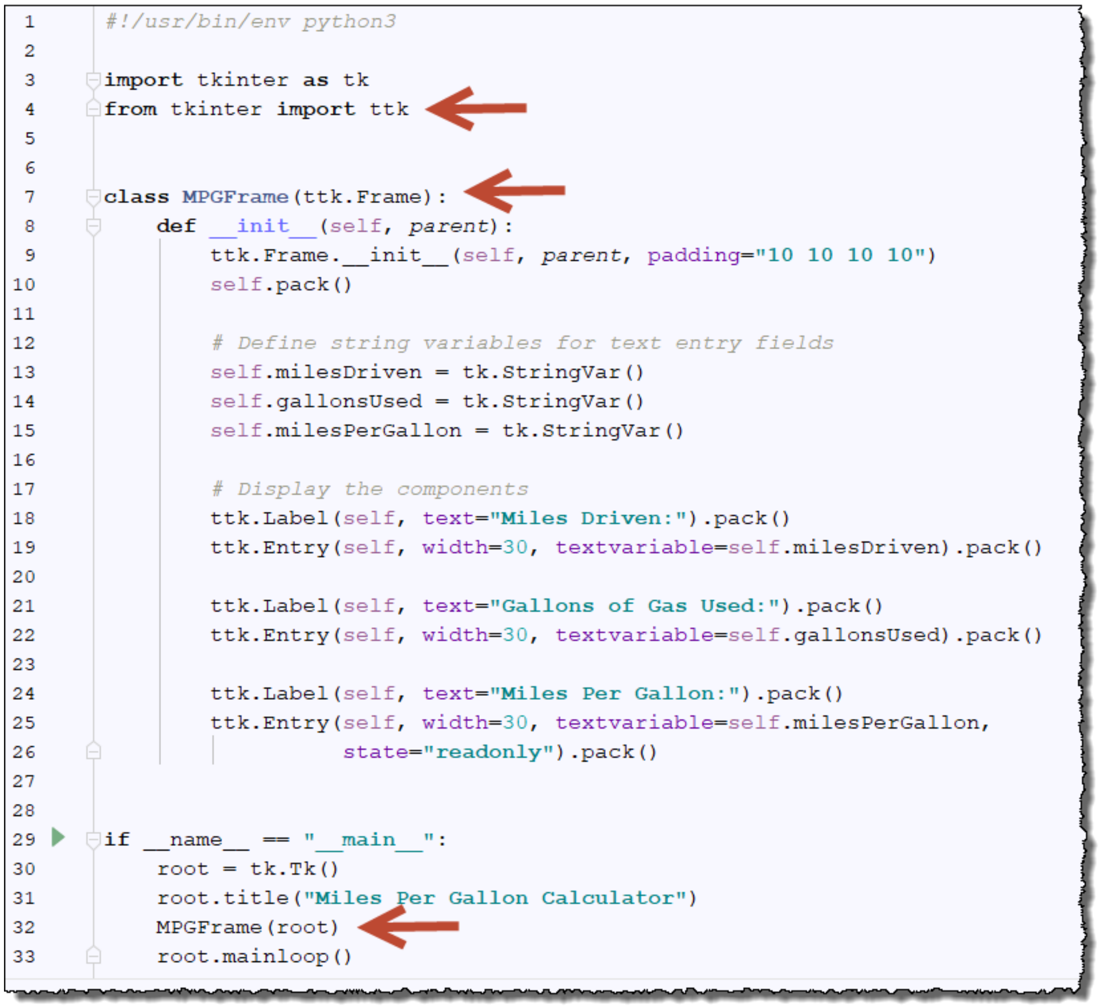


Adding Frames and Components

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

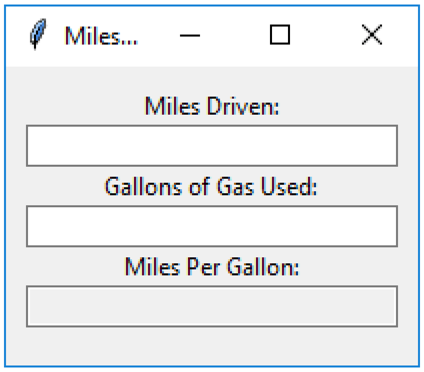
3. Code as follows:
   Screen Capture #3



```
#!/usr/bin/env python3

import tkinter as tk
from tkinter import ttk


class MPGFrame(ttk.Frame):
    def __init__(self, parent):
        ttk.Frame.__init__(self, parent, padding="10 10 10 10")
        self.pack()

        # Define string variables for text entry fields
        self.milesDriven = tk.StringVar()
        self.gallonsUsed = tk.StringVar()
        self.milesPerGallon = tk.StringVar()

        # Display the components
        ttk.Label(self, text="Miles Driven:").pack()
        ttk.Entry(self, width=30, textvariable=self.milesDriven).pack()

        ttk.Label(self, text="Gallons of Gas Used:").pack()
        ttk.Entry(self, width=30, textvariable=self.gallonsUsed).pack()

        ttk.Label(self, text="Miles Per Gallon:").pack()
        ttk.Entry(self, width=30, textvariable=self.milesPerGallon,
                  state="readonly").pack()


if __name__ == "__main__":
    root = tk.Tk()
    root.title("Miles Per Gallon Calculator")
    MPGFrame(root)
    root.mainloop()
```

4. Once again, run the code to test

Screen Capture #4

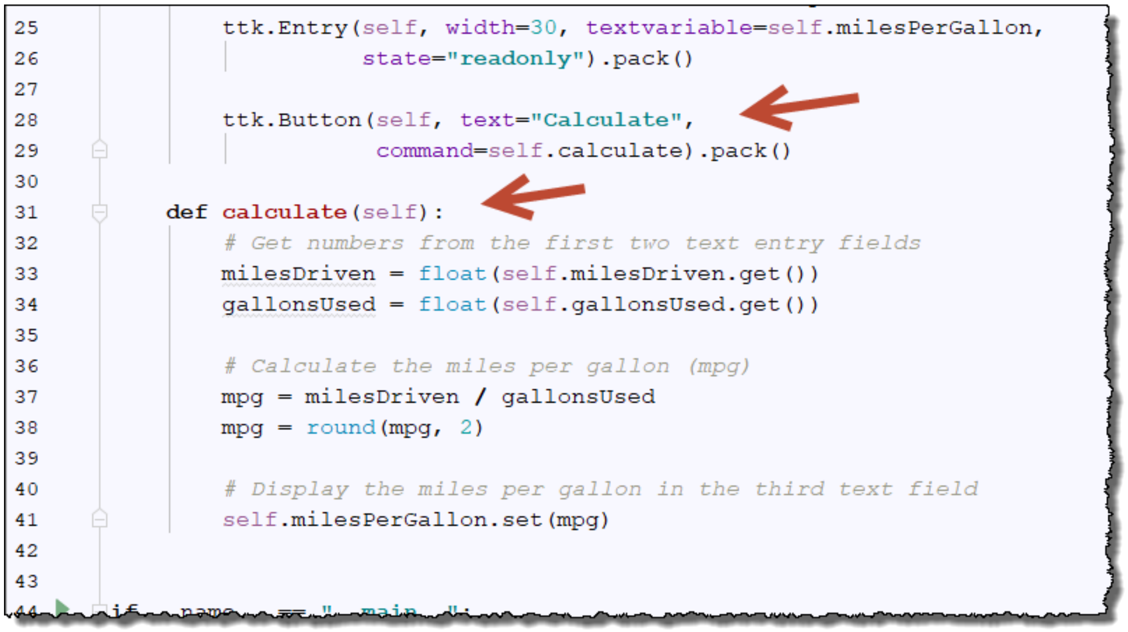


Add the Button and Handling the Click Event

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.

5. Code as follows:

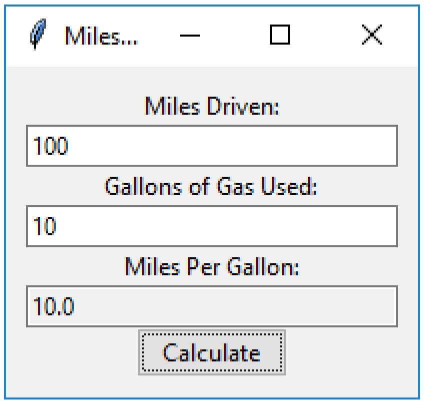
Screen Capture #5



```
        ttk.Entry(self, width=30, textvariable=self.milesPerGallon,
                  state="readonly").pack()

        ttk.Button(self, text="Calculate",
                   command=self.calculate).pack()

    def calculate(self):
        # Get numbers from the first two text entry fields
        milesDriven = float(self.milesDriven.get())
        gallonsUsed = float(self.gallonsUsed.get())

        # Calculate the miles per gallon (mpg)
        mpg = milesDriven / gallonsUsed
        mpg = round(mpg, 2)

        # Display the miles per gallon in the third text field
        self.milesPerGallon.set(mpg)


if __name__ == "__main__":
```

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

Screen Capture #6

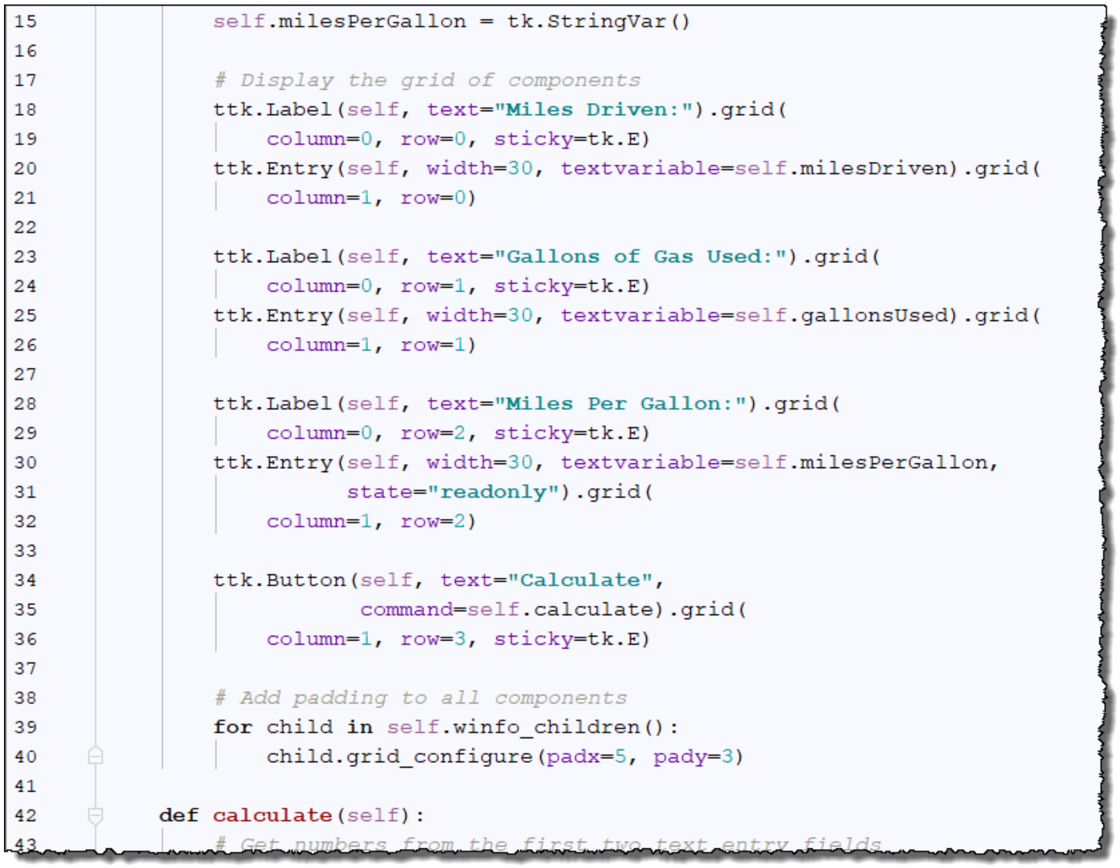


Miles Driven: 100
Gallons of Gas Used: 10
Miles Per Gallon: 10.0

Working with a Grid Layout

7. Code as follows:

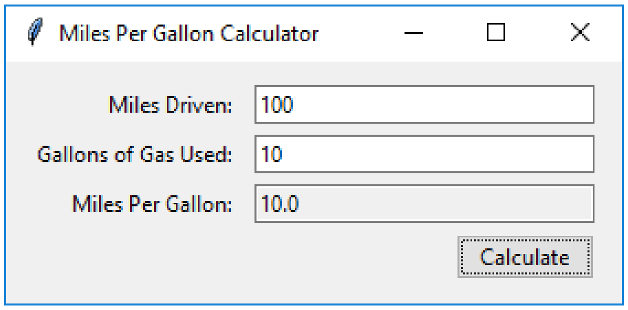
Screen Capture #7



```
        self.milesPerGallon = tk.StringVar()

        # Display the grid of components
        ttk.Label(self, text="Miles Driven:").grid(
            column=0, row=0, sticky=tk.E)
        ttk.Entry(self, width=30, textvariable=self.milesDriven).grid(
            column=1, row=0)

        ttk.Label(self, text="Gallons of Gas Used:").grid(
            column=0, row=1, sticky=tk.E)
        ttk.Entry(self, width=30, textvariable=self.gallonsUsed).grid(
            column=1, row=1)

        ttk.Label(self, text="Miles Per Gallon:").grid(
            column=0, row=2, sticky=tk.E)
        ttk.Entry(self, width=30, textvariable=self.milesPerGallon,
                  state="readonly").grid(
            column=1, row=2)

        ttk.Button(self, text="Calculate",
                   command=self.calculate).grid(
            column=1, row=3, sticky=tk.E)

        # Add padding to all components
        for child in self.winfo_children():
            child.grid_configure(padx=5, pady=3)

    def calculate(self):
        # Get numbers from the first two text entry fields
```

8. Once again, run the code to test

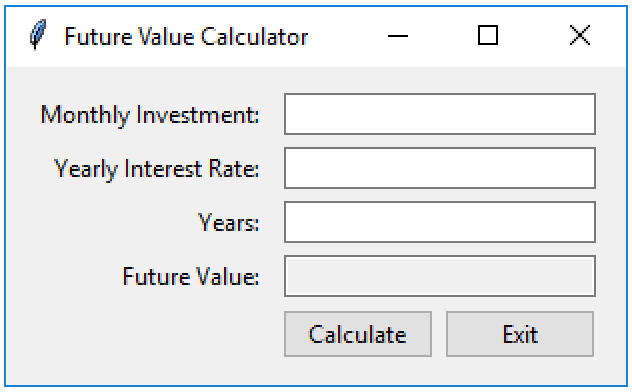
Screen Capture #8



Miles Driven: 100
Gallons of Gas Used: 10
Miles Per Gallon: 10.0

Future Value Program

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

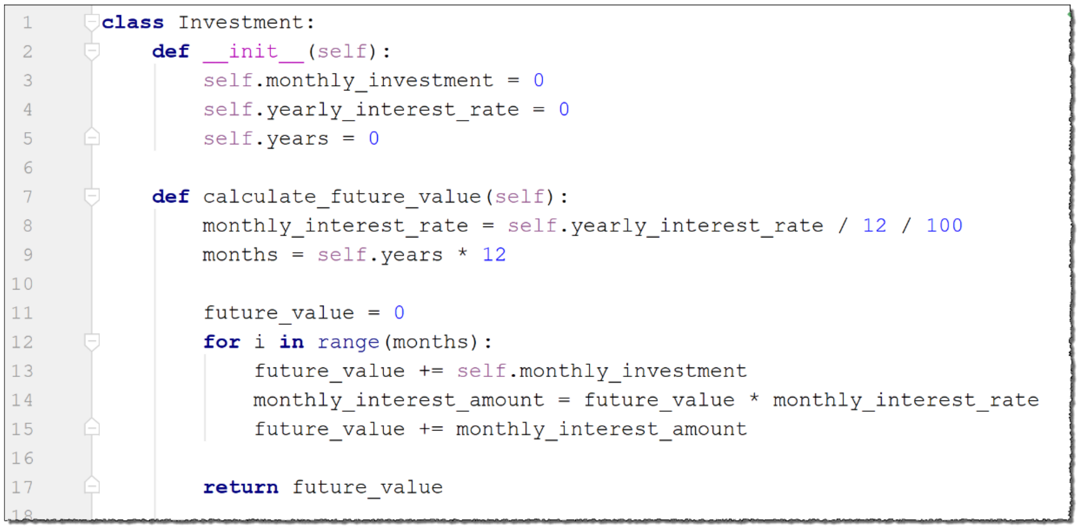


Business Class

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

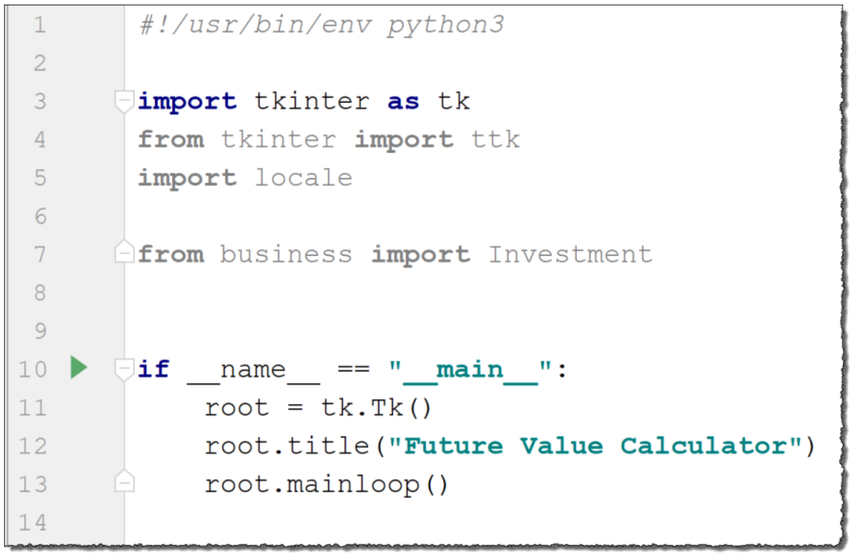
9. Add a new python file named **business.py.**
10. Code as follows:

Screen Capture #9



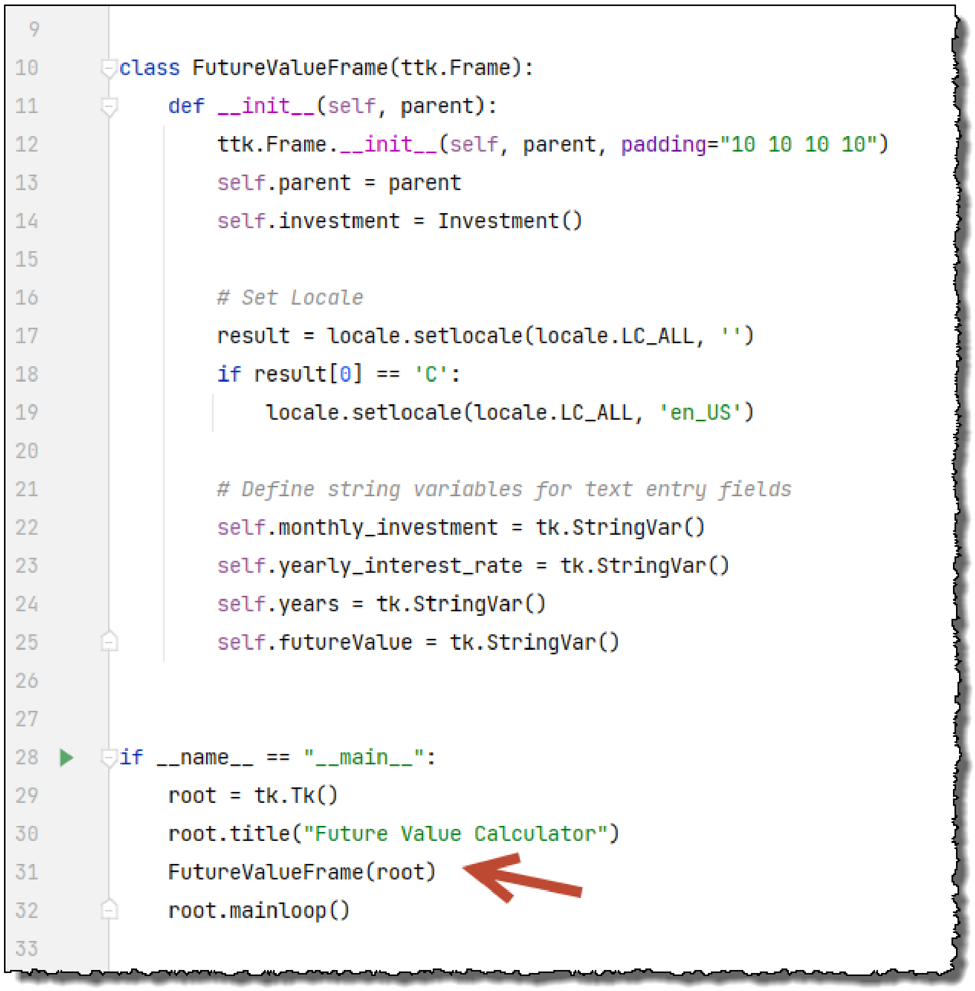
```python
class Investment:
    def __init__(self):
        self.monthly_investment = 0
        self.yearly_interest_rate = 0
        self.years = 0

    def calculate_future_value(self):
        monthly_interest_rate = self.yearly_interest_rate / 12 / 100
        months = self.years * 12

        future_value = 0
        for i in range(months):
            future_value += self.monthly_investment
            monthly_interest_amount = future_value * monthly_interest_rate
            future_value += monthly_interest_amount

        return future_value
```

11. Next, create a ui.py file. This is where we will build our user interface.
12. Start by creating the structure.



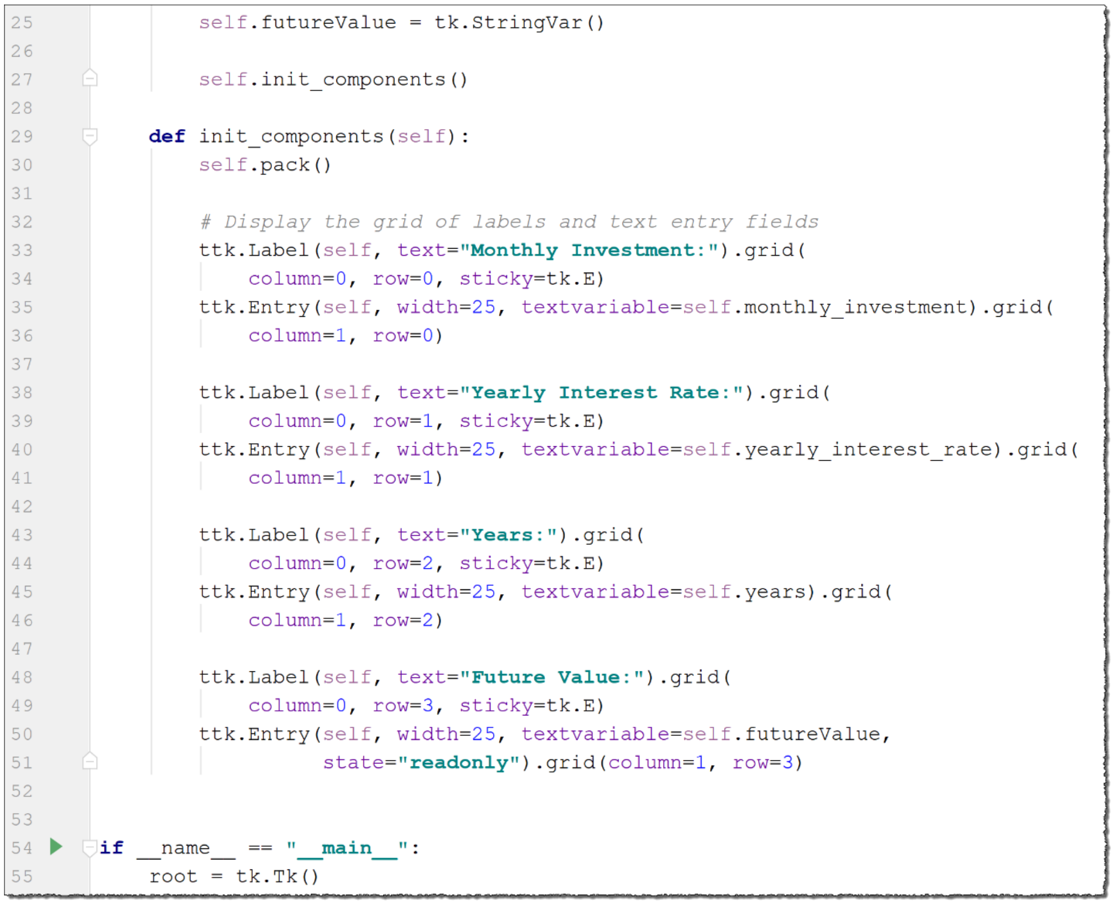
```python
#!/usr/bin/env python3

import tkinter as tk
from tkinter import ttk
import locale

from business import Investment


if __name__ == "__main__":
    root = tk.Tk()
    root.title("Future Value Calculator")
    root.mainloop()
```

13. Now add the class for the frame layout



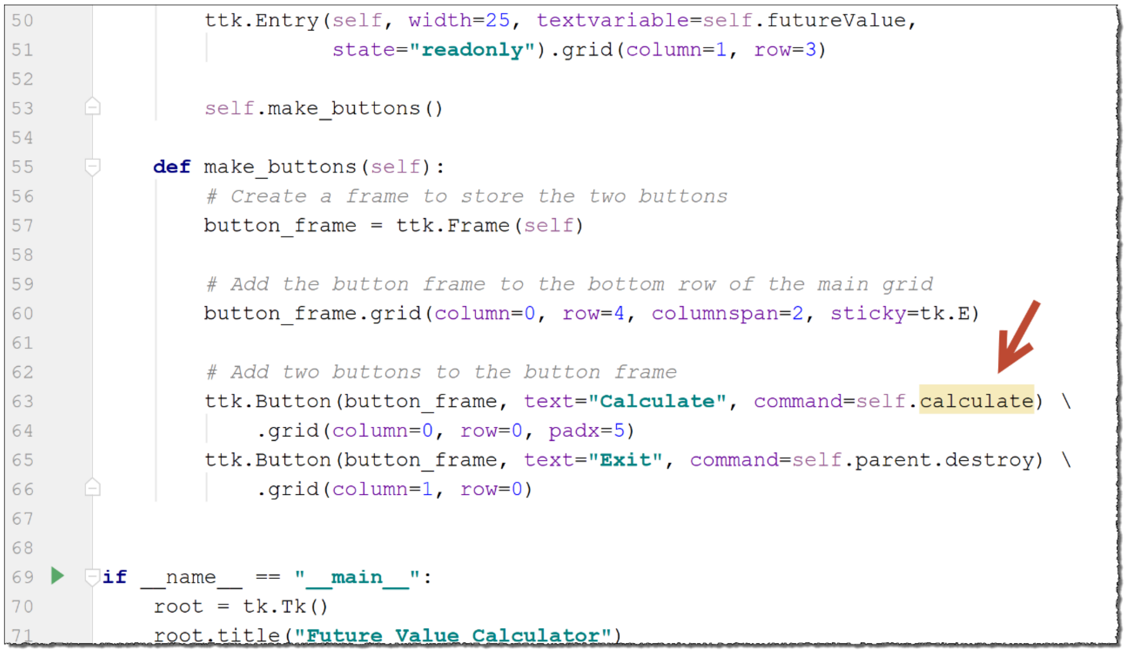
```python

class FutureValueFrame(ttk.Frame):
    def __init__(self, parent):
        ttk.Frame.__init__(self, parent, padding="10 10 10 10")
        self.parent = parent
        self.investment = Investment()

        # Set Locale
        result = locale.setlocale(locale.LC_ALL, '')
        if result[0] == 'C':
            locale.setlocale(locale.LC_ALL, 'en_US')

        # Define string variables for text entry fields
        self.monthly_investment = tk.StringVar()
        self.yearly_interest_rate = tk.StringVar()
        self.years = tk.StringVar()
        self.futureValue = tk.StringVar()


if __name__ == "__main__":
    root = tk.Tk()
    root.title("Future Value Calculator")
    FutureValueFrame(root)
    root.mainloop()
```

14. Add the components for our layout



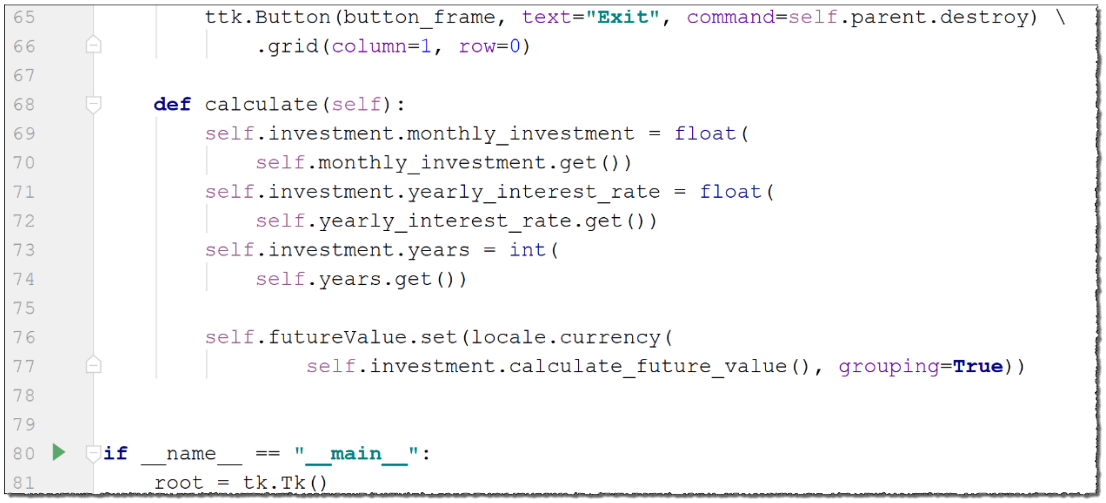
```
        self.futureValue = tk.StringVar()

        self.init_components()

    def init_components(self):
        self.pack()

        # Display the grid of labels and text entry fields
        ttk.Label(self, text="Monthly Investment:").grid(
            column=0, row=0, sticky=tk.E)
        ttk.Entry(self, width=25, textvariable=self.monthly_investment).grid(
            column=1, row=0)

        ttk.Label(self, text="Yearly Interest Rate:").grid(
            column=0, row=1, sticky=tk.E)
        ttk.Entry(self, width=25, textvariable=self.yearly_interest_rate).grid(
            column=1, row=1)

        ttk.Label(self, text="Years:").grid(
            column=0, row=2, sticky=tk.E)
        ttk.Entry(self, width=25, textvariable=self.years).grid(
            column=1, row=2)

        ttk.Label(self, text="Future Value:").grid(
            column=0, row=3, sticky=tk.E)
        ttk.Entry(self, width=25, textvariable=self.futureValue,
                  state="readonly").grid(column=1, row=3)


if __name__ == "__main__":
    root = tk.Tk()
```

15. Next add the buttons
    a. Note: We will add the calculate method next



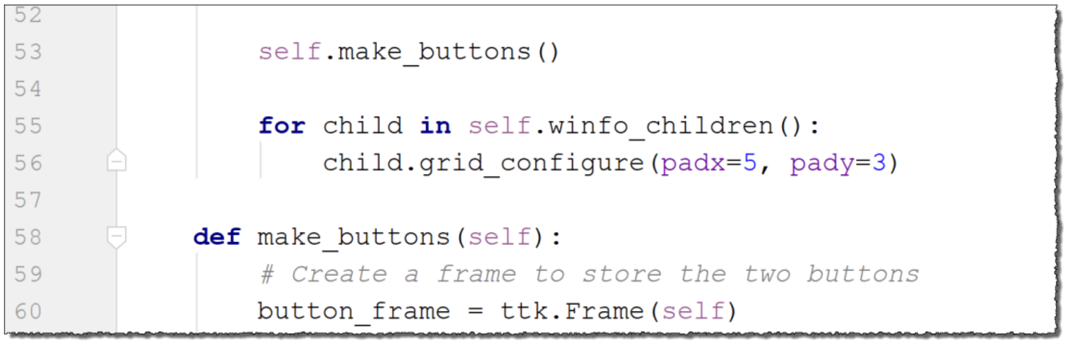
```
        ttk.Entry(self, width=25, textvariable=self.futureValue,
                  state="readonly").grid(column=1, row=3)

        self.make_buttons()

    def make_buttons(self):
        # Create a frame to store the two buttons
        button_frame = ttk.Frame(self)

        # Add the button frame to the bottom row of the main grid
        button_frame.grid(column=0, row=4, columnspan=2, sticky=tk.E)

        # Add two buttons to the button frame
        ttk.Button(button_frame, text="Calculate", command=self.calculate) \
            .grid(column=0, row=0, padx=5)
        ttk.Button(button_frame, text="Exit", command=self.parent.destroy) \
            .grid(column=1, row=0)


if __name__ == "__main__":
    root = tk.Tk()
    root.title("Future Value Calculator")
```

16. Add the calculate method



```
            ttk.Button(button_frame, text="Exit", command=self.parent.destroy) \
                .grid(column=1, row=0)

    def calculate(self):
        self.investment.monthly_investment = float(
            self.monthly_investment.get())
        self.investment.yearly_interest_rate = float(
            self.yearly_interest_rate.get())
        self.investment.years = int(
            self.years.get())

        self.futureValue.set(locale.currency(
                self.investment.calculate_future_value(), grouping=True))


if __name__ == "__main__":
    root = tk.Tk()
```

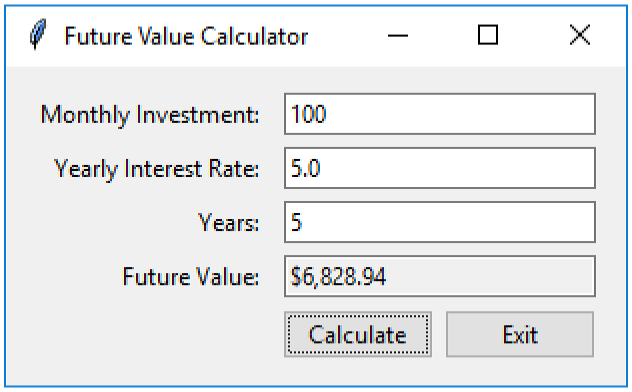
17. And we'll finish off by adding some padding to the components



```

        self.make_buttons()

        for child in self.winfo_children():
            child.grid_configure(padx=5, pady=3)

    def make_buttons(self):
        # Create a frame to store the two buttons
        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

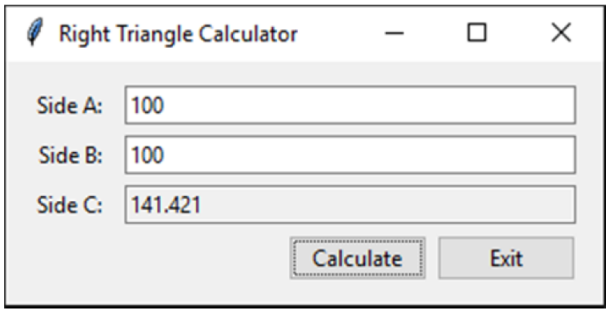
Future Value: $6,828.94

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)

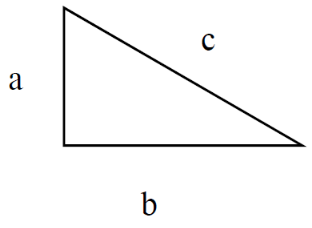
Create a GUI program that calculates the hypotenuse of a right triangle after the user enters the lengths of the two short sides and clicks the Calculate button.



Side A:	100
Side B:	100
Side C:	141.421

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:



$$c^2 = a^2 + b^2$$

- 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.