Program 01 - Annual Rainfall

You are asked to collect the monthly rainfall for one year and enter it into your computer program.

- Write a program to input and display rainfall for each month, accumulate and display the total inches of rain for the year and the average rainfall per month.
  - use a for loop to complete this program
  - use f-string for output formatting with proper pacing (\t) and 2 floating point decimals.
- check for errors on input and take appropriate action to ensure program does not abort. I will try to break it with bad input!!
- Be sure your program has intuitive input and output messages (follow example below).
- Name the program: **rainfall.py**
- Upload to Canvas exam link when complete and correct.

Example output:

```
Calculate total and average rain for a 12 month period
------------------------------------------------------------
Enter the rainfall amount for month 1: 1
Enter the rainfall amount for month 2: .5
Enter the rainfall amount for month 3: 1
Enter the rainfall amount for month 4: .5
Enter the rainfall amount for month 5: 1
Enter the rainfall amount for month 6: .5
Enter the rainfall amount for month 7: 1
Enter the rainfall amount for month 8: .5
Enter the rainfall amount for month 9: 1
Enter the rainfall amount for month 10: .5
Enter the rainfall amount for month 11: 1
Enter the rainfall amount for month 12: .5
--------------------------------------------------
For 12 months
   Total rainfall:                9.00 inches
   Average monthly rainfall:      0.75 inches

Program Successfully Terminated
```

Program 02 – shell.py

Create a new tkinter program called shell.py with the code below.

```
from tkinter import *
from tkinter import ttk

def main():
    global label
    rootWindow = Tk()

    label = ttk.Label( rootWindow, text="Hello World!" )
    label.pack()

    rootWindow.mainloop()

main()
```

Run/test the program.

1. Add a Button

```
from tkinter import *
from tkinter import ttk

def main():
    rootWindow = Tk()

    label = ttk.Label( rootWindow, text="Hello World!" )
    label.pack()

    button1 = ttk.Button( rootWindow, text="Change Label" )
    button1.pack()

    rootWindow.mainloop()

main()
```

Run/test the code.

2. Add a new function to the program, and "attach" it to the button.

```
from tkinter import *
from tkinter import ttk

def change():
    print( "change function called" )

def main():
    global label
    rootWindow = Tk()

    label = ttk.Label( rootWindow, text="Hello World!" )
    label.pack()

    button1 = ttk.Button( rootWindow, text="Change Label",command=change )
    button1.pack()

    rootWindow.mainloop()

main()
```

Run/test your code (verify that when you click the button, something gets printed in the Console). Using the console is not necessarily something we want to do with GUI applications, but, in this case, the console is a good way to debug our program and verify that it works as expected.

3. Challenge #1: Add a Second Button

- Add a second button to your GUI. Call it button2.
- Add a new function and make the new button activate this function. Your new function will print some simple string on the console. Make sure the string and the function are different from the already defined change function, and the string it prints.

4. Button Changing The text of the Label

   - Make the button change the text of the label when it (the button) is clicked:

```
from tkinter import *
from tkinter import ttk

label = None # this variable will hold the label created by the GUI and will be accessible by the change1() function.

def change1():
        global label
        label.config( text = "Goodbye World!" )

def main():
      global label
      rootWindow = Tk()

      label = ttk.Label( rootWindow, text="Hello World!" )
      label.pack()

      button1 = ttk.Button( rootWindow, text="Bye!", command=change1 )
      button1.pack()

      rootWindow.mainloop()

main()
```

Run/test your code. Verify that clicking on the button makes the label change.

5. Challenge #2: Add a Second Button That Changes the Label, Too

   - Add a second button so that the first button changes the label to "Goodbye World!", and
   - the second button changes the label to "Hello World!".

6. Placing Widgets in a Grid -

   - all three widgets (label, button1 and button2) on the same line. This corresponds of a grid with 1 row and 3 columns. \

   - Only 3 lines have to change. They are shown below:

```
label.grid( row=0, column=0 )
button1.grid( row=0, column=1 )
button2.grid( row=0, column=2 )
```

• Make the modification and run/test your App. Do the 3 widgets display in a horizontal alignment?

7. Challenge #3: Reorganize the Widgets

    - Change the organization so that the label is on a row by itself, and the two buttons are next to each other on a row below the label.

8. Save this program (shell.py) so you can upload it to Canvas when you submit your test.

Program 03 – gui.py

Organizing the GUI as a Class

1. Create a python program with the following code and name it GUI.py

```
from tkinter import *
from tkinter import ttk

class GUI:
        def __init__( self, rootWindow ):
            self.label = ttk.Label( rootWindow, text="Hello World!" )
            self.label.grid( row=0, column=0 )

            self.button1 = ttk.Button( rootWindow, text="Hello", command=self.hello )
            self.button1.grid( row=0, column=1 )
            self.button2 = ttk.Button( rootWindow, text="Bye", command=self.bye )
            self.button2.grid( row=0, column=2 )

        def bye( self ):
            self.label.config( text = "Goodbye World!" )

        def hello( self ):
            self.label.config( text = "Hello World!" )

        def main():
            global label
            rootWindow = Tk()

            gui = GUI( rootWindow )
            rootWindow.mainloop()

main()
```

Run/Test code

2. Adding A Checkbox

- Add a checkbox to your GUI.py program.
- Add the code below to your constructor:

```
self.buttonsEnabled = IntVar()
self.buttonsEnabled.set( 1 )

self.check1 = ttk.Checkbutton( rootWindow, text="Enable", variable=self.buttonsEnabled )
self.check1.grid( row=1, column=0 )
```

Note: to change the value of the buttonsEnable object, which is instantiated from the class IntVar, we use its set() method. To get the value of such an object, there is another method, called get(), that we could use.

- Modify the bye() and hello() methods of your GUI class, as shown below:

```
def bye( self ):
    self.label.config( text = "Goodbye World!" )
    print( "buttonsEnabled = ", self.buttonsEnabled.get() )

def hello( self ):
    self.label.config( text = "Hello World!" )
    self.buttonsEnabled.set( 1-self.buttonsEnabled.get() )
```

Run/test program

3. Challenge #4: Controlling the Buttons with the Checkbox

- The real purpose of the checkbox, you will have probably guessed, is to control whether the buttons modify the label or not.
- We want the buttons to change the label only when the checkbox is checked, and not change the label when the checkbox is unchecked.
- Modify your GUI class so that you make it adopt this behavior.

Hint: you only need to add 1 if statement to each of the 2 class methods and then comment out the second line of each. **Reminder:** the value of a checkbox is true/false, 0/1, or on/off.

4. Adding a Text Area

Before adding the text area, on Row 0:

- First:
  - comment out your checkbox code (so I can see it for grading part 01).
  - copy your Bye button and code, paste it in and comment it out (so I can see it for grading part 01).
- Then
  - change the remaining Bye button and code to the destroy method.
  - change the label to "Text Editor".

You will add the text area on Row 1.

- Add these lines to the constructor of your GUI class.

```
self.text = Text( rootWindow, width=80, height=30, background="yellow" )
self.text.grid( row=1, column=0, columnspan=4 )
```

Run/test your code

- Click on the yellow-colored block and enter some text. Copy some text from some another window and paste it in the text-area of your GUI.

5. Clearing the Text Area

    - clearing the text area requires a simple command:

```
self.text.delete(1.0, END)
```

6. Challenge #5: Button to Clear the Text Area

    - Modify your GUI class, and change your Hello button, so that its text becomes "Clear", and its action clears the whole text area.
    - save this program (GUI.py) and upload it to Canvas when you submit your exam