At the start of each of these problems, the name of a Python file is given in **blue**: **foo.py**. You should create and save the requested Python program source code in a file with the same name. Also add a comment at the top of each giving your name.

When finished, upload each **.py** file with the specified name to the Canvas **H4 Assignment** link.  
  
No starting code for this assignment. Do each of the following. Create a new PyCharm project and create each of the following files within it, as described below. For any turtle graphics problem, be sure to have **wn.exitonclick()** as the last statement of your script.

The Canvas assignment with this posted handout also has links to help videos for these problems. Some of these might describe slightly different earlier versions of the problems, but they should help you in how to approach doing them.

**[H4-1]** (**revstr.py**) The code below uses the accumulator pattern to reverse a string **s**:

**reversed = ''**

**for char in s:**

**reversed = char + reversed**

(This is like **debug\_4.py** in **[L3-2]**.) Use this code to define a function **def reverse(st)** which returns the reversed version of **st** as its result. Then define a **main()** function which reads the string **s** from the user, and prints out the value returned by **reverse(s)**. Finally, call your **main()** function as the last line in your program.

**[H4-2]** (**test\_revstr.py**) Write PyTest unit tests for your **reverse()** function of the previous problem. You should test your function with at least ten (10) different unit test functions (name them **test\_**...), with asserts within each that test for 10 different input values. You must **copy** your **reverse()** function of **[H4-1]** into the **test\_revstr.py** file for your submission.

To avoid this copying, it's possible to **import revstr** for testing into your separate **test\_revstr** file, using the **if \_\_name\_\_ == '\_\_main\_\_':**  approach we'll discuss in class. Unfortunately, this won't work for Canvas submissions, since it renames the file names when I download them. So be **sure** to submit your **reverse()** function and your PyTest code both within the same **.py** file for this problem.

Remember that PyTest does **not** allow your imported module to read or write anything to the console: it will report such attempts as **failures** when running the tests.

**[H4-3]** (**ftoc.py**) Write a program that reads a temperature in degrees Fahrenheit **fahr**, then converts it to the equivalent degrees Celsius and prints the results like this: **32.0 degrees F is to 0.0 degrees C**.

Do this by defining the function **def f2c(fahr)** that computes and returns the Celsius equivalent of its Fahrenheit parameter **fahr**, **rounded to two decimal places**. (Use Python built-in round(vRecall the formula for this conversion: subtract **32.0** from degrees Fahrenheit, then multiply by **5/9**. Then define a **main()** method that reads a Fahrenheit **float** value from the user, calls your function, then prints out the result. You should call your **main()** function at the end of program.

**[H4-4]** (**test\_f2c.py**) Write PyTest unit tests for your **f2c()** function of the previous problem. You should test your function with at least ten (10) different unit test functions, with an **assert** within each that tests f2c() for a particular input value. You must copy your **f2c()** function of **[H4-3]** into the **test\_f2c.py** file. Be sure all of your tests pass!

**[H4-5]** (**htt6\_2.py**) Do Exercise 2 at the end of HTT Chapter 6, generalized as follows. Read the number **N** of nested squares to draw, then do so, starting with the innermost square of side 20. Each nested square should be centered at the origin (0,0). Note this problem is \*much\* easier to solve if you first define a function that draws a centered square of some side length, then restores the turtle's original state as it was before the call. Here's some pseudocode that uses this approach:

**init side to 20**

**do the following N times:**

**draw square with given side,**

**centered at origin and**

**preserving original turtle state**

**add 20 to side**