Instructions

These exercises are designed to help cement the concepts from lecture and give you an opportunity to apply them yourself. You are welcome to discuss and share ideas with each other, but should write the answers and code yourself.
Some questions ask you to complete a checkoff, which means you need to come to office hours at this link and talk to a staff member. Staff members may complete checkoffs for multiple students at one time, meaning you may not answer every question yourself but instead should listen to what your peers have to say and ask questions! Collaboration is an important part of computer science.

1 Tasks

1.1 Writing your first program - hello.py

Write a program that will print "Greetings, your_name!!!" in a box like this:

```
########################
#                      #
# Greetings, Kylie!!!  #
#                      #
########################
```

Hint: You will need to use more than one print command. You should put each command on a new line.
Note: Due to the length of your name, the number of #'s in your result may not be exactly the same as the one above. Don't worry as long as you get a similar layout.

1.2 Adding comments

At the beginning, your programs will be small and easy to understand. Over time, they'll get bigger and more complicated. Instead of having to understand what every line of code does, you can add comments to your code that explain what you were trying to do.
You can add a comment using the # sign:

```
# Print a message to the screen
print("hello, wtp!")
```

Comments are ignored by the computer – you can write anything you want there. Best practice, and what we would like you to do in this class, is to write code that explains what your program does and what you want it to do. For example, if you were writing a game like rock-paper-scissors, but you didn't know where to start, you could write this:

```
# First player picks one of rock, paper, or scissors
# Second player picks one of rock, paper, or scissors
# If first player beats second player, print a winning screen for her
# If second player beats first player, print a winning screen for her
```

For this part of the problem set, we want to add comments to the **hello.py** file that we just wrote. Now, edit the program from the previous exercise and add a comment at the beginning of the file with the filename, your name, anyone you worked with (none if it's no one), and a sentence about what it does. You should do that for all future programs.

> *Checkoff #1—Show off your hello.py file! Why are comments so important?*

1.3 String operators - stringop.py

String operators are less intuitive than those on numbers. Given the following variables, what are the values of these expressions? (Try getting the answer on your own and use IDLE (interactively) only to check your answer. Create a file called **stringop.py** and write your answers in it using comments.)

```
# Variables
look = "Look at me!"
now = " NOW"


# Expressions
look[:4]

look[-1]

look*2

look[:-1] + now + look[-1]

now[1]

now[4]

look*2 + look[:-1] + now + look[-1]
```

> Food for thought:
> Does this problem give you some hints about doing Problem 1.1 in an efficient way instead of typing a lot of #'s on the screen?

1.4 Doing math - calculator.py

We can also use Python as a calculator. The basic operations are:

+	Addition
−	Subtraction
*	Multiplication
/	Division
**	Exponentiation

Create a file called **calculator.py** and use Python to solve the following problem:
If you run a 5K race in 20 minutes 30 seconds, what is your average time (minutes) per mile? What is your average speed in miles per hour?

> **Hint**: A 5K is five kilometers. There are 1.61 kilometers in a mile. In your program, store useful values in variables, and print your final answers!

> *Checkoff #2—Why is giving variables helpful names important?*

Food for thought:
How would you expand this to calculate the average speed in mph for *any* distance in km and *any* time in minutes? We'll talk about one method to realize it later this week.

1.5 Greetings - greetings.py

Print a phrase and greeting using only the variables provided in **greetings.py**. Complete instructions are in the file.

2 Optional Challenge[1]:

Revisit Problem 1.1, and write a program that prints "Greetings, name!!!" in a box of stars for ANY name.

> **Hint**: Find the length of a string with len(string).

Note: In the coming lectures, we'll talk about "functions", which can help to solve this problem more efficiently.

Submitting your PSET

After you've finished your PSET and checkoff: Log into your Canvas account, find the post for Problem Set 2 in Assignments, and submit all of the files that you created or edited. After you turn in your assignment, you're all done!

[1] optional problems are provided for those who have further interests and want to explore more. You are not responsible for those questions, however.