*In this workshop, you'll gain practice with writing functions in Python, with accumulator patterns, for loops, and other concepts we have seen so far. Your workshop leader will guide you through the process. Please **do not** work ahead.*

1. Consider the following code:

```
# this program uses a function to print a message
def printMessage(name):
    print("Welcome to Python!")
    print("Behold the power of functions,", name + "!")

def main():
    name = input("Hello, human, what is your name? ")
    printMessage(name)

main()
```

   (a) What Python **keyword** is used to indicate that a line of code starts a function definition?

   (b) What are the two *function headers* in the code above? Circle them.

   (c) What are the *names* of the two functions above?

   (d) Given the input `"Antoine"`, what would the output of the program be?

   (e) What line of code would you add (and where) to cause the program to print the last two lines of output *twice*?

2. Examine the following program:

```
import math

def calculateArea(radius):
    area = math.pi * radius ** 2the frogs will *not* be printed on the same 4 lines (you
    print("The area of a circle with radius", radius, "is", round(area, 2))

def main():
    radius = int(input("Enter the circle radius: "))
    calculateArea(radius)
```

(a) If the above program is run, what will it do? Briefly explain.

(b) What line of code needs to be added (and where) to have the program ask for the circle radius?

(c) In the code above, the name of the parameter in the function definition for `calculateArea` is the same name as the argument to the function. Is this required? Briefly explain.

(d) If you wanted to change where the area of the circle is *printed* from within the `calculateArea` function to `main`, how would the program need to change?

3. Consider the following program:

```
def addNumbers(num1, num2):
    print(num1, "+", num2, "=", num1+num2)

def subtractNumbers(num1, num2):
    print(num1, "-", num2, "=", num1-num2)

def main():
    first = int(input("Enter a number between 2 and 10: "))
    second = int(input("Enter another number between 1 and 10: "))
    operator = input("Enter '+' to do addition or '-' to do subtraction: ")
    if operator == '+':
        addNumbers(first, second)
    if operator == '-':
        subtractNumbers(first, second)
    else:
        print("Invalid operator!")

main()
```

(a) Say you run the program above, typing the numbers 3 and 5, and the operator `'+'`. What will the output be?

(b) To fix the output in the program, what needs to change?

(c) Say that we want to modify the `addNumbers` and `subtractNumbers` functions so that they read as follows:

```
def addNumbers(num1, num2):
    print(num1, "+", num2, "=", end=" ")
    # new line 1

def subtractNumbers(num1, num2):
    print(num1, "-", num2, "=", end=" ")
    # new line 2
```

What lines would need to be added to these functions to *return* the result of applying a given operation (addition or subtraction)? What would need to change in `main` to print the result of calling the `addNumbers` and `subtractNumbers` functions?

(d) Assume that we run the program and it shows the following output:

```
Enter a number between 1 and 10: 56
Enter another number between 1 and 10: 4
Enter '+' to do addition or '-' to do subtraction: +
56 + 4 = 60
```

Notice that the program asks for a number between 1 and 10 but still accepts the given input. What lines need to be added to the program to cause it to print a warning such as "WARNING: at least one of the numbers you entered was out of range!"

4. Write a function `frog()` that draws a frog like the following:

```
  @..@
 (----)
( >--< )
^^ ~~ ^^
```

5. Revise (or, in detail, describe how you would revise) the `frog()` function so that it takes a parameter with a number of spaces to print at the beginning of each line. For example, if the function is provided the value 10, it should print the following frog:

```
          @..@
         (----)
        ( >--< )
        ^^ ~~ ^^
```

   (Notice that the frog is shifted to the right by 10 spaces exactly.)

6. Write another function (use the back side of this page) `moveFrog` that accepts a parameter `N` and calls `frog` `N` times. The argument to each call of the `frog` function should be a successive multiple of 5, starting at 1. For example, if the number 4 is entered, the program should print 4 frogs: the first frog should be 5 spaces from the left, the next frog should be 10 spaces from the left, the next should be 15 spaces from the left, and the last frog should be 20 spaces from the left. Note that the frogs will *not* be printed on the same 4 lines (your program would print a total of 16 lines; 4 lines for each of the 4 frogs).