Activity 2.2.3 *Python* Functions

Introduction

In this activity, you will create **functions**. Functions in *Python* are almost the same as procedures in App Inventor:

- A *Python* function is defined in a block of code.
- A *Python* function can be defined with **arguments**. The arguments are variables that only be used locally inside the definition of the function.
- A *Python* function can **return** a value back to the line of code that **called** the function.
- Code that calls a function **passes** a value for each of the function's arguments.

Materials

- Computer with *Python*

Procedure

1. You **call** a function and **pass values** for arguments. Use parentheses with the function's name, followed by parentheses. Put values for the arguments inside the parentheses. An example:

```
In []: range(3) # Call the range() function with one argument.
Out[]: [0, 1, 2]
```

   The `range()` function is one of the 80 functions built into *Python*. A full list is at **http://docs.python.org/2/library/functions.html**. Name another.

2. A function can be described in three ways:

   - **Code** uses Python, App Inventor, etc.
   - **Human language** uses human sentences, in English, Spanish, etc.
   - **Pseudocode** uses human language fragments mixed with code.

Pieces of code that are used in pseudocode **Conditions can be expressions in math or human language**
`variable = value`

`for item in list:`
  block of pseudocode
 	
`while` condition:
  block of pseudocode

`if` condition:
  block of pseudocode
`else:`
  block of pseudocode

A good description includes

- arguments the function accepts or requires
- the **algorithm** used to calculate the result
- the type of value returned

Example descriptions for `range()` are shown in the following table.

Type of description	Description
Human language	`sequence(stop)` accepts a numeric value for `stop`. It returns a list that counts by 1 from 0 up to `stop`, except that it stops short of actually including `stop`.

Pseudocode	`sequence(stop)` does this:
	
	`counter = 0`
	`while counter<stop:`
	put counter in a list to be returned
	`counter = counter + 1`
	return the list

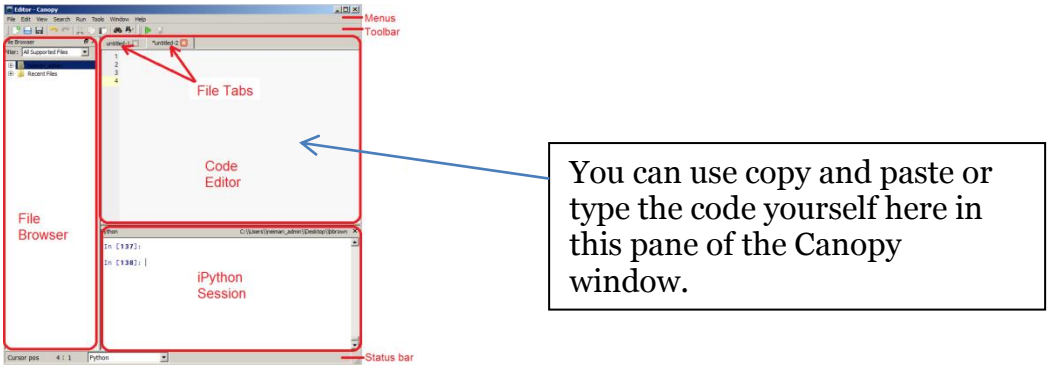
Python	`def sequence(stop):`
	`  number_list = []`
	`  counter = 0`
	`  while counter<stop:`
	`  number_list.append(counter)`
	`  counter = counter + 1`
	`  return number_list`

When a programmer creates a function, sometimes it is helpful to start out describing the function using human language or pseudocode. Code has the advantage of being precise and able to execute on a computer. Why would someone ever start by describing a function with human language or pseudocode?

3. Experiment with calling the `abs()` function.

```
In []: abs(-7) # absolute value function
Out[]: 7
```

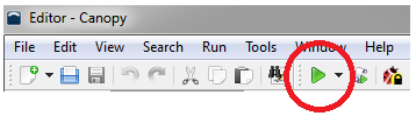
4. You can also define your own function with *Python*'s `def` statement. You can write the code description in the editor and execute it to create the function. Then you can call the function from the interactive session.



a. Enter the following code in the code editor. Note that Python uses a **colon** and **indentation** to create blocks of code. Each level of indentation is 4 spaces.

```
1  def rectangle_area(width, height):
2      ''' Calculates the area of a rectangle
3      Accepts numeric values for width, height
4      Returns int or float
5      '''
6      area = width * height
7      return area
```

b. Execute the code to define the function.



c. Call the function from the IPython command line. Pass values for arguments.

```
In []:  rectangle_area(3, 5)
Out[]:  15
```

5. Create another function `rectangle_perimeter(width, height)` that accepts numeric values for width and height and returns the perimeter of a rectangle with this width and height.

a. Create the definition in the code editor. Type or copy and paste from the starter code here. Keep the `rectangle_area()` function definition that you created earlier.

```
9  def rectangle_perimeter(width, height):
10     ''' Describe function in human language here.
11     Accepts...
12     Returns...
13     '''
14     perimeter = ...complete this
15     return ...put something here
```

b. Execute your function definition.

c. Test your function by calling it from the command line. Pass values for the arguments.

```
In []:  rectangle_perimeter(3, 5)
Out[]:  16
```

6. Many useful functions have already been written and shared. The *Python* standard library contains hundreds of these functions you can use. For example, the file `random.py` contains many functions including the `randint(min, max)` function. When you **import** a file (also called a **module**) to execute its function definitions, the function names are organized in the *Python* namespace, a **tree** of names. Call the function using the module name and a period. Try the following input in the IPython session. Continue experimenting with the function . Use the up arrow to visit your command history and repeatedly execute the last line calling `randint()`.

```
In []:  import random
In []:  random.randint(1, 3)
In []:  random.randint(1, 3)
In []:  random.randint(1, 3)
In []:  random.randint(1, 3)
```

The `randint(min, max)` function accepts _____.

The function returns _____.

> **Note**: To use the random library in the code editor, place the import statement in the code editor file. Import statements are placed at or near the top of the code editor file, so that multiple import statements will be grouped there together.

7. When creating a function or analyzing an existing one, think about the special cases. Your testing of `randint()` probably left some ambiguities. Do you know what happens if `min` or `max` is not an integer? What if `min` is bigger than `max`? Fill in the table with any additional special cases you can think of, especially **boundary cases**.

Ambiguity	Test case	Result
What if min or max is not an integer?	`randint(1, 3.6)`	
What if min > max?	`randint(3, 2)`	

8. ***Connections.*** You learned about the `randint()` function by using it, without being able to look at the code that defines it. This approach connects to many other topics of the course.

    a. ***Ethics.*** Figuring out what code does by testing it with various arguments is called **reverse engineering** code. Reverse engineering enables a programmer to mimic another engineer's software. **Proprietary** software is owned by a person or company and usually costs money. Do you think it should be legal to create a competing piece of software by reverse engineering? Explain why.

    b. ***Cybersecurity***. Hackers test functions with different input, trying to find boundary cases the programmer hadn't thought of. Errors and unexpected results can allow a hacker to use code in a way it wasn't intended. Can you think of an object you've "hacked," in the sense of using the object in a clever way that was not what the object was designed for?

    c. ***Software engineering.*** One way of writing software is called **test-driven development (TDD)**. In TDD, coders write down the desired results for various arguments first. These are tests the function must pass, including boundary conditions. After writing the tests down, the programmer writes a function to pass the tests. The advantage of this method is that clear expectations produce success. How is this similar to seeing the rubric for an assignment before you start working on the assignment?

Name _____ Hour ____

2.2.3A Python Functions Conclusion Questions

1. Instead of defining a function, you could just type the code from inside the function every time you needed it. What is the advantage of using a function instead of just using the function's code every time you need it?

2. Steps 4 and 5 each had lettered parts a, b, and c. These parts walked you through the process to create and use a function. What are the three parts of the process?

3. Describe how you test a function to see whether the code defining the function is correct.

4. The `range()` function is one of the 80 functions built into *Python*. A full list is at **http://docs.python.org/2/library/functions.html**. Name another.

5. Use step 6 for the following question.

   The `randint(min, max)` function accepts _____.

   The function returns _____.

6. Practice creating and testing functions by completing the following challenges. Write your function after you have tested it.

   a. Create a function `add_tax(amount, tax_rate)` that accepts numeric values `amount` and `tax_rate` and returns a new total amount with the tax amount included. Note that tax rate and tax amount are different.

   ```
   In []:  add_tax(200.00, 0.05)
   Out[]:  210.0
   ```

   b. Create a function `roll_pair_of_dice()` that returns the sum of two random numbers each from 1 to 6.

   ```
   In []:  roll_pair_of_dice()
   Out[]:  7
   ```

c. Define a function `mean(a, b, c)` that returns the mean of three numbers. *Hint:* Divide by `3.0` to get a `float`.

```
In []:  mean(2, 3, 5)
Out[]:  3.333333333333333
```

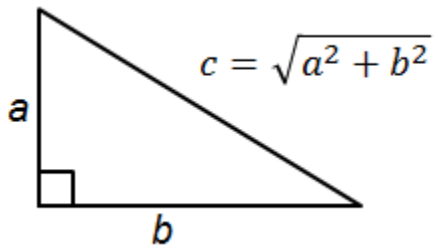
d. Create a function `hypotenuse(leg1, leg2)` that returns the length of the hypotenuse of a right triangle.

```
In []:  hypotenuse(3, 4)
Out[]:  5
In []:  hypotenuse(6, 8)
Out[]:  10
In []:  hypotenuse(5, 12)
Out[]:  13
```

Use your function to determine: What is the hyptoenuse of a right triangle with legs 5 and 10?

The following reminders from mathematics might help.

- A **right** triangle has a 90° angle.
- The **legs** are the sides adjacent to the right angle. The **hypotenuse** is the side opposite the right angle.
- The **Pythagorean Theorem** says that if a and b are the sides of a right triangle and c is the hypotenuse of a right triangle, then
$$a^2 + b^2 = c^2.$$
- If you know a and b (the lengths of the legs), you can calculate c (the length of the hypotenuse) by taking the square root of both sides.



- You can use `number**0.5` to take the square root of a number in *Python*. Something to the ½ power is the square root of that thing.