**Python Functions**

Python functions act very similarly to JavaScript functions. They must have parentheses **()** after the function name, can contain parameters and can return values if you want. We don’t use curly brackets **{}** to encapsulate the function’s code block (just indent instead) and we must add a colon **:** after the function name and parentheses.

**Similarities** **Differences**

- Parenthesis after function name - Use **def** instead of **function** when defining function

- Uses parameters - No {} use indent instead

- Parameters default values - Use : after () when defining

- return optional - Functions are not hoisted

- Calling function with incorrect number of parameters

& without default values causes error

***#*** *defining functions*

**def myFunction(): def add( num1, num2 ):**

**print( "No parameters" ) return num1 + num2**

***#*** *calling functions*

**myFunction() add( 1, 2 )**

Python functions are not hoisted therefore you must define them before calling/invoking them.

***#*** *calling function*

**sayFullName( "James", "Finn" ) *#*** *ERROR because function isn’t defined*

***#*** *defining functions*

**def sayFullName( fName="", sName="" ):**

**print( fName + " " + sName )**

**Lambda Functions**

In Python, a Lambda function is a quick, one-line anonymous function. It can only contain a single function expression and automatically returns a value without using the **return** keyword. It can have numerous parameters. You do not use the **def** keyword when defining them, but you need to use the **lambda** keyword.

**subtract = lambda n1, n2 : n1 - n2**

**subtract( 4, 3 ) *#*** *returns 1*

Here is the same Lambda function but with the different parts highlighted. Function name, parameters, function expression (code block).

**subtract = lambda n1, n2 : n1 - n2**

**subtract( 4, 3 ) *#*** *returns 1*