Monroe Township Library Coding Bootcamp

Class 5 Notes

* Functions
* Calling functions
* Arguments and parameters
* Return
* Default arguments
* Why should you use functions?

**Functions:**

* Functions are a way to bind a specific block of code functionality to a function name which can be used over and over
* New functions are created using the def keyword, assigning a name to the function, then closing with parentheses and a colon
  + Function names follow the same naming conventions as variables
  + As usual, the code that comes after the colon must be indented
  + The code contained within the function block is bound to the function’s name
* To actually use the logic within a function, it needs to be *called* which is done by simply typing the name of the function with the opening and closing parentheses
* We’ve been using functions for a while; print() and input() for example are both built-in functions that come with Python

**Arguments & Parameters:**

* Optionally, you can set up your function to accept **arguments** which are bits of information that can be passed to your function when it is called and used in the body of the function
* When defining your function, you can choose to create **parameters** which go inside the parentheses after the function name; if you have more than one parameter, each one should be separated by a comma
  + Parameters act as a sort of placeholder, and will be replaced by the data submitted as arguments when the function is called
  + Functions can accept arguments of any data type in Python; like variables, you do not need to specify which data type your arguments will be when you define your function
* When calling your function, arguments are simply passed by putting them within the parentheses, separated by commas
  + You will get a TypeError if you try to call a function with two few or too many arguments
  + If you have multiple arguments, they should be passed in positional order

**Return Statement:**

* Functions can, and typically do, contain a return statement which produces a specific result which can then be saved to a variable or used in some other way
  + To create a function that returns a value, just use the return keyword in front of whichever value you want the function to return
  + Once your program reaches a return keyword, it will exit out of that function
  + Functions can contain multiple return statements and can return different values if different conditions are met
* Functions that don’t reach a return statement will return None by default

**Default Arguments:**

* Optionally, you can define your function to have default values by assigning the value with the assignment operator (=) when defining the parameters in your function
  + Parameters with default values have to come after any parameters that do not contain default values
* When the function is called, if no argument is passed to the function, it will use the default value for the argument
  + If an argument *is* passed in when the function is called, it will override the default value and use the passed-in argument instead
  + All other parameters in your function that do not have default values are still required to be passed in
    - For example, if you create a function with one regular parameter and one parameter with a default value, *at least 1* argument must be passed in when the function is called, otherwise you will get a TypeError

**Why should you use functions?:**

* Functions are **reusable** which means you only have to type out a block of code once, and then can call it as many times as needed in your program, passing in different arguments each time
* Functions give your code **structure**
  + Typically, functions should do just one specific task; having your code broken down into several properly named/documented functions make it much cleaner and more readable
* Functions make **debugging** much easier
  + If there are bugs in your code, having your logic broken up into functions make it much easier to isolate the problem because they can all be tested individually

**Project: Caesar Cipher**

**Check class files at** [**github.com/monroecoding**](https://github.com/monroecoding)

Chris DiFazio

Adult Services Librarian

cdifazio@monroetpl.org