Monroe Township Library Coding Bootcamp

4/2 Class Notes

* Functions
* Calling functions
* Function arguments
* Return
* Default arguments
* Multiple arguments (\*args)
* Keyword arguments (\*\*kwargs)

**Functions:**

* Functions are a way to bind a specific block of code functionality to a keyword 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
  + 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

**Arguments:**

* 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
* When defining your function, place arguments within the function’s parentheses; if you have more than one argument, they should be separated by a comma
  + 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 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 the return keyword, it will exit out of that function
* Functions that don’t reach a return statement will return None

**Default Arguments:**

* Optionally, you can give your arguments a default value by assigning the value with the assignment operator (=) in the function definition
  + Default arguments have to come after any arguments 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

**Multiple Arguments:**

* If you want to create a function that takes an unspecified number of arguments, you can put an asterisk (\*) in front of the argument name
  + \*args is a generic name that is often used, but use something more descriptive if it makes sense
* There is no limit on the number of arguments you can pass in this way; the values will be passed into the function as a tuple which can be accessed from the argument name given in the function definition

**Keyword Arguments:**

* Using a double asterisk (\*\*) in front of an argument, allows your function to accept an unspecified number of keyword arguments
  + \*\*kwargs is a commonly used name
* When calling the function, keyword arguments must be passed in using value assignment (keyword=value)
  + The arguments are stored in a dictionary of key:value pairs, that can be accessed in the function with the name of the name of the \*\*argument

**Project: Caesar Cipher**

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

Chris DiFazio

Adult Services Librarian

cdifazio@monroetpl.org