**Defining a Function:**

keyword def is used to inform Python that you’re defining a function. docstring is a comment, which describes what the function does.A function call tells Python to execute the code in the function. To call a function, you write the name of the function, followed by any necessary information in parentheses.

**Passing Arguments:**

Python needs to match each argument in a function call with a parameter in the function definition when you call a function. The easiest method to accomplish this is to use the arguments in the given order. Positional arguments are values matched in this manner.A keyword argument is a name-value pair that you pass to a function.You can specify a default value for each argument when writing a function.

Python utilizes the argument value if one is supplied for a parameter in the function call. If not, the default value for the option is used.

**Return Values:**

A function doesn’t always have to display its output directly. Instead, it can process some data and then return a value or set of values. The value the function returns is called a return value.The return statement takes a value from inside a function and sends it back to the line that called the function.Optional values allow functions to handle a wide range of use cases while letting function calls remain as simple as possible.A function can return any kind of value you need it to, including more complicated data structures like lists and dictionaries.

**Passing a List:**

A function that receives a list as input has the ability to alter the input. Because any modifications made to the list inside the function's body are permanent, you can continue to work effectively even while working with massive volumes of data.You should send the original list to functions unless you have a specific reason to pass a copy, even if you can preserve the contents of a list by passing a duplicate of it to your functions. A function can operate more efficiently if it uses an existing list rather than creating a new one, which saves time and memory. This is particularly valid when handling big lists.

**Passing an Arbitrary Number of Arguments:**

The asterisk in the parameter name tells Python to make a tuple, containing all the values this function receives.Python packs the arguments into a tuple, even if the function receives only one value. If you want a function to accept several different kinds of arguments, the parameter that accepts an arbitrary number of arguments must be placed last in the function definition.The double asterisks before the parameter cause Python to create a dictionary containing all the extra name-value pairs the function receives. Within the function, you can access the key-value pairs in user\_info just as you would for any dictionary.

**Storing Your Functions in Modules:**  
Using descriptive names for your functions makes your programs much easier to understand. To take things a step further, you can import your functions from a separate file known as a module into your main application. Python is instructed to make a module's code available in the program file that is now running by using an import statement.A module is a file ending in .py that contains the code you want to import into your program.You can also import a specific function from a module. Here’s the general syntax for this approach: from *module\_name* import *function\_name*.You can import as many functions as you want from a module by separating each function’s name with a comma: from *module\_name* import *function\_0, function\_1, function\_2*.You can use a short, unique alias—another name for the function that sounds a bit like a nickname—if the name of the function you're importing conflicts with another name that already exists in your program or if the function name is lengthy. When you import the function, you'll give it its unique moniker.The general syntax for providing an alias is:from *module\_name* import f*unction\_name* as *fn*.You can also provide an alias for a module name. The general syntax for this approach is: import *module\_name* as *mn*. You can tell Python to import every function in a module by using the asterisk (\*) operator. The general syntax for this approach is: from *module\_name* import \*.

**Styling Functions:**

If you specify a default value for a parameter, no spaces should be used on either side of the equal sign: def *function\_name(parameter\_0, parameter\_1='default value')* The same convention should be used for keyword arguments in function calls: *function\_name(value\_0, parameter\_1='value').* PEP 8 recommends that you limit lines of code to 79 characters so every line is visible in a reasonably sized editor window. Most editors automatically line up any additional lines of arguments to match the indentation you have established on the first line:

def *function\_name(*

*parameter\_0, parameter\_1, parameter\_2,*

*parameter\_3, parameter\_4, parameter\_5):*

*function body...*

All import statements should be written at the beginning of a file.