1. A function is a block of code that has a certain poropuse, it is supposed to do only one thing and be responsible for a very specific task. It runs only when it is called. You can call a function in main like that: func(). And the syntax to write a function is:

def func():

1. At the end of each function there is a return value, the return value is the value that the functions returns after doing the task. For example if there is a function that calculates the sum of two numbers, then the return value is the sum of the function. When you call the function in main, and you assign the function to a variable, then the variable is going to get the return value of the function. An example for a function with return value is:

def nums\_sum(a, b):

return a + b

1. There are two types of of scenarios with functions. One is pass by value and the other is pass by reference.

Pass by value refers to when you pass a value to a function as an input and any changes that will occur on the variable in the function will not affect the variable in the main. This bahavior happens to numbers, string and tuples.

The other scenario is pass by reference, which is the opposite. So if a value is passed in a function and changes are happening on it, then the variable will change in main as well. This happens with dictionaries, lists.

1. There are a few scopes in a program. A scope is an area which a variable is known at.

The first and biggest scope is global scope which is when variables can be accessed from anywhere in the program, even within functions defined in the same file.

They are created when they are defined throughout program execution.

Local scope: variables created within a function are accessible only within that function.

They are created when the function is called and are destroyed when the function completes its execution.

Enclosing Scope: python allows functions to access variables from the enclosing scope, such as a function within another function.

For example, an inner function can access variables from its outer function.

Lifetime of variables/objects: global variables are created at the beginning of the program and oocur until the program ends.

Local variables are created when the function is called and are deleted when the function is finished.

Variables in enclosing areas live until the outermost function has finished its execution.

9. Recursion is a process in which a function calls itself as part of the process. In most cases, the function is called with different parameters than the previous call. The idea behind recursion is often used to solve problems by breaking them down into smaller problems until you arrive at a base case that can be solved directly, and then it goes forward again with the solved small problems until it solves the function.

We will want to use recursion when the problem can be turn into small problems, when we want to use an algorithm in an efficient way (it does not work will all algorithms), and more. We don’t always have you use recursion, we can do everything without it but many times it saves a lot of memory and time and is a lot more efficient.

It depends on the algorithm and the problem we want to solve, but most of the time loops are more efficient and readable. But recursion can be simpler, and are better for most of mathematic problems.