GENERAL ASSEMBLY

Functions in Python

Patrick D. Smith

Lead Instructor, General Assembly DSI

LEARNING OBJECTIVES

- Learn how to write python functions from scratch
- Learn how to utilize python functions

Opening

Functions in Python

In a base sense, functions are reusable chunks of code that allow us to easily reproduce processes

This helps us then *call* the function and utilize it wherever we want

Functions are **the most important** building block of programming

Intro to Functions

Let's take a look at some examples!

Elements of Functions

Function Parameters

- A function can take parameters, which are values you supply to the function so that the function can do something utilising those values.
- These parameters are just like variables except that the values of these variables are defined when we call the function and are already assigned values when the function runs.

Function Parameters

- Parameters are specified within the pair of parentheses in the function definition, separated by commas. When we call the function, we supply the values in the same way.
- *Note the terminology used* - the names given in the function definition are called parameters whereas the values you supply in the function call are called arguments.

Let's take a look at some examples!

Local Variables

- When you declare variables inside a function definition, they are not related in any way to other variables with the same names used outside the function, IE: variable names are local to the function.
- This is called the scope of the variable. All variables have the scope of the block they are declared in starting from the point of definition of the name.

Let's take a look at some examples!

Global Variables

- If you want to assign a value to a name defined at the top level of the program (i.e. not inside any kind of scope such as functions or classes), then you have to tell Python that the name is not *local,* but it is ***global.***
- We do this using the global statement. It is impossible to assign a value to a variable defined outside a function without the global statement.

Global Variables

- You can use the values of such variables defined outside the function (assuming there is no variable with the same name within the function).
- However, this is not encouraged and should be avoided since it becomes unclear to the reader of the program as to where that variable's definition is.
- Using the global statement makes it amply clear that the variable is defined in an outermost block.

Let's take a look at some examples!

Default Arguments

- For some functions, you may want to make some parameters optional and use default values in case the user does not want to provide values for them. This is done with the help of ***default argument values.***
- You can specify default argument values for parameters by appending to the parameter name in the function definition the assignment operator (=) followed by the default value.
- The default argument value should be a constant.

Let's take a look at some examples!

Keyword Arguments

- If you have some functions with many parameters and you want to specify only some of them, then you can give values for such parameters by naming them.
- We call these ***keyword arguments***
- We use the name (keyword) instead of the position (which we have been using all along) to specify the arguments to the function.

Let's take a look at some examples!

VarArgs

- We can use VarArgs (Variable Arguments) when we want our function to take on any number of parameters

Let's take a look at some examples!

The RETURN Statement

- The return statement is used to return from a function i.e. break out of the function. We can optionally return a value from the function as well.

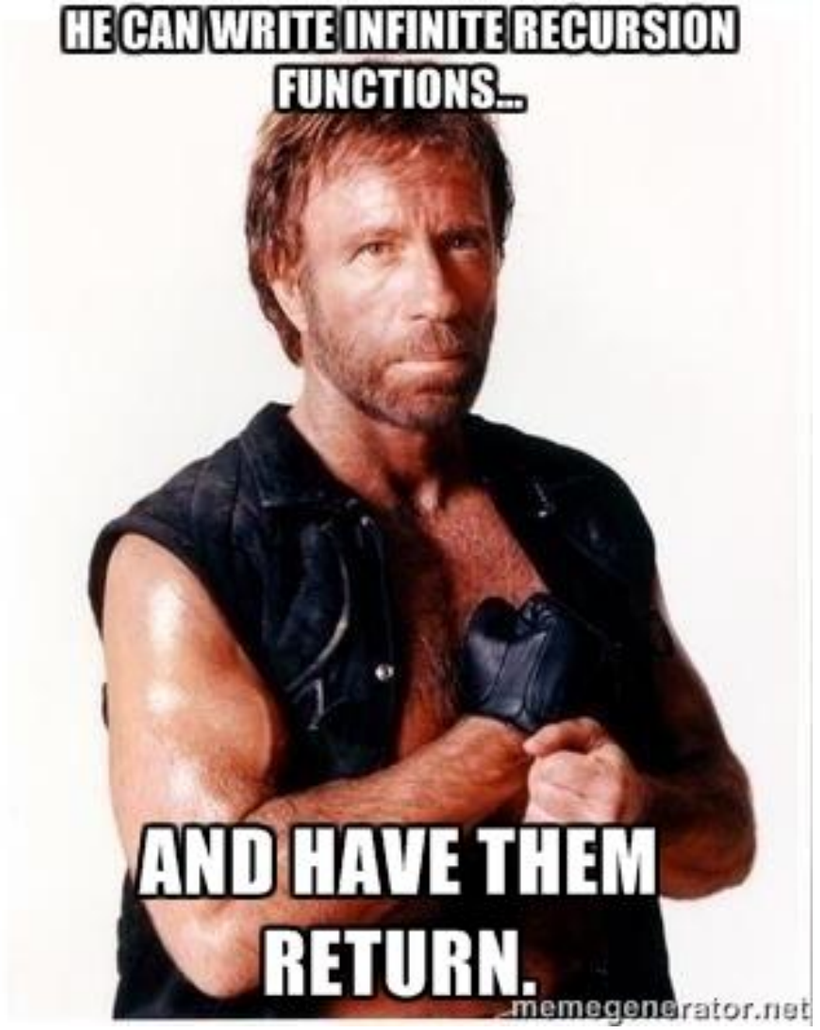
Let's take a look at some examples!

Functions - Special Cases

Functions in Python

Recursive Functions

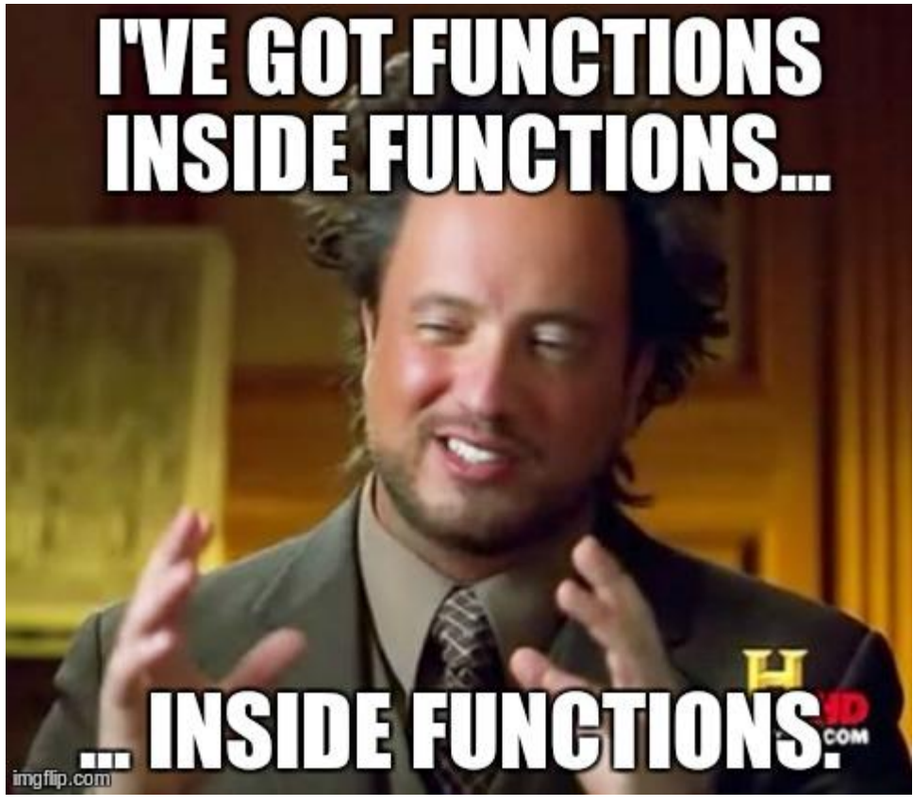
- Programming languages generally support recursion, which means that, in order to solve a problem, functions can call themselves to solve smaller subproblems.



Functions in Python

Nested Functions

- The ***for*** statement is used to iterate over the elements of a sequence.
- It's used when you have a piece of code which you want to repeat n number of times.
- You can use any object (such as strings, arrays, lists, tuples, dict and so on) in a for loop in Python.



Independent Practice

Conclusion