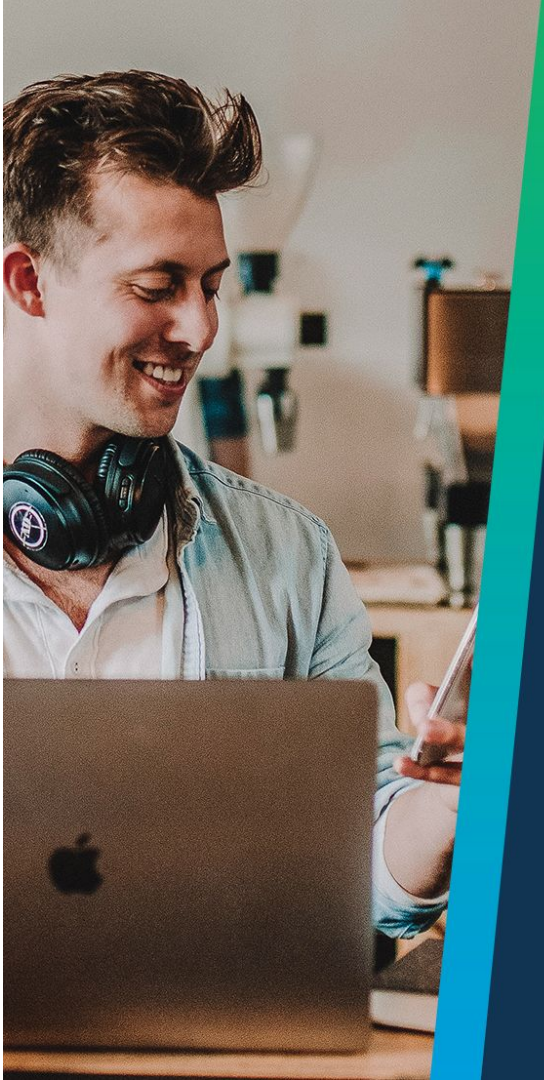


Cyber Security Bootcamp

Hyperiondev

Beginner Programming with Functions

Welcome

Your Lecturer for this session

Liano Naidoo

Objectives

1. Understanding functions
2. Calling functions
3. Creating self-defined functions
4. Understanding function scope

What is a Function?

- Reusable and Organised block of code.
- Sometimes called a ‘method’.
- Similar to functions in maths - f(x) takes input x and produces some output.
- Useful for abstraction.
    - For example, “make a cup of tea” vs “boil water, add tea bag, add sugar, add milk, stir”.

Calling Functions

- Functions with one **required positional** input:
  - my_function(input1)
- Functions with two **required positional** inputs:
  - my_function(input1, input2)
- Functions with one **required positional** input and one **optional keyword** input:
  - my_function(input1, keyword_arg=input2)

Functions in Python

- Python comes bundled with in-built functions.
- Examples:
  - **print(string)** - prints string to console.
  - **input(string)** - prints string to console, then reads input from the console as string.
  - **len(array)** - returns the length of an array.
  - **int(data)** - converts the value to an integer.

Is that all of the Functions in Python?

- The list of functions that you can use in Python doesn't just stop with what is built in.
- Using Pip (python package manager), you can install various packages containing **modules**.
  - Note: Some packages are already installed by default in Python, such as the maths package.
- These modules can be imported into your script using an import statement.

Importing Modules

- Let's take a look at the maths module. Let's say that you want to use round(), which rounds a number off. There are two ways to access this:
- import math

  ```
  my_result = math.round(my_num, 2)
  ```

- from math import round

  ```
  my_result = round(my_num, 2)
  ```

Creating our own Functions

- Uses the def keyword (for define):
  - ```
    def add_one(x): # function called add_one
        y = x + 1
        return y
    ```
- Important keywords:
  - def - tells Python you are defining a function
  - return - if your function returns a value, then use this keyword to return it.
    ```

Some Important Terms

- **Function** - A block of code that performs an action(in the form of code).
- **Method** - Same thing as a function(Called Differently).
- **Parameters** - The defined input of a function.
- **Arguments** - The values passed to parameters.

Why Functions?

- **Reusable code** - Sometimes you need to do the same task over and over again.
- **Error checking/validation** - Makes this easier, as you can define all rules in one place.
- **Divide code up into manageable chunks** - Makes code easier to understand.
- **More rapid application development** - The same functionality doesn't need to be defined again.
- **Easier maintenance** - Code only needs to be changed in one place.

Scope

- Where is a variable accessible in Python?
- Generally, whenever code is executed, variables become accessible across the entire script.
    - For example, you can define a variable within an if-statement and access it outside the if-statement after it is executed.
- Functions are different, however. Variables declared within functions are not accessible outside the function.
    - This avoids variable names being overwritten.

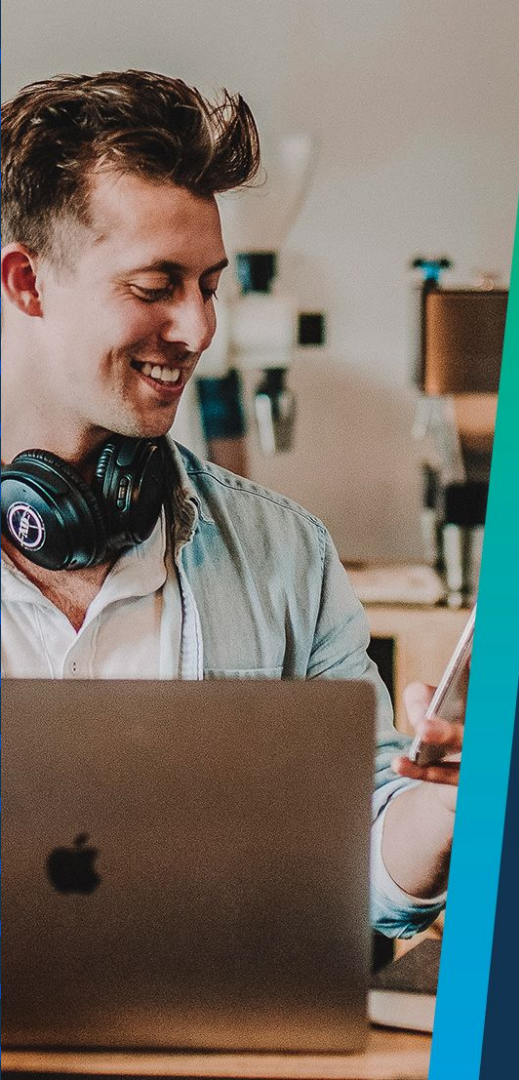
Default Values

- Remember **optional keyword** arguments? These are made with default values.
- def multiply(num1, num2 = 5):
- This can be called with multiply(10), for example.
- The default value can be overwritten with multiply(10, num2=6).

Hyperiondev

Q & A Section

Please use this time to ask any questions relating to the topic explained, should you have any



Hyperiondev

Thank you for joining us