Introduction to Functions: Takeaways

Syntax

FUNCTIONS

- Creating a function with a single return path:

```
def function_name(input_variable):
    result = input_variable * 5
    return(result)
```

- Creating a function with multiple return paths:

```
def function_name(input_variable):
    if input_variable > 5:
        return("Higher than 5")
    else:
        return("Lower than 5")
```

- Creating a function with multiple arguments:

```
def function_name(input_one, input_two):
    result = input_one + input_two
    return(result)
```

- Creating a function with an optional argument:

```
def function_name(input_one, input_two=10):
    result = input_one + input_two
    return(result)
```

- Calling a function inside another function:

```
def sum(input_one, input_two):
    return(input_one + input_two)
def average(input_three, input_four):
    total = sum(input)
    return(total / number_items)
def calculation(input_three, input_four):
    result2 = function_one(input_three, input_four)
    return(result)
```

Concepts

- Functions allow us to make code more reusable. Other than reusability, there are 3 main advantages of using functions:
    - They allow us to use other people's code without the necessity to have a deep understanding of how it was written. This is called **information hiding**.
    - They break down complex logic into smaller components or modules. Instead of writing very lengthy and complicated code, we can progress function by function. This is called **modularization**.
    - They streamline our code and make it easier to maintain. Programmers reuse the same functions in multiple situations across a project. This means that they generalize the function as much as possible to maximize its usefulness. This is called **abstraction**.
- Functions can have more than one return paths. While having more return paths can make a function harder to understand, it lets us hide complexity more effectively to people who want to use our function.
- Functions can also have one or more optional arguments, which allow us to write a more general and flexible function. Optional arguments let us write functions that adapt to the amount of information we want to pass in.

Resources

- Python Documentation: Defining Functions