Day 13: Python Functions

A function is a block of code which only runs when it is called.

```
# Creating a Function
def my_function():
    print("Hello from a function")

# Calling a Function
my_function()
```

Arguments

Information can be passed into functions as arguments.

```
# Single Parameter
def greetings(fname):
    print(f"Hello {fname}!")

greetings("Harry")

# Multiple Parameter
def greetings(fname, lname):
    print(f"Hello {fname} {lname}!")

greetings("Harry", "Jain")
```

Parameters or Arguments

- A parameter is the variable listed inside the parentheses in the function definition.
- An argument is the value that is sent to the function when it is called.

Number of Arguments

By default, a function must be called with the correct number of arguments. Meaning that if your function expects 2 arguments, you have to call the function with 2 arguments, not more, and not less.

```
# Example 1
def greetings(fname):
    print(f"Hello {fname}!")

greetings("Harry", "Jain")

# Example 2
def greetings(fname, lname):
    print(f"Hello {fname}!")

greetings("Harry")

# Example 3 (Default Value)
def greetings(fname, lname = "Styles"):
    print(f"Hello {fname} {lname}!")

greetings("Harry")
```

Keyword Arguments

You can also send arguments with the *key = value* syntax.

This way the order of the arguments does not matter.

```
def kids(child3, child2, child1):
  print("The youngest child is " + child3)

kids(child1 = "Emily", child2 = "Tobias", child3 = "Linus")
```

Arbitrary Arguments, *args

If you do not know how many arguments that will be passed into your function, add a `*` before the parameter name in the function definition.

```
def my_function(*kids):
  print("The youngest child is " + kids[2])

my_function("Emily", "Tobias", "Linus")
```

Passing a List as an argument

```
def my_function(food):
  for x in food:
    print(x)

fruits = ["apple", "banana", "cherry"]

my_function(fruits)
```

Return Values

```
# Multiply by 5
def my_function(x):
  return 5 * x

print(my_function(3))
print(my_function(5))
print(my_function(9))
```

The pass Statement

`function` definitions cannot be empty, but if you for some reason have a `function` definition with no content, put in the `pass` statement to avoid getting an error

```
def myfunction():
  pass
```