Creating a Function

In Python a function is defined using the def keyword:

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

Calling a Function

To call a function, use the function name followed by parenthesis:

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


my_function()
```

Arguments

Information can be passed into functions as arguments.

Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.

```
def my_function(fname):
  print(fname + "Institute")


my_function("DISHA")
my_function("Ravet")
my_function("Pimpri")
```

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.

```
def my_function(fname, lname):
  print(fname + " " + lname)


my_function("DISHA", "INSTITUTE")
```

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.

This way the function will receive a tuple of arguments, and can access the items accordingly:

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


my_function("AMAN", "TUSHAR", "KIRAN")
```

Keyword Arguments

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

This way the order of the arguments does not matter.

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


my_function(child1 = "DISHA", child2 = "TUSHAR", child3 = "KIRAN")
```

Default Parameter Value

If we call the function without argument, it uses the default value:

```
def my_function(country = "Norway"):
  print("I am from " + country)


my_function("Sweden")
my_function("India")
my_function()
my_function("Brazil")
```

Passing a List as an Argument

You can send any data types of argument to a function (string, number, list, dictionary etc.), and it will be treated as the same data type inside the function.

E.g. if you send a List as an argument, it will still be a List when it reaches the function:

```
def my_function(food):
  for x in food:
    print(x)
fruits = ["apple", "banana", "cherry"]
my_function(fruits)
```

Return Values

To let a function return a value, use the return statement:

```
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
```