## Function

* A function is a block of code which only runs when it is called.
* You can pass data, known as parameters, into a function.
* A function can return data as a result.

## 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.
* By default, a function must be called with the correct number of arguments.
* From a function's perspective:
* 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.

## Create and Call function

The following example has a function with one argument (fname). When the function is called, we pass along a first name, which is used inside the function to print the full name:

Creating a function and calling the function

* Use “def” keyword to create a function with one parameter

def myFunc(fname):  
 print("Hello " + fname)

* call the function by passing the argument for the defined parameter

myFunc("Emily") *# Hello Emily*myFunc("Sheran") *# Hello Sheran*

## 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 myFunc(\*fname):  
 print("Hello " + fname[1])  
  
  
myFunc("Emily", "Sheran", "Medhavi") *# Hello Sheran*

## Keyword Arguments

Send arguments with key=value syntax. Therefore, order of the arguments does not matter.

def myFunc(fname1,fname2,fname3):  
 print("Hello " + fname2)  
  
  
myFunc(fname1="Emily", fname3="Sheran", fname2="Medhavi") *# Hello Medhavi*

## Arbitrary Keyword Arguments, \*\*kwargs

If you do not know how many keyword arguments that will be passed into your function, add two asterisk: \*\* before the parameter name in the function definition.

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

def func(\*\*fname):  
 print (fname["fname1"],fname["fname2"])  
  
func(fname1="Emily",fname2="Sheran",fname3="Medhavi") *#Emily Sheran*

## Default Parameter Value

Default parameter value can be assigned within the function definition with parameter. It can be called with empty function.

def func(fname="Medhavi"):  
 print ("My name is : " + fname)  
  
func() *# My name is : Medhavi*func("Sheran") *# My name is : Sheran*

## Passing an iterable as an argument

def func(iterable):  
 for x in iterable:  
 print(x, end=" ")  
 print()  
  
  
name = ["Medhavi", "Janitha", "Nayomi"]  
number = (100, 200, 300)  
dictionary = {1: "One", 2: "Two", 3: "Three"}  
  
func(name) *# Medhavi Janitha Nayomi*func(number) *# 100 200 300*func(dictionary.values())*# One Two Three*

## Return Values

To let a function return value, use the “return” statement

def myfunc(number, by):  
 return number \* by  
  
print("number \* by is ", myfunc(3,2)) *# number \* by is 6*print("number \* by is ", myfunc(2,2)) *# number \* by is 4*

## The pass statements

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 myfunc(number, by):  
 pass