**Function**

* A function is a block of organized, reusable code that is used to perform a single, related action.
* Functions provide better modularity for your application and a high degree of code reusing.
* Python gives many built-in functions like print(), etc. but you can also create your own functions. These functions are called *user-defined functions.*

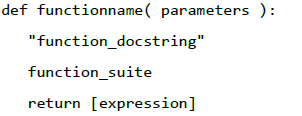
**Defining a Function**

* You can define functions to provide the required functionality.

Here are simple **rules** to define a function in Python.

* Function blocks begin with the keyword **def** followed by the function name and parentheses ( ( ) ).
* Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses.
* The first statement of a function can be an optional statement - the documentation string of the function or docstring.
* The code block within every function starts with a colon (:) and is indented.
* The statement return [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None.

**Syntax:**



**Example**

The following function takes a string as input parameter and prints it on the standard screen.

def printme( str ):

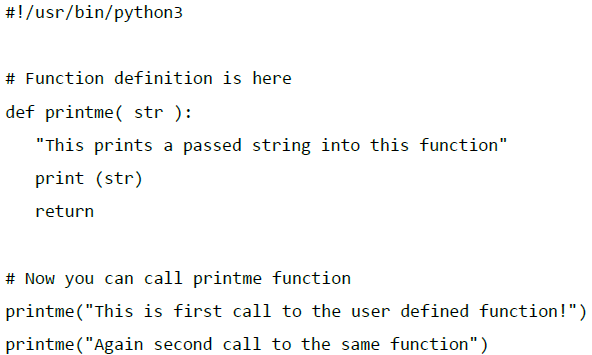
"This prints a passed string into this function"

print (str)

return

**Calling a Function**

* Defining a function gives it a name, specifies the parameters that are to be included in the function and structures the blocks of code.
* Once the basic structure of a function is finalized, you can execute it by calling it from another function or directly from the Python prompt.



**Output:**



**Pass by Reference vs Value**

* All parameters (arguments) in the Python language are passed by reference
* It means if you change what a parameter refers to within a function, the change also reflects back in the calling function.

#!/usr/bin/python3

# Function definition is here

def changeme( mylist ):

"This changes a passed list into this function"

print ("Values inside the function before change: ", mylist)

mylist[2]=50

print ("Values inside the function after change: ", mylist)

return

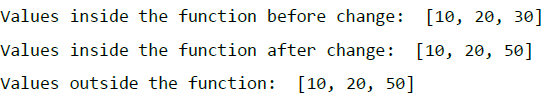
# Now you can call changeme function

= [10,20,30]

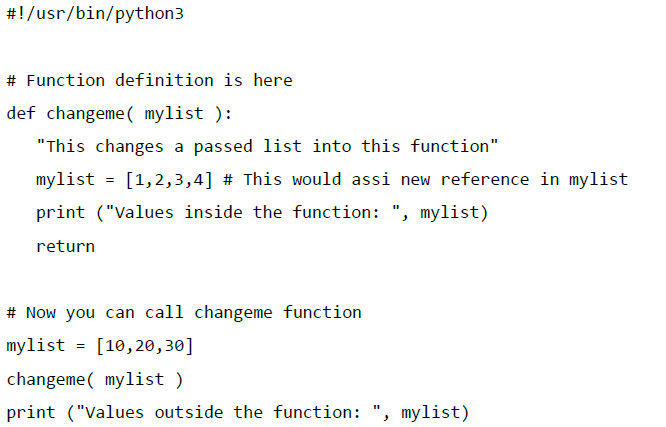
changeme( mylist )

print ("Values outside the function: ", mylist)

**Output:**



There is one more example where argument is being passed by reference and the reference is being overwritten inside the called function.

****

**Output:**

****

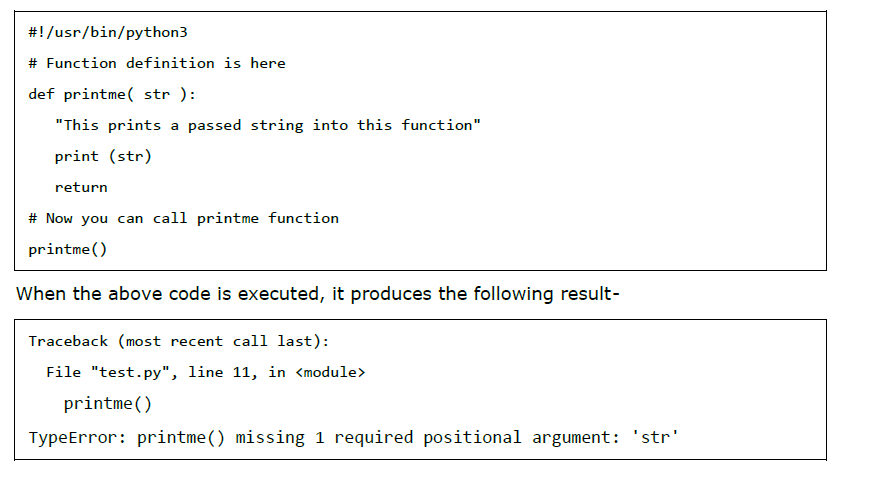
**Function Arguments**

You can call a function by using the following types of formal arguments-

* Required arguments
* Keyword arguments
* Default arguments
* Variable-length arguments

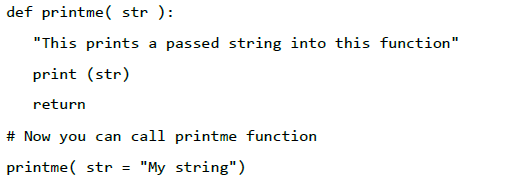
**Required Arguments**

* Required arguments are the arguments passed to a function in correct positional order.
* Here, the number of arguments in the function call should match exactly with the function definition. To call the function printme(), you definitely need to pass one argument, otherwise it gives a syntax error as follows-



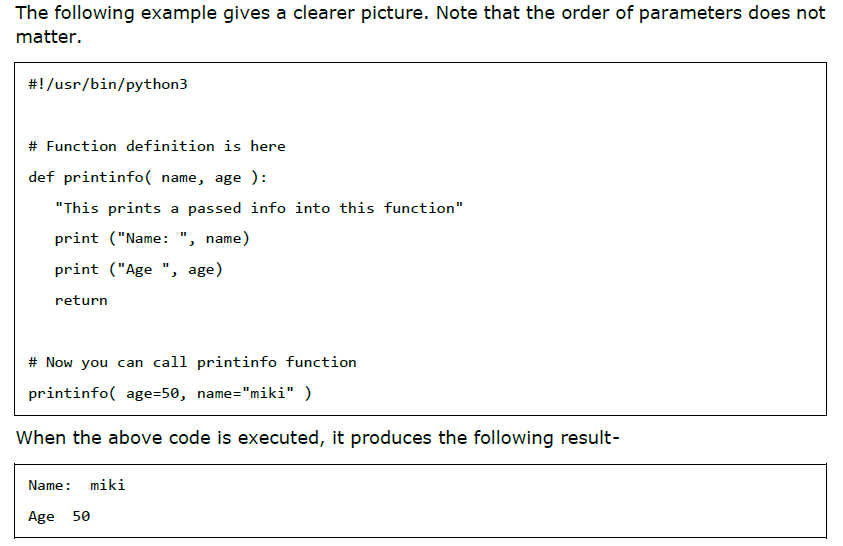
**Keyword Arguments**

* Keyword arguments are related to the function calls.
* When you use keyword arguments in a function call, the caller identifies the arguments by the parameter name.
* This allows you to skip arguments or place them out of order because the Python interpreter is able to use the keywords provided to match the values with parameters.
* You can also make keyword calls to the printme() function in the following ways-



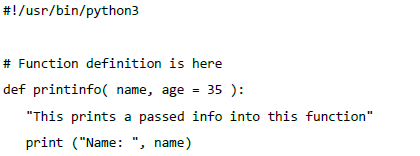
When the above code is executed, it produces the following result-

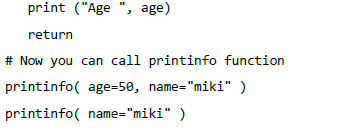
**My string**

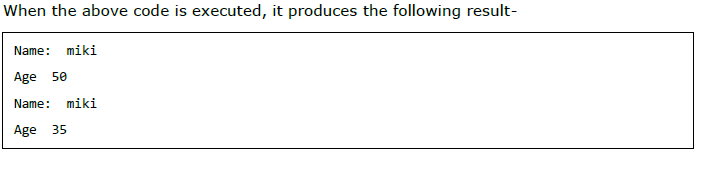
****

**Default Arguments**

* A default argument is an argument that assumes a default value if a value is not provided in the function call for that argument.
* The following example gives an idea on default arguments, it prints default age if it is not passed.

****

****

****

**Variable-length Arguments**

* You may need to process a function for more arguments than you specified while defining the function.
* These arguments are called variable-length arguments and are not named in the function definition, unlike required and default arguments.
* Syntax for a function with non-keyword variable arguments is given below

