Python Functions

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.

# To take input from the user

#num = int(input("Enter a number: "))

factorial = 1

# check if the number is negative, positive or zero

if num < 0:

print("Sorry, factorial does not exist for negative numbers")

elif num == 0:

print("The factorial of 0 is 1")

print("The factorial of 0 is 1")

else:

for i in range(1,num + 1):

factorial = factorial\*i

print("The factorial of",num,"is",factorial)

## Creating a Function

In Python a function is defined using the def keyword:

### Example

def my\_function():  
  print("Hello from a function")

## Calling a Function

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

### Example

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.

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:

### Example

def my\_function(**fname**):  
  print(fname + " Refsnes")  
  
my\_function(**"Emil"**)  
my\_function(**"Tobias"**)  
my\_function(**"Linus"**)

Write a program in Python for printing factorial of a number without function

# To take input from the user

num = int(input("Enter a number: "))

factorial = 1

# check if the number is negative, positive or zero

if num < 0:

print("Sorry, factorial does not exist for negative numbers")

elif num == 0:

print("The factorial of 0 is 1")

print("The factorial of 0 is 1")

else:

for i in range(1,num + 1):

factorial = factorial\*i

print("The factorial of",num,"is",factorial)

Write a program in Python for printing factorial of a number using simple function with passing single parameter

def fact(num):

# check if the number is negative, positive or zero

fact1=1

if num < 0:

print("Sorry, factorial does not exist for negative numbers")

elif num == 0:

print("The factorial of 0 is 1")

else:

for i in range(1,num+1,1):

fact1 = fact1\*i

return fact1

# To take input from the user

num = int(input("Enter a number: "))

f=fact(num)

Print(“The factorial is :”,f)

## 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:

### Example

If the number of arguments is unknown, add a \* before the parameter name:

def my\_function(\*kids):  
  print("The youngest child is " + kids[2])  
  
my\_function("Emil", "Tobias", "Linus")

## Keyword Arguments

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

This way the order of the arguments does not matter.

### Example

def my\_function(child3, child2, child1):  
  print("The youngest child is " + child3)  
  
my\_function(child1 = "Emil", child2 = "Tobias", child3 = "Linus")

## Default Parameter Value

The following example shows how to use a default parameter value.

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

### Example

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:

### Example

## def my\_function(food):   for x in food:     print(x) fruits = ["apple", "banana", "cherry"] my\_function(fruits) 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.

### Example

def myfunction():  
  pass

## Passing Tuple to function

Python has a built-in data type called a tuple. Data inside a tuple can be of any type say, integer, string or a float value, or even a tuple type. The tuple uses comma-separated values within round brackets or parentheses to store data. Tuples can be defined using any variable name and then assigning different values to the tuple inside the round brackets. The tuple is ordered, unchangeable, and allows duplicate values.

def tupleArg(inputTuple):

print("Tuple argument passed as input to the function is: ", inputTuple)

tupleArg((1, 2, 3))

## Passing Dictionaries to Functions

We can pass a dictionary to a function by passing the name of the dictionary. Let's define a function that accepts a dictionary as a parameter.

def fun(dictionary):

for i, j in dictionary.items():

print(i, '--', j)

**A Python function to accept a dictionary and display its elements.**

def fun(dictionary):

for i, j in dictionary.items():

print(i, '--', j)

d = {'a':'Apple', 'b': 'Book', 'c': 'Cook'}

fun(d)

Recursion in Python

When a function call itself with some other parameter till some condition is called as recurstion.

Write a program in Python for printing factorial of a number using recursion

def factorial(n):

if n == 0:

return 1

else:

return n \* factorial(n-1)

n=int(input("Input a number to compute the factiorial : "))

print(factorial(n))

# Python Arrays

Arrays are used to store multiple values in one single variable:

### Example[Get your own Python Server](https://www.w3schools.com/spaces/)

Create an array containing car names:

cars = ["Ford", "Volvo", "BMW"]

## What is an Array?

An array is a special variable, which can hold more than one value at a time.

If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

car1 = "Ford"  
car2 = "Volvo"  
car3 = "BMW"

## Access the Elements of an Array

You refer to an array element by referring to the index number.

### Example

Get the value of the first array item:

x = cars[0]

### Example

Modify the value of the first array item:

cars[0] = "Toyota"

## The Length of an Array

Use the len() method to return the length of an array (the number of elements in an array).

### Example

Return the number of elements in the cars array:

x = len(cars)

## Looping Array Elements

You can use the for in loop to loop through all the elements of an array.

### Example

Print each item in the cars array:

for x in cars:  
  print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_array5)

## Adding Array Elements

You can use the append() method to add an element to an array.

### Example

Add one more element to the cars array:

cars.append("Honda")

## Removing Array Elements

You can use the pop() method to remove an element from the array.

### Example

Delete the second element of the cars array:

cars.pop(1)

### Example

Delete the element that has the value "Volvo":

cars.remove("Volvo")

## Array Methods

Python has a set of built-in methods that you can use on lists/arrays.

|  |  |
| --- | --- |
| **Method** | **Description** |
| [append()](https://www.w3schools.com/python/ref_list_append.asp) | Adds an element at the end of the list |
| [clear()](https://www.w3schools.com/python/ref_list_clear.asp) | Removes all the elements from the list |
| [copy()](https://www.w3schools.com/python/ref_list_copy.asp) | Returns a copy of the list |
| [count()](https://www.w3schools.com/python/ref_list_count.asp) | Returns the number of elements with the specified value |
| [extend()](https://www.w3schools.com/python/ref_list_extend.asp) | Add the elements of a list (or any iterable), to the end of the current list |
| [index()](https://www.w3schools.com/python/ref_list_index.asp) | Returns the index of the first element with the specified value |
| [insert()](https://www.w3schools.com/python/ref_list_insert.asp) | Adds an element at the specified position |
| [pop()](https://www.w3schools.com/python/ref_list_pop.asp) | Removes the element at the specified position |
| [remove()](https://www.w3schools.com/python/ref_list_remove.asp) | Removes the first item with the specified value |
| [reverse()](https://www.w3schools.com/python/ref_list_reverse.asp) | Reverses the order of the list |
| [sort()](https://www.w3schools.com/python/ref_list_sort.asp) | Sorts the list |

**Note:** Python does not have built-in support for Arrays, but Python Lists can be used instead.

# Python Lambda

A lambda function is a small anonymous function.

A lambda function can take any number of arguments, but can only have one expression.

## Syntax

lambda arguments : expression

The expression is executed and the result is returned:

### Example

Add 10 to argument a, and return the result:

x = lambda a : a + 10  
print(x(5))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_lambda)

### Example

Multiply argument a with argument b and return the result:

x = lambda a, b : a \* b  
print(x(5, 6))

### Example

Summarize argument a, b, and c and return the result:

x = lambda a, b, c : a + b + c  
print(x(5, 6, 2))

# Scope of Variable in Python

A variable is only available from inside the region it is created. This is called **scope**.

## Local Scope

A variable created inside a function belongs to the local scope of that function, and can only be used inside that function.

### Example

A variable created inside a function is available inside that function:

def myfunc():  
  x = 300  
  print(x)  
  
myfunc()

### Function Inside Function

As explained in the example above, the variable x is not available outside the function, but it is available for any function inside the function:

### Example

The local variable can be accessed from a function within the function:

def myfunc():  
  x = 300  
  def myinnerfunc():  
    print(x)  
  myinnerfunc()  
myfunc()

## Global Scope

A variable created in the main body of the Python code is a global variable and belongs to the global scope.

Global variables are available from within any scope, global and local.

### Example

A variable created outside of a function is global and can be used by anyone:

x = 300  
def myfunc():  
  print(x)  
myfunc()  
print(x)

### Naming Variables with Global Keyboard

If you operate with the same variable name inside and outside of a function, Python will treat them as two separate variables, one available in the global scope (outside the function) and one available in the local scope (inside the function):

### Example

The function will print the local x, and then the code will print the global x:

x = 300  
def myfunc():  
  x = 200  
  print(x)  
myfunc()  
print(x)

## Global Keyword

If you need to create a global variable, but are stuck in the local scope, you can use the global keyword.

The global keyword makes the variable global.

### Example

If you use the global keyword, the variable belongs to the global scope:

def myfunc():  
  global x  
  x = 300  
  
myfunc()  
  
print(x)

Also, use the global keyword if you want to make a change to a global variable inside a function.

### Example

To change the value of a global variable inside a function, refer to the variable by using the global keyword:

x = 300  
  
def myfunc():  
  global x  
  x = 200  
  
myfunc()  
  
print(x)