Functions

By Aksadur Rahman

aksadur@yahoo.com

Functions

Functions
Parameters/Arguments
Keyword
Parameters/Arguments
Default Parameter Value
Return Statement
Lambda Function
Map and Filter function
Zip Function
Recursion

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.

Built in function

```
print("Hello World")


mylist = ["apple", "banana", "cherry"]
x = len(mylist)


x = max(5, 10)


print('Enter your name:')
x = input()
print('Hello, ' + x)
```

User define function

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

my_function()
```

Parameters/Arguments

```
def my_function(fname):  #Parameters
  print("Good morning!", fname)

my_function("Emil")  #Arguments
my_function("Tobias")
my_function("Linus")
```

```
#This function expects 2 arguments, and gets 2 arguments:
def my_function(fname, lname):
  print(fname,  lname)

my_function("Emil", "Refsnes")
```

Keyword Parameters/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 = "Emil", child2 = "Tobias", child3 = "Linus")
```

Default Parameter Value

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

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

Return Statement

```
def my_function(x):
  return 5 * x

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

```
def add_and_multiple(n1,n2):
    sum = n1 + n2
    mult = n1 * n2
    return sum, mult

x, y = add_and_multiple(2,3)
print(x, y)
```

Docstring

```
def Add(a, b):
    '''
    This will take two parameter
    Return value is integer
    '''
    return a + b


print(Add(10, 20))
print(Add.__doc__)
print(help(Add))
```

Lambda Function

Python Lambda Functions are anonymous function means that the function is without a name.

(lambda parameters**:** expression) (arguments)

```
print((lambda a : a + 10) (5))
```

```
x = lambda a : a + 10
print(x(5))
```

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

```
def cube(y):
  return y * y * y

print(cube(5))

# using the lambda function
lambda_cube = lambda y: y * y * y
print(lambda_cube(5))
```

Map and Filter function

```
def myfunc(a):
  return len(a)

x = map(myfunc, ('apple', 'banana', 'cherry'))
print(list(x))
```

```
def square(x):
    return x*x

num = [1, 2, 3, 4, 5]
result = list(map(square, num))

print(result)
```

```
def myFunc(x):
  if x < 18:
    return False
  else:
    return True

ages = [5, 12, 17, 18, 24, 32]

adults = list(filter(myFunc, ages))

print(adults)
```

```
num = [5, 12, 17, 18, 25, 32]

result = list(filter(lambda x: x%2==0, num))

print(result)
print(num)
```