Functions in Python

Tushar B. Kute,
http://tusharkute.com

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.
- In Python a function is defined using the def keyword:

Function Types

- Basically, we can divide functions into the following two types:
  - Built-in functions - Functions that are built into Python.
  - User-defined functions - Functions defined by the users themselves.

A Sample Function

```
# Function Definition
def show():
    print("Hello World")

# Function Call
show()
```

Parameterized Functions

- Information can be passed to functions as parameter.
- Parameters are specified after the function name, inside the parentheses.
- You can add as many parameters as you want, just separate them with a comma.

Example

```python
# Function definition
def square(x):
    y = x * x
    print("Square: ", y)

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

square(num)  # Variable argument
square(34)   # Constant argument
```

Function Returning values

```python
num = int(input("Enter number: "))

def square(x):
    y = x * x
    return y

def cube(x):
    y = x * square(x)
    return y

print("Square is", square(num))
a = cube(num)
print("Cube is", a)
print("Cube is", num * square(num))
```

Default Parameters

```python
def mul(x = 2, y = 5, z = 10):
    result = x * y * z
    return result

print("Multi is", mul(12, 3, 6)) #216
print("Multi is", mul(12, 3)) #360
print("Multi is", mul(12)) #600
print("Multi is", mul()) #100
print("Multi is", mul(z=2,y=3,x=7)) #42
```

Multi – return statement

- The Python function can return multiple values at a time with multiple independent variables.

Example

```
# Functions returning multiple values
def array(n):
    add = 0
    for x in n:
        add += x
    avg = add / len(n)
    return add, avg        # Multi-return

arr = [43,65,76,11.0,23,67,82]
a, b = array(arr)          # Function Call
print("Addition is", a)
print("Average is %.2f" %b)
```

Recursion

- Python functions have ability to call by themselves. This is termed as Recursion.

```
def factorial(n):
    if n <= 1:
        return 1
    else:
        return n * factorial(n-1)

num = int(input("Enter the number: "))
print("Factorial is", factorial(num))
```

Variable length arguments

```
# Simple function to loop args
def show(*args):
    for a in args:
        print(a)

# Call the function
show(1,2,3)
show('x','y',11, True)
```

Variable length arguments

```
# Simple function to loop
def display(**kwargs):
    for a in kwargs:
        print(a, kwargs[a])

# Call the function
display(name='Rashmi', age=30)
display(pi=3.14)
```

Anonymous Function

- In Python, anonymous function is a function that is defined without a name.
- While normal functions are defined using the def keyword, in Python anonymous functions are defined using the lambda keyword.
- Hence, anonymous functions are also called lambda functions.

Syntax:

- A lambda function in python has the following syntax.

```
lambda arguments: expression
```

- Lambda functions can have any number of arguments but only one expression.
- The expression is evaluated and returned. Lambda functions can be used wherever function objects are required.

Example:

```
# Use of lambda functions

square = lambda x: x ** 2

# Output: 144

print(square(12))
```

Using lambda function

- We use lambda functions when we require a nameless function for a short period of time.
- In Python, we generally use it as an argument to a higher-order function (a function that takes in other functions as arguments).
- Lambda functions are used along with built-in functions like filter(), map() etc.

The filter( )

- The filter() function in Python takes in a function and a list as arguments.
- The function is called with all the items in the list and a new list is returned which contains items for which the function evaluates to True.

Example:

```
# a list contains both even and odd numbers.
seq = [0, 11, 2, 3, 5, 8, 13]

# result contains odd numbers of the list
result = filter(lambda x: x > 5, seq)
print(list(result))

# result contains even numbers of the list
result = filter(lambda x: x % 2 == 0, seq)
print(list(result))
```

The map( )

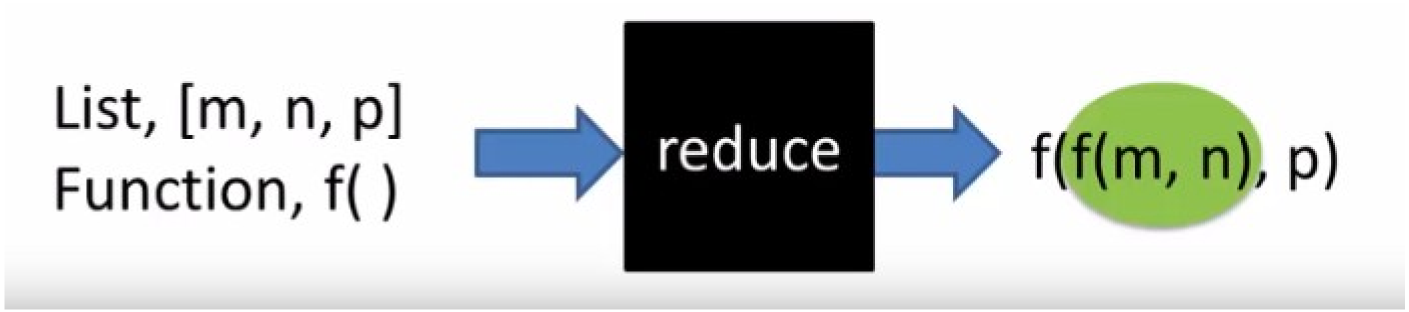
- The map() function in Python takes in a function and a list.
- The function is called with all the items in the list and a new list is returned which contains items returned by that function for each item.

Example:

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

# We square all numbers using map()
numbers = (1, 2, 3, 4)
result = map(square, numbers)
print(list(result))

# List of strings
l = ['sat', 'bat', 'cat', 'mat']

# map() can listify the list of strings
test = map(list, l)
print(list(test))
```

The reduce()

- The reduce() function accepts a function and a sequence and returns a single value calculated as follows:
  - Initially, the function is called with the first two items from the sequence and the result is returned.
  - The function is then called again with the result obtained in step 1 and the next value in the sequence. This process keeps repeating until there are items in the sequence.



The reduce()

```
from functools import reduce

seq=[2,3,4,5,6]
multiply=reduce(lambda a,b:a*b,seq)

print(multiply)
```

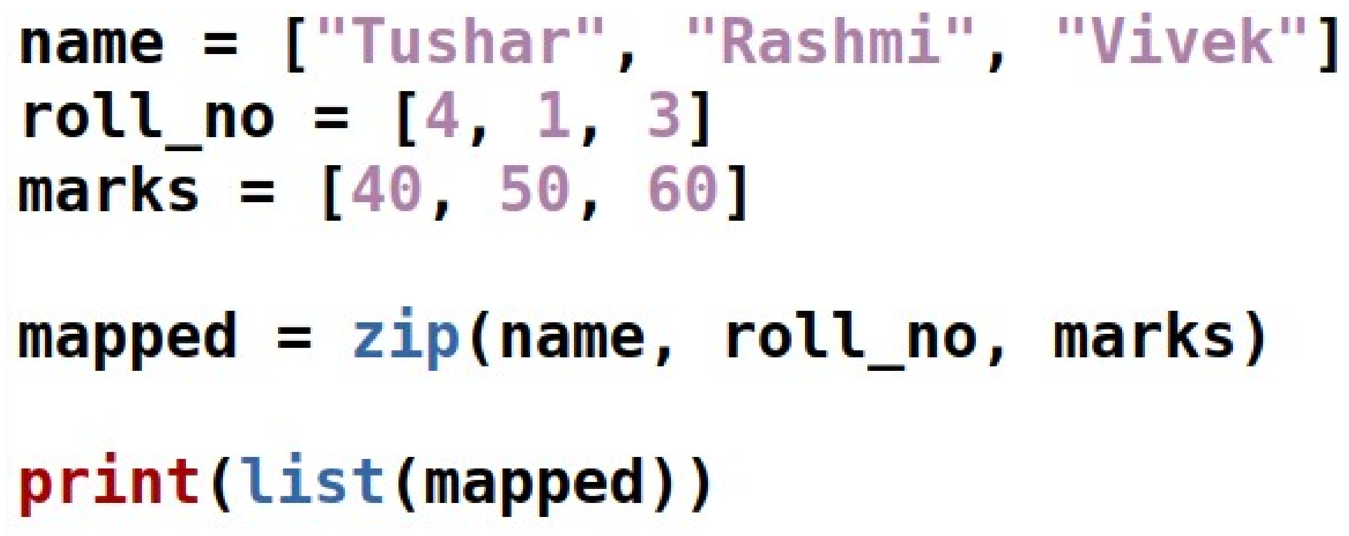
The zip()

- The zip() function take iterables (can be zero or more), makes iterator that aggregates elements based on the iterables passed, and returns an iterator of tuples.
- Syantax: zip(*iterables)

Returns from zip()

- The zip() function returns an iterator of tuples based on the iterable object.
  - If no parameters are passed, zip() returns an empty iterator
  - If a single iterable is passed, zip() returns an iterator of 1-tuples. Meaning, the number of elements in each tuple is 1.
  - If multiple iterables are passed, ith tuple contains ith Suppose, two iterables are passed; one iterable containing 3 and other containing 5 elements. Then, the returned iterator has 3 tuples. It’s because iterator stops when shortest iterable is exhaused.

Example:

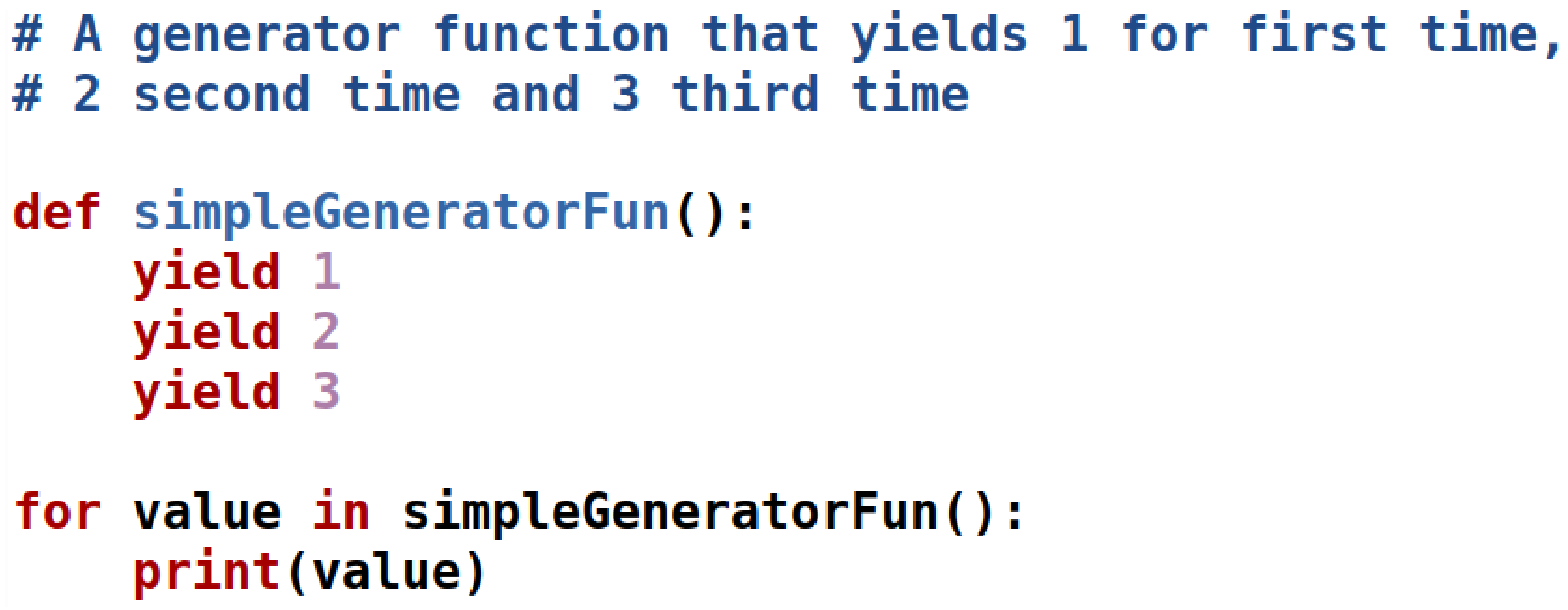


```
name = ["Tushar", "Rashmi", "Vivek"]
roll_no = [4, 1, 3]
marks = [40, 50, 60]

mapped = zip(name, roll_no, marks)

print(list(mapped))
```

Generator function

- A generator-function is defined like a normal function, but whenever it needs to generate a value, it does so with the yield keyword rather than return.
- If the body of a def contains yield, the function automatically becomes a generator function.

Generator function



```python
# A generator function that yields 1 for first time,
# 2 second time and 3 third time

def simpleGeneratorFun():
    yield 1
    yield 2
    yield 3

for value in simpleGeneratorFun():
    print(value)
```

The global keyword

- In Python, global keyword allows you to modify the variable outside of the current scope. It is used to create a global variable and make changes to the variable in a local context.
- Rules of global Keyword
  - When we create a variable inside a function, it’s local by default.
  - When we define a variable outside of a function, it’s global by default. You don’t have to use global keyword.
  - We use global keyword to read and write a global variable inside a function.
  - Use of global keyword outside a function has no effect

Example:

```
c = 0 # global variable

def add():
    global c
    c = c + 2 # increment by 2
    print "Inside add():", c

add()
print "In main:", c
```

Exercises

- Write a Python function to find the Max of three numbers.
- Write a Python function to check whether a number is in a given range.
- Write a Python program to print the even numbers from a given list. Go to the editor.
  - Sample List : [1, 2, 3, 4, 5, 6, 7, 8, 9]
  - Expected Result : [2, 4, 6, 8]
- Write a Python function that checks whether a passed string is palindrome or not.

Thank you

@mitu_skillologies

/mITuSkillologies

@mitu_group

/company/mitu-skillologies

c/MITUSkillologies

Web Resources
https://mitu.co.in
http://tusharkute.com

contact@mitu.co.in

tushar@tusharkute.com