Step0 - Functions

May 8, 2018

0.0.1 User Defined Functions

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.

```
In [1]: # simple function to add two numbers
        def sum_two_numbers(a, b):
            return a + b

        # after this line x will hold the value 3!
        x = sum_two_numbers(1,2)
        print x
```

```
3
```

0.0.2 function with arguments

```
In [2]: def sum_two_numbers(a, b = 10):
            return a + b

        print sum_two_numbers(10)
        print sum_two_numbers(10, 5)
```

```
20
15
```

0.0.3 Scope of variables

The scope of a variable determines the portion of the program where you can access a particular identifier. There are two basic scopes of variables in Python

- Global variables
- Local variables

```
In [3]: # Global variable
        a = 10
```

```
        # Simple function to add two numbers
        def sum_two_numbers(b):
            return a + b

        # Call the function and print result
        print sum_two_numbers(10)
```

```
20
```

0.0.4 Default argument

A default argument is an argument that assumes a default value if a value is not provided in the function call for that argument.

```
In [6]: # Simple function to add two number with b having default value of 10
        def sum_two_numbers(a, b = 10):
            return a + b

        # Call the function and print result
        print sum_two_numbers(10)

        print sum_two_numbers(10, 5)
```

```
20
15
```

0.0.5 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-lengtharguments and are not named in the function definition, unlike required and default arguments. The *args and **kwargs is a common idiom to allow arbitrary number of arguments.

The *args will give you all function parameters as a tuple:

```
In [7]: # Simple function to loop through arguments and print them
        def foo(*args):
            for a in args:
                print a

        # Call the function
        foo(1,2,3)
```

```
1
2
3
```

The **kwargs will give you all keyword arguments except for those corresponding to a formal parameter as a dictionary

```
In [8]: # Simple function to loop through arguments and print them
        def foo(**kwargs):
            for a in kwargs:
                print a, kwargs[a]


        # Call the function
        foo(name='John', age=27)
```

```
age 27
name John
```

Reference: Mastering machine learning using python in six-steps