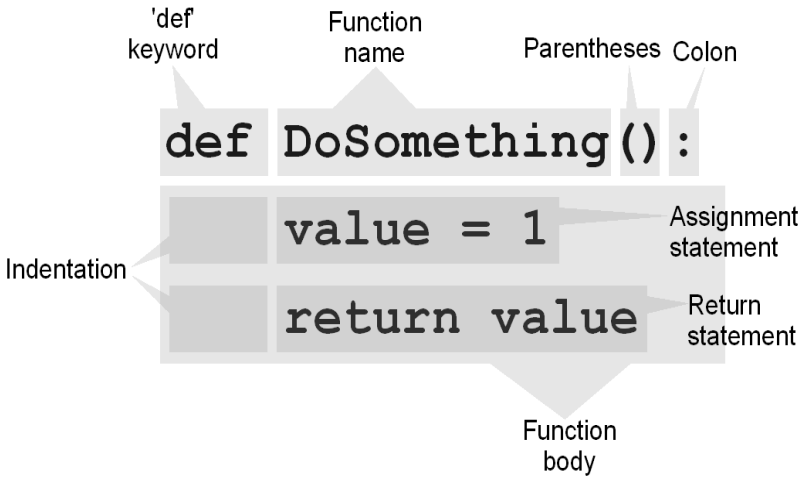
Python's Functions

- Functions in Python are very flexible to define and call
- Syntax



- call: *DoSomething()*
- **Arguments**
  - No Argument
  - One Argument
  - Multiple Arguments
  - Default or Keyword Argument
  - No fixed value of non keyword arguments
  - No fixed value of keyword arguments
- **Return**
  - None
  - One Value
  - Multiple values
- **Examples**
  - Print “Awesome Guide” inside a function
  - def awesome():
        print “Awesome Guide”
        # by default return none
    output = awesome()
    Awesome Guide
    output
    # does not print any value because None (default return)

  - Calculate area of a squre
  - def squre_area(arm):
            area = arm *4
            return area
    ouput = squrea_area(5)
    output
    20
    # one argument and one return statement

- Calculate area and perimeter of a circle
- def area_circle(radius):
        return 3.14*radius**2, 2*3.14*radius

  area, peri = area_circle(4)
  area
  20.125
  peri
  12.30
  # one argument and multiple return

- Calculate sum and multiplication of two operands
- def sum_multi(a, b):
        return a+b, a*b

  sumval, multival = sum_multi(4, 5)
  sumval
  9
  multival
  20
  # multiple argument , multiple return statements

- Calculate sum and multiplication of two operands while one operand have defualt value
- def sum_list_default(a, b=3):
   return a+b, a*b

  sumval, multival = sum_list_default(20)
  sumval
  23
  multival
  60
  # defualt value maintained in above example

  sumval, multival = sum_list_default(20, 5)
  sumval
  25
  multival
  100
  # default value overwritten in above example

- Calculate sum of all non keyword arguments and return all keyword arguments
- def last_fun(*args, **kwargs):
        return sum(args), kwargs

  sumval, dictval = last_fun(2,5,8,7, a=2, b=”hello”)
  sumval
  22
  dictval
  {'a':2, 'b':'hello'}

  sumval, dictval = last_fun(2, 4, a=3, b=5, c=”newvalue”)
  sumval
  6
  dictval

```
{'a':3, 'b':5, c:'newvalue'}

sumval, dictval = last_fun(2, a = 5, 6)
# this will give error
# after a keyword argument , non keyword argument is not allowed
```

Written By:
Vikas Kumar Sharma
vikas.pypy@gmail.com
defpy.com

Feedback, Queries and Suggestions are invited on above mail id.