User Defined Function – Unit 3

By : Bhavika Vaghela
Asst. Prof.
PICA - Parul University

- **function** is a named sequence of statements that performs a computation.
- When you define a function, you specify the name and the sequence of statements.
- Later, you can “call” the function by name.

<u>Python Support two types of function</u>

1) **Inbuilt function** like len(), type, pow(), sqrt(), append() and many more.
2) **User defined function**: which declare by user using **def** keyword.

- A function is a group of related statements that performs a specific task.

Syntax :

```
def <function name> (parameter /argument list):
        //body of function.
```

- It is common to say that a function “takes” an argument and “returns” a result. The result is called the return value.
- It is not mandatory that each function must have parameter or argument list.
- Again return is also not mandatory you can directly print result in function body. but its good habit to return result.

Type Conversion Function

- Type conversion function is use to convert data type of value.
- **int()** function can convert floating-point values to integers, but it doesn’t round off; it chops off the fraction

```
>>> int(3.99999)   #Output   3
>>> int(-2.3)      #Output   -2
>>> int('Hello')   #Output   ValueError: invalid literal for int(): Hello
```

- **float()** converts integers and strings to floating-point numbers.

```
>>> float(32)         # output 32.0
>>> float('3.14159')  #output  3.14159
```

Note : here 3.14159 is string because it pass within single quote.

- str() string function type cast the data or value into string form.
- It does not mean that it will give a character of value 65 or any other ASCII value.

```
>>> str(32)      #output    '32'
>>> str(3.14159)     #output   '3.14159‘
```

Math functions

- Python has a **math** module that provides most of the familiar mathematical functions.
- A **module** is a file that contains a collection of related functions.
- Before use the math module, we have to import it:
- `>>> import math`

- When you import package it will create modulo object of object of that package which you imported.
- To access one of the functions, you have to specify the name of the module and the name of the function, separated by a dot (also known as a period). This format is called **dot notation**.
- Like math.pow(x,y) so here pow() is funciton name.
- If you import package like,

import math as m than m.pow(x,y)

- Like math.sin(), math.log10() and many more functions.
- The expression math.pi gets the variable pi from the math module. The value of this variable is an approximation of p, accurate to about 15 digits.

Composition

- One of the most useful features of programming languages is their ability to take small building blocks and **compose** them.
- For example, the argument of a function can be any kind of expression, including arithmetic operators:

x = math.sin(degrees / 360.0 * 2 * math.pi)   And even function calls:

x = math.exp(math.log(x+1))

- Almost anywhere you can put a value, you can put an arbitrary expression, with one exception: the left side of an assignment statement has to be a variable name
- **>>> minutes = hours * 60 # right**
- **>>> hours * 60 = minutes # wrong!**

Add new function or how to add or create user define function

- Any function is basically divided into three main part.

1) **Header** : is a first line of function definition.

2) **body:** The sequence of statements inside a function definition.

3) **parameter:** A name used inside a function to refer to the value passed as an argument.

- **function call**: A statement that executes a function. It consists of the function name followed by an argument list.
- **argument**: A value provided to a function when the function is called. This value is assigned to the corresponding parameter in the function.

Syntax :

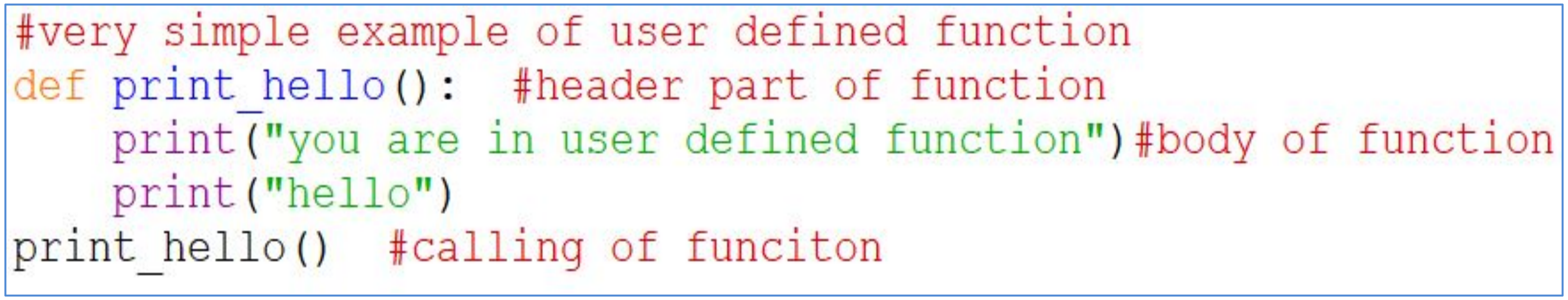
```
def <function name> ( arg1, arg2…):
        //function executable statements
          return statement if any
```

- Here def is keyword which use to define user defined function.
- As a function name you can give any but not reserved keyword.
- Argument list is not mandatory one can leave it as blank also.
- Return statement also not mandatory but its good habit to write return statement.
- **Question is : can we have more than one return statements in same function?**

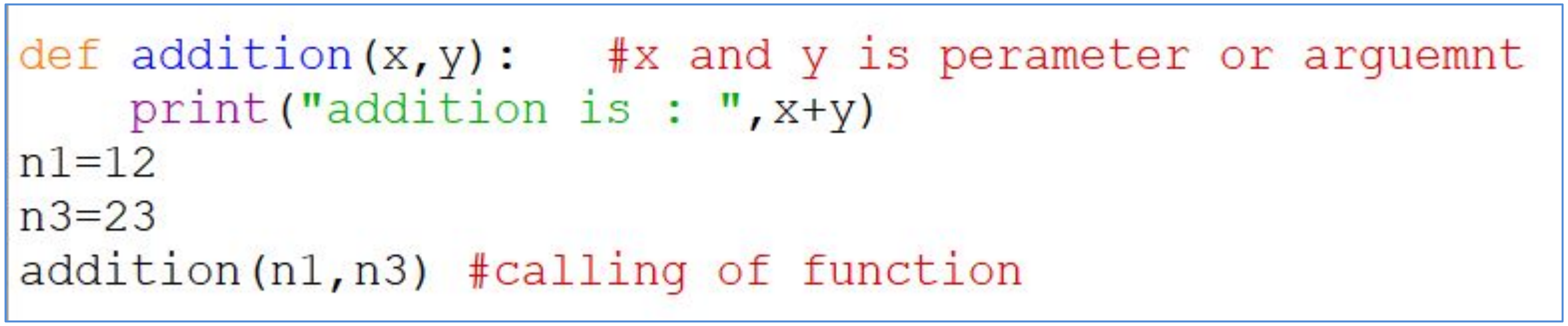
- The first line of the function definition is called the **header**; the rest is called the **body**. The header has to end with a colon and the body has to be indented.
- The **return** statement is used to exit a function and go back to the place from where it was called.
- Let’s discuss about function with very simple example.

```
def myfirst_fun():
    print(“hello this my first function”)
def mysec_fun():
    print(“second function”)

for I in range(1,6):
      mysec_fun()    #calling of function

myfirst_fun()     #calling of function
```

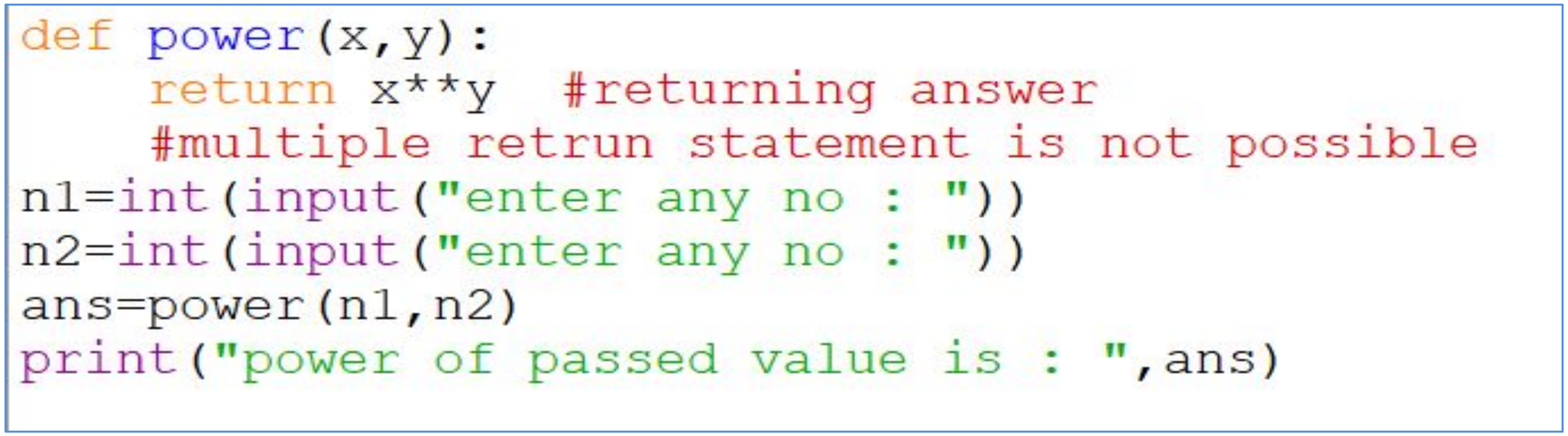
Few Example of UDF



```python
#very simple example of user defined function
def print_hello():  #header part of function
    print("you are in user defined function")#body of function
    print("hello")
print_hello()  #calling of funciton
```



```python
def addition(x,y):   #x and y is perameter or arguemnt
    print("addition is : ",x+y)
n1=12
n3=23
addition(n1,n3) #calling of function
```



```python
def power(x,y):
    return x**y  #returning answer
    #multiple retrun statement is not possible
n1=int(input("enter any no : "))
n2=int(input("enter any no : "))
ans=power(n1,n2)
print("power of passed value is : ",ans)
```

Why to use function? Or advantage of function

- we can avoid rewriting same logic/code again and again in a program.
- We can call python functions any number of times in a program and from any place in a program.
- We can track a large python program easily when it is divided into multiple functions.
- **Reusability is the main achievement of python functions.**
- Functions help break our program into smaller and modular chunks. As our program grows larger and larger, functions make it more organized and manageable.

How you can call function in python.

- In python, a function must be defined before the function calling otherwise the python interpreter gives an error. Once the function is defined, we can call it from another function or the python prompt. To call the function, use the function name followed by the parentheses.
- **In python, all the functions are called by reference,** i.e., all the changes made to the reference inside the function revert back to the original value referred by the reference.
- **However, there is an exception in the case of immutable objects** since the changes made to the mutable objects like string do not revert to the original string rather, a new string object is made, and therefore the two different objects are printed.

Let’s see example of mutable and immutable argument passing.

```
#passing mutable  data type list as function argument.
def change_list(list1):
   list1.append(20);
   list1.append(30);
   print("list inside function = ",list1)

#defining the list
list1 = [10,30,40,50]

#calling the function
change_list(list1);
print("list outside function = ",list1);
```

- Here list1 is change after calling the function.

In below example string pass as argument

```
#passing string as argument

def change_string (str):
    str = str + " Hows you";
    print("printing the string inside function :",str);

string1 = "Hi I am there"

#calling the function
change_string(string1)

print("printing the string outside function :",string1)
```

- Here string is not going to change because its immutable so you people also try with passing tuple, set

Types of arguments

- Required arguments
- Keyword arguments
- Default arguments
- Variable-length arguments

Required Argument : As far as the required arguments are concerned, these are the arguments which are required to be passed at the time of function calling with the exact match of their positions in the function call and function definition. If either of the arguments is not provided in the function call, or the position of the arguments is changed, then the python interpreter will show the error.

e.g. of required argument

```
#the function simple_interest accepts three arguments and returns the simple inte
rest accordingly
def simple_interest(p,t,r):
    return (p*t*r)/100
p = float(input("Enter the principle amount? "))
r = float(input("Enter the rate of interest? "))
t = float(input("Enter the time in years? "))
print("Simple Interest: ",simple_interest(p,r,t))
```

<u>Other example</u>

```
#the function calculate returns the sum of two arguments a and b
def calculate(a,b):
    return a+b
calculate(10) # this causes an error as we are missing a required arguments b.
```

Keyword Argument

The name of the arguments is treated as the keywords and matched in the function calling and definition. If the same match is found, the values of the arguments are copied in the function definition.

```
def func(name,message):
    print("printing the message with",name,"and ",message)
func(name = "John",message="hello")



def simple_interest(p,t,r):
    return (p*t*r)/100
print("Simple Interest: ",simple_interest(t=10,r=10,p=1900))
```

****here its match the keyword name mean name of arguments.**

```
def simple_interest(p,t,r):
    return (p*t*r)/100

print("Simple Interest: ",simple_interest(time=10,rate=10,principle=1900)) # doesn't
find the exact match
```

Default Arguments

- Python allows us to initialize the arguments at the function definition. If the value of any of the argument is not provided at the time of function call, then that argument can be initialized with the value given in the definition even if the argument is not specified at the function call.

```
def printme(name,age=22):
    print("My name is",name,"and age is",age)
printme(name = "john") #the variable age is not passed into the function




// another example

def printme(name,age=22):
    print("My name is",name,"and age is",age)
printme(name = "john")
#the variable age is not passed into the function however the default value of age is
considered in the function
printme(age = 10,name="David") #the value of age is overwritten here, 10 will be pri
nted as age
```

Variable length Argument

- In the large projects, sometimes we may not know the number of arguments to be passed in advance. In such cases, Python provides us the flexibility to provide the comma separated values which are internally treated as tuples at the function call.
- However, at the function definition, we have to define the variable with * (star) as *<variable - name >.

```
def printme(*names):
    print("type of passed argument is ",type(names))
    print("printing the passed arguments...")
    for name in names:
        print(name)
printme("john","David","smith","nick")
```

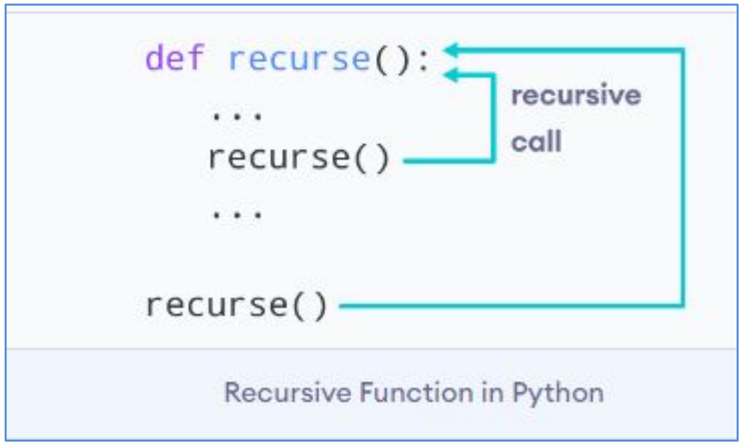
Scope of Variable

It mean how long and at which location your variable is accessible.

1) **Local variable** : Declare within loop or function or block. Which is not access out side of it.
2) **Global variable** : is declare in global space and which access any where within a program.

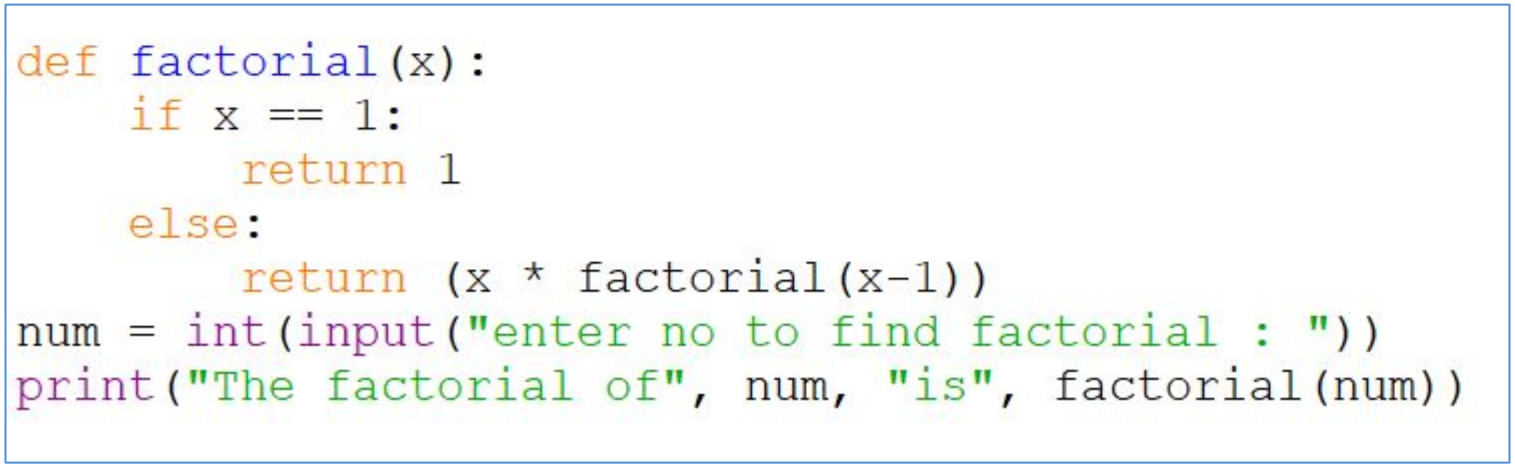
Function recursion : function call itself within a function is known as recursion call or recursion function. E.g. call function to find factorial, find prime number within given range etc.

Example of recursive function



```
def recurse():
    ...
    recurse()
    ...

recurse()
```

Recursive Function in Python



```
def factorial(x):
    if x == 1:
        return 1
    else:
        return (x * factorial(x-1))
num = int(input("enter no to find factorial : "))
print("The factorial of", num, "is", factorial(num))
```

FruitFul Function Some of the functions we are using, such as the math functions, yield results; for lack of a better name, I call them **fruitful functions**.

Void Function

- The function which not return any thing.