"Success is more a function of consistent common sense than it is of genius"

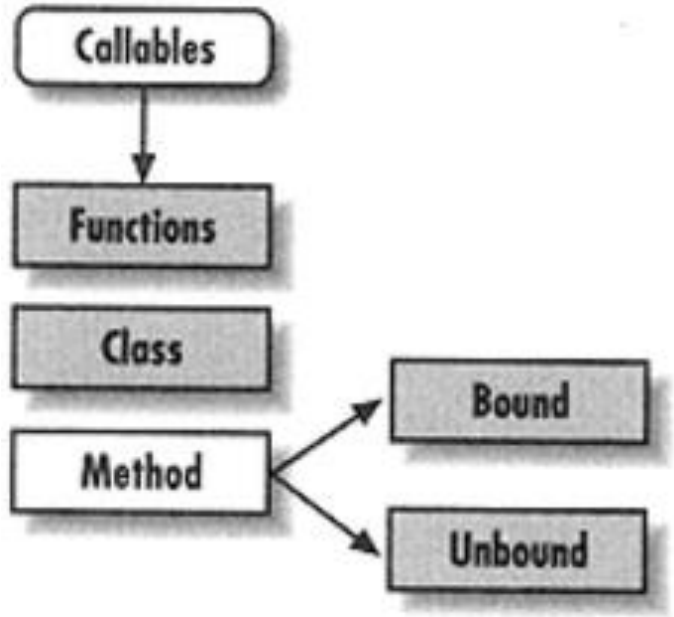
(An Wang, Computer engineer and inventor, 1920 - 1990)

Python-Loops, Functions

Dr. Sarwan Singh

Agenda

- Functions – arguments
- Type of function arguments
- Call by reference/value



One guiding principle of Python code is that
"explicit is better than implicit"

रा.इ.सू.प्रौ.सं
NIELIT
Chandigarh

O Level
M3-R5

Module Unit	Written Marks (Max.)
Introduction to Programming, Algorithm and Flowcharts to solve problems	20
Introduction to Python, Operators, Expressions and Python Statements, Sequence data types	30
Functions, File Processing, Modules	40
NumPy Basics	10
Total	100

Module Unit	Duration (Theory) in Hours	Duration (Practical) in Hours
• **Introduction to Programming**	2	3
• **Algorithm and Flowcharts to solve problems**	6	9
• **Introduction to Python**	2	3
• **Operators, Expressions and Python Statements**	10	15
• **Sequence data types**	6	9
• **Functions**	10	15
• **File Processing**	6	9
• **Modules**	2	3
• **NumPy Basics**	4	6
Total	48	72

रा.इ.सू.प्रौ.सं
NIELIT
Chandigarh

Detailed syllabus (marks)

- **Introduction to Programming (20)**
- **Algorithms and Flowcharts to Solve Problems**

- **Introduction to Python (30)**
- **Operators, Expressions and Python Statements**
- **Sequence Data Types**

- **Functions (40)**
  - Top-down approach of problem solving, Modular programming and functions, Function parameters, Local variables, the Return statement, DocStrings, global statement, Default argument values, keyword arguments, VarArgs parameters.
  - Library function-input(), eval(),print(), String Functions: count(), find(), rfind(), capitalize(), title(), lower(), upper(), swapcase(), islower(), isupper(), istitle(), replace(), strip(), lstrip(), rstrip(), aplit(), partition(), join(), isspace(), isalpha(), isdigit(), isalnum(), startswith(), endswith(), encode(), decode(), String: Slicing, Membership, Pattern Matching, Numeric Functions: eval(), max(), min(), pow(), round(), int(), random(), ceil(), floor(), sqrt(), Date & Time Functions, Recursion.

- **File Processing**
  - Concept of Files, File opening in various modes and closing of a file, Reading from a file, Writing onto a file, File functions-open(), close(), read(), readline(),readlines(),write(), writelines(),tell(),seek(), Command Line arguments.

- **Scope and Modules**
  - Scope of objects and Names, LEGB Rule
  - Module Basics, Module Files as Namespaces, Import Model, Reloading Modules.

- **NumPy Basics (10)**

loops

```
while expression :
        statements()
```

```
i=0
while (i<5) :
    print (i, 'Jai Ho')
    i=i+1
```

```
0 Jai Ho
1 Jai Ho
2 Jai Ho
3 Jai Ho
4 Jai Ho
```

```
while expression :
        statements()
else :
        statements()
```

```
i=0
while (i<5) :
    print (i, 'Jai Ho')
    i=i+1
else :
    print (i, ' Its over now')
```

```
0 Jai Ho
1 Jai Ho
2 Jai Ho
3 Jai Ho
4 Jai Ho
5  Its over now
```

for loop

for iteratingVariable in sequence

statement/s

```
In [3]: states=['J&K', 'HimachalPradesh','Punjab','Delhi']
        for st in states:
            print (st)
```

```
J&K
HimachalPradesh
Punjab
Delhi
```

```
In [4]: for st in range(len(states)):
            print (states[st])
```

```
J&K
HimachalPradesh
Punjab
Delhi
```

```
In [5]: for alpha in 'India':
            print(alpha)
```

```
I
n
d
i
a
```

for iteratingVariable in sequence

statement/s

else:

statement/s

```
In [7]: for st in range(len(states)):
            print (states[st])
        else :
            print('------Its over ------ ')
```

```
J&K
HimachalPradesh
Punjab
Delhi
------Its over ------
```

For loop: Example

1.
```
for n in range(21, 0, -3):
        print(n, '', end='')
```

Output: 21 18 15 12 9 6 3

2.
```
for n in range(1000) :
    print(n, end=‘ ‘)
```

Output: 0, 1, 2, . . . , 999.

3.
```
sum = 0
for i in range(1, 100):
        sum += i
print(sum)
```

Output: adds nos from 1 to 99

Iteration : for

range(10) → 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

range(1, 10) → 1, 2, 3, 4, 5, 6, 7, 8, 9

range(1, 10, 2) → 1, 3, 5, 7, 9

range(10, 0, -1) → 10, 9, 8, 7, 6, 5, 4, 3, 2, 1

range(10, 0, -2) → 10, 8, 6, 4, 2

range(2, 11, 2) → 2, 4, 6, 8, 10

range(-5, 5) → −5, −4, −3, −2, −1, 0, 1, 2, 3, 4

range(1, 2) → 1

range(1, 1) → (empty)

range(1, -1) → (empty)

range(1, -1, -1) → 1, 0

range(0) → (empty)

Exercise

1. Write program to check whether given number is prime or not
2. Write a program to find those numbers which are divisible by 7 and multiple of 5, between 1500 and 2700 (both included).
3. Write a Python program to get the Fibonacci series between 0 to 50.
4. Write a program to construct the pattern, using a nested for loop.

```
*
**
***
****
*****
****
***
**
*
```



```
    *
  * * *
* * * * *
```

```
1
22
333
4444
55555
666666
7777777
88888888
999999999
```

Exercise

1. Write a program to print the table of given number entered by the user
2. Write a program which can compute the factorial of a given numbers.

5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50

Loop Control Statements

Break : Terminates loop statement

```
for alpha in 'Greatness':
    if alpha == 'n':
        break
    print ('letter ', alpha)
```

```
letter  G
letter  r
letter  e
letter  a
letter  t
```

continue : returns the control to the beginning of the while/for loop

```
for alpha in 'Greatness':
    if alpha == 'n':
        continue
    print ('letter ', alpha)
```

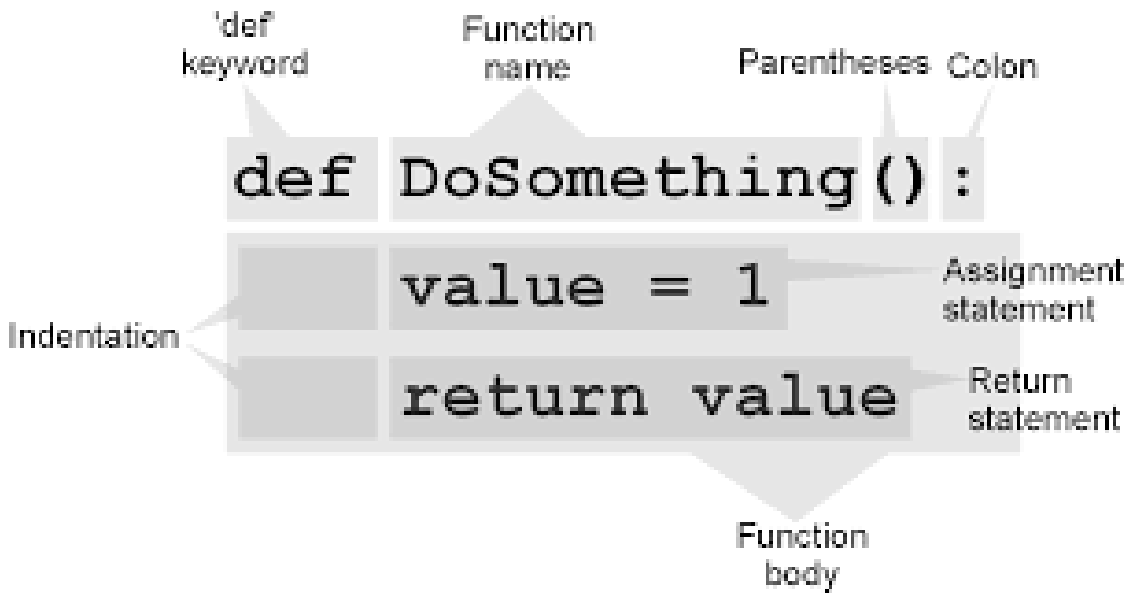
```
letter  G
letter  r
letter  e
letter  a
letter  t
letter  e
letter  s
letter  s
```

pass : is used when a statement is required syntactically but you do not want any command or code to execute

```
for alpha in 'Greatness':
    if alpha == 'n':
        pass
        print ('Pass block')
    print ('letter ', alpha)
```

```
letter  G
letter  r
letter  e
letter  a
letter  t
Pass block
letter  n
letter  e
letter  s
letter  s
```

Functions



* Organises related code in blocks, so that it can be efficiently reused
* Better modularity.

Best practices

- Function should be written keeping in mind process of incremental development.
- Scaffolding : code like print statements used for building the program but is not part of the final product
- Initially write individual statements , later consolidate multiple statements in compound statement
- Composition is ability of to call function from another function

```
def functionname( arg1 , arg2, arg3,
      arg4=default_Value ) :
    ‘’’function_docstring’’’
    statements…..
    return [expression]
```

- Function can be system defined (e.g. print) or user defined
- Function can have variable number of arguments (e.g. print)
- All parameters (arguments) in the Python language are passed by reference.

```
def onePlus(a):
    '''function increments the passed argument by one'''
    return a+1
```

```
onePlus.__doc__
```

```
'function increments the passed argument by one'
```

```
onePlus(10)
```

```
11
```

```
print (onePlus(21))
```

```
22
```

रा.इ.सू.प्रौ.सं
NIELIT
Chandigarh

Function arguments

- Required arguments
  - Error on calling > onePlus()
- Keyword arguments
- Default arguments
  - addMe(a, b=10)
- Variable-length arguments

```
[1] def addMe(a,b):
      c=a+b
      return(c)
```

```
[2] addMe(10,20)
```

```
30
```

```
[3] x=10
    addMe(x,20)
```

```
30
```

```
[4] addMe(b=30,a=20)
```

```
50
```

```
[5] addMe(b=30,20)
```

```
  File "<ipython-input-5-9b58c5f1abf7>", line 1
    addMe(b=30,20)
                ^
SyntaxError: positional argument follows keyword argumer
```

SEARCH STACK OVERFLOW

Function arguments

- Keyword argument
- Default argument
- Variable length arguments

```
def functionname([formal_args,] *var_args_tuple
):
        for var in vartuple:
                print var
```

```
def fibSeries(n):
    a, b = 0, 1
    while b < n:
        print (b,)
        a, b = b, a+b
```

```
fibSeries(50)
```

```
1
1
2
3
5
8
13
21
34
```

Variable argument passing

```
def printVariables( arg1, *vararg ):
        print (“Arguments Received: “)
        print (arg1 )
        for v in vararg:
                print v
        return;
```

Function calling

```
printVariables( 10 )
printVariables( 70, 60, 50 )
```

Default arguments

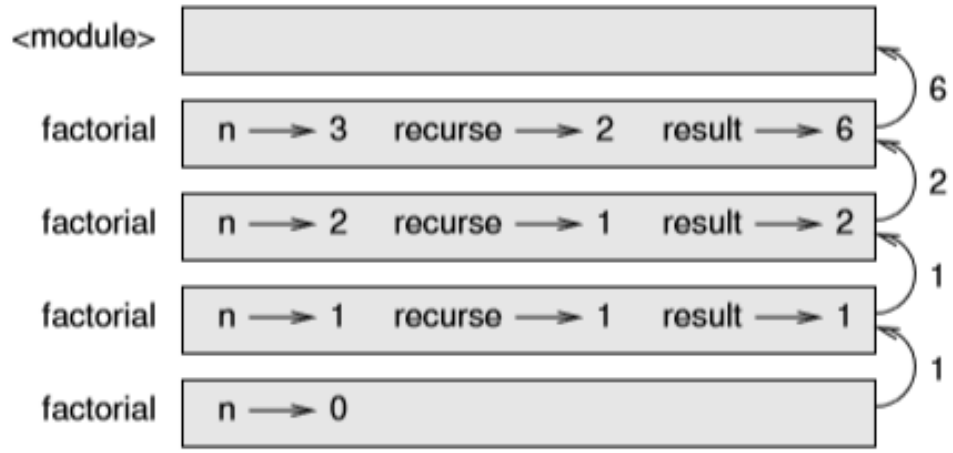
```
def studentInfo( name, fee = 3500 ):
        print ("Name: ", name)
        print ("Fee :  ", fee )
```

Function calling

```
studentInfo(‘Amrit’ )
studentInfo(‘Amrit’ ,2000)
studentInfo( fee=5000, name=“Amrit" )
studentInfo( name=“Amrit" )
```

Recursion

```
def factorial(n):
        if n == 0:
                return 1
        else:
                recurse = factorial(n-1)
                result = n * recurse
        return result
```



Ex: write recursive function to generate/print Fibonacci series

Source : *thinkpython*

Anonymous Functions

- anonymous functions are not declared using the *def* keyword
- uses *lambda* keyword
- Syntax : lambda [arg1 [,arg2,.....argn]]:expression
- Example: sum = lambda arg1, arg2 : arg1 + arg2;
- Calling: print "Value of total : ", sum(10, 20)

Call by value / Reference

```
def changeMe(x):
    print ('Value received    : ',x, '  id(x)=',id(x))
    x=x+20
    print ('Value changed     : ',x, '  id(x)=',id(x))
    return
```

```
a=20
print ('Value before calling :',a , '  id(a)=',id(a))
changeMe(a)
print ('Value after calling  :',a , '  id(a)=',id(a))
```

```
Value before calling : 20   id(a)= 1629250304
Value received     :  20   id(x)= 1629250304
Value changed      :  40   id(x)= 1629250944
Value after calling  : 20   id(a)= 1629250304
```

Call by value / Reference

```
def changeList(x):
    print ('Value received    : ',x, '  id(x)=',id(x))
    x+=[5,6]
    print ('Value changed     : ',x, '  id(x)=',id(x))
    return
```

```
arr=[1,2,3,4]
print ('List before calling :',arr , '  id(a)=',id(arr))
changeList(arr)
print ('List after calling  :',arr , '  id(a)=',id(arr))
```

```
List before calling : [1, 2, 3, 4]   id(a)= 2214105128328
Value received    :  [1, 2, 3, 4]   id(x)= 2214105128328
Value changed     :  [1, 2, 3, 4, 5, 6]   id(x)= 2214105128328
List after calling  : [1, 2, 3, 4, 5, 6]   id(a)= 2214105128328
```

```
arr=[1,2,3,4]
print ('List before calling :',arr , '  id(a)=',id(arr))
changeList(arr[:]) #passing a copy ( shallow copy )
print ('List after calling  :',arr , '  id(a)=',id(arr))
```

```
List before calling : [1, 2, 3, 4]   id(a)= 2214105040584
Value received    :  [1, 2, 3, 4]   id(x)= 2214105177096
Value changed     :  [1, 2, 3, 4, 5, 6]   id(x)= 2214105177096
List after calling  : [1, 2, 3, 4]   id(a)= 2214105040584
```

Exercise

1. Write a function to check the argument passed is part of Fibonacci series or not . isInFibo(x) returns true/false
2. Write a function to check the argument passed is prime number. isPrime(x) returns true/false
3. Write a program to calculate the arithmetic mean of a variable number of values.
4. Write a function to find letter in a word def find(word, letter)
5. Modify above function with third parameter specifying where to start the search def find (word, letter, start)

Exercise

1. *ROT13 is a weak form of encryption that involves "rotating" each letter in a word by 13 places.* Write a function to encrypt- decrypt passed text according to ROT13.

   rotCipher(string, 'e') / rotCipher(string, 'd')

1. Write a function to check whether passed string is palindrome or not *: isPalindrome(word)*

Exercise

1. Write function to check whether input number is even or odd.
2. Write a program to accept marks from user and print the division.
3. Write a function to return sum of digits of given number.
4. Write a program to print pascal's triangle
5. Write a function to check passed string

   is a pangram (pangram- words or sentences

   containing every letter of the alphabet at least once

e.g. "The quick brown fox jumps over the lazy dog"