



Functions

Based on CBSE Curriculum

Class-12

Lesson-2

By:

Neha Tyagi, PGT CS

KV No-5, 2nd Shift

Jaipur

Functions in Python

- Function is a collection of statements which is made to perform a specific task.
- To Execute function we have to call it in the program.
- Instead of writing a large program we can write small functions as a specific part of program to accomplish the task.
- Once written a function can also be used in other programs as library functions.
- Functions can be categorized in three types-
 1. Built-in
 2. Modules
 3. User Defined

Built-in Functions

- These are the functions which are predefined in python we have to just call them to use.
- Functions make any programming language efficient and provide a structure to language.
- Python has many built-in functions which makes programming easy, fast and efficient.
- They always reside in standard library and we need not to import any module to use them.
- We will study about some built-in function in next slide.

Built-in Functions. . .

1. Type Conversion Functions: These are the functions which convert the values from one type to another-

1. `int()` – To convert the string into integer.
2. `str()` – To convert any value into string.
3. `float()` – To convert string into float.

2. Input Functions: This function is used to take input from user in the form of string.

e.g. `name=input("Enter your name : ")`

3. eval function: This function is used to evaluate the value of a string.

e.g. `x=eval("45+10")`

`print(x)`

`# answer will be 55`

Built-in Functions. . .

3. min Function: This function returns the smallest value among given list of values.

4. max Function: This function returns the biggest value among given list of values.

5. abs Function: This function returns the absolute value of any integer which is always positive.

```
>>> min(12, 5, 7, -102)
-102
>>> max(12, 5, 7, -102)
12
>>> abs(-22)
22
>>> abs(22)
22
```

Built-in Functions. . .

6. type Function: This function is used to identify the type of any value or variable.

7. len Function: This function returns the length of given string.

8. round Function: This function returns the rounded number of given number up to given position.

```
>>> type(10)
<class 'int'>
>>> a="hello"
>>> type(a)
<class 'str'>
>>> b=20
>>> type(b)
<class 'int'>
>>> len(a)
5
>>> round(11.12345,2)
11.12
>>> round(11.12345,3)
11.123
>>> round(11.12345,0)
11.0
```

Built-in Functions. . .

9. range Function: If you want the series between two numbers then you can use this function. This is good tool for FOR Loop. Its syntax is -

range(start, stop, step)

This gives the series from START to STOP-1 and the interval between two numbers of series will be STEP.

```
>>> list(range(0,5))  
[0, 1, 2, 3, 4]
```

```
for i in range(1,10):  
    print(i)
```

```
= RESTART:
```

```
1  
2  
3  
4  
5  
6  
7  
8  
9
```

Python Modules

- Module is a .py file which contains the definitions of functions and variables.
- Module is a simple python file.
- When we divide a program into modules then each module contains functions and variables. And each functions is made for a special task.
- Once written code in modules can be used in other programs.
- When we make such functions which may be used in other programs also then we write them in module.
- We can import those module in any program an we can use the functions.

Python Modules. . .

- Python provides two ways to import a module -
- **import statement:** to import full module.
- **from:** To import all or selected functions from the module.

```
import math  
a=20  
print(math.sqrt(a))
```

```
from math import sqrt  
a=20  
print(sqrt(a))
```

4.47213595499958

Python Modules. . .

- In last slide's example first the math.py file is searched.
- Then a space is created where all the variables and functions of the math module may be stored.
- Then statements of modules are executed.

```
import math  
a=20  
print (math.sqrt (a) )
```

4.47213595499958

math Module

- math module contains following functions–

- **ceil(x)** returns integer bigger than x or x integer.
- **floor(x)** returns integer smaller than x or x integer.
- **pow(x, n)** returns x^n .
- **sqrt(x)** returns square root of x.
- **log10(x)** returns logarithm of x with base-10
- **cos(x)** returns cosine of x in radians.
- **sin(x)** returns sine of x in radians.
- **tan(x)** returns tangent of x in radians.

```
>>> import math
>>> math.ceil(5.6)
6
>>> math.floor(-43.7)
-44
>>> math.floor(43.7)
43
>>> math.pow(3,4)
81.0
>>> math.sqrt(25)
5.0
>>> math.log10(100)
2.0
>>> math.cos(60)
-0.9524129804151563
>>> math.cos(0)
1.0
>>> math.sin(45)
0.8509035245341184
>>> math.sin(0)
0.0
>>> math.tan(45)
1.6197751905438615
>>> math.tan(90)
-1.995200412208242
```

help Function

- If you forgot that how a library function works then `help()` is very useful for this situation.
- If this function is used with any module then it provides the information and details of the given module and its all the contents.

```
>>> import math
>>> print(help(math.cos))
Help on built-in function cos in module math:

cos(...)
    cos(x)

    Return the cosine of x (measured in radians).

None
```

string Module

- We have already studied about string module in class XI. Here are some other functions of string module.
 - [String.capitalize\(\)](#) Converts first character to Capital Letter
 - [String.find\(\)](#) Returns the Lowest Index of Substring
 - [String.index\(\)](#) Returns Index of Substring
 - [String.isalnum\(\)](#) Checks Alphanumeric Character
 - [String.isalpha\(\)](#) Checks if All Characters are Alphabets
 - [String.isdigit\(\)](#) Checks Digit Characters
 - [String.islower\(\)](#) Checks if all Alphabets in a String, are Lowercase
 - [String.isupper\(\)](#) returns if all characters are uppercase characters
 - [String.join\(\)](#) Returns a Concatenated String
 - [String.lower\(\)](#) returns lowercased string
 - [String.upper\(\)](#) returns uppercased string
 - [len\(\)](#) Returns Length of an Object
 - [ord\(\)](#) returns Unicode code point for Unicode character
 - [reversed\(\)](#) returns reversed iterator of a sequence
 - [slice\(\)](#) creates a slice object specified by range()

random Module

- When we require such numbers which are not known earlier e.g. captcha or any type of serial numbers then in this situation we need random numbers. And here random module helps us. This contains following functions-

- randrange ()**: This method always returns any integer between given lower and upper limit to this method. default lower value is zero (0) and upper value is one(1).

```
>>> import random
>>> random.randrange(15)
0
>>> random.randrange(15)
10
```

```
import random
sub=["CS", "IP", "Phy", "Chem", "Maths"]
ran_index=random.randrange(3)
print(sub[ran_index])
```

```
= RESTART:
IP
>>> |
```

- random ()**: This generates floating value between 0 and 1. it does not require any argument.

```
>>> import random
>>> random.random()
0.9576497696403067
>>> random.random()
0.05005384317166639
```

random Module. . .

- **randint ()**: This method takes 2 parameters a,b in which first one is lower and second is upper limit. This may return any number between these two numbers including both limits. This method is very useful for guessing applications.

```
#Program to generate any two numbers between 0 to 10
```

```
import random
print(random.randint(0,10))
print(random.randint(0,10))
```

```
= RESTART:
8
4
```

- **uniform ()**: This method return any floating-point number between two given numbers.

```
>>> import random
>>> print("Uniform Lottery Number (1,100): ",random.uniform(1,100))
Uniform Lottery Number (1,100): 85.66278453560943
```

random Module. . .

- **choice ()**: this method is used for random selection from list, tuple or string.

```
#Program to select My food from a list
```

```
import random
lst=["Daal", "Chaaval", "Roti", "MixVeg", "Achar", "PaneerMasala", "malaiKofta"]
print("My First item is : ",random.choice(lst))
print("My Second item is : ",random.choice(lst))
```

```
= RESTART: C:/Users/KVBBKServer
My First item is : malaiKofta
My Second item is : MixVeg
```

- **shuffle ()**: this method can shuffle or swap the items of a given list.

```
#Program to shuffle My food list
```

```
import random
lst=["Daal", "Chaaval", "Roti", "MixVeg", "Achar", "PaneerMasala", "malaiKofta"]
random.shuffle(lst)
print("My shuffled list is : ",lst)
random.shuffle(lst)
print("My Second time shuffled list is : ",lst)
```

```
= RESTART: C:/Users/KVBBKServer/AppData/Local/Programs/Python/Python36/a.py =
My shuffled list is : ['Achar', 'Daal', 'Chaaval', 'malaiKofta', 'PaneerMasala', 'Roti', 'MixVeg']
My Second time shuffled list is : ['Chaaval', 'PaneerMasala', 'Daal', 'MixVeg', 'Roti', 'Achar', 'malaiKofta']
```


User-defined Functions

- These are the functions which are made by user as per the requirement of the user.
- Function is a kind of collection of statements which are written for a specific task.
- We can use them in any part of our program by calling them.
- *def* keyword is used to make user defined functions.

User-defined Functions. . .

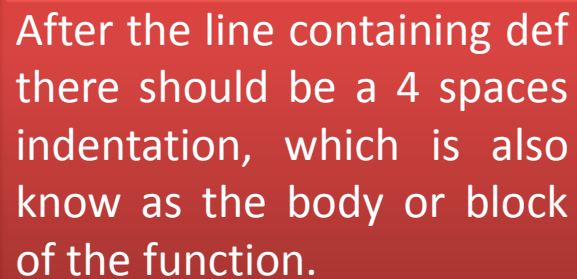
- We use following syntax with def keyword to prepare a user defined function.

```
def Function_Name(List_Of_Parameters):
```

```
    """docstring"""  
    statement(s)
```



Keyword



After the line containing def there should be a 4 spaces indentation, which is also know as the body or block of the function.



Function Definition

User-defined Functions **without argument** and **without return** (Practical)

```
#Program to add two numbers using function without argument
```

```
def add():  
    a=int(input("Enter value of a : " ))  
    b=int(input("Enter value of b : " ))  
    c=a+b  
    print("The sum is : ",c)
```

Function
Definition

```
add()
```

Function Call


```
= RESTART: C:/Users/KV  
Enter value of a : 12  
Enter value of b : 23  
The sum is : 35
```

User-defined Functions **with argument** and **without return** (Practical)

```
#Program to add two numbers using function with argument

def add(a,b):
    c=a+b
    print("The sum is : ",c)

x=int(input("Enter value of a : " ))
y=int(input("Enter value of b : " ))
add(x,y)
```



In the case of arguments, the values are passed in the function's parenthesis. And they are declared in definition as well.

```
= RESTART: C:/Users/KV
Enter value of a : 12
Enter value of b : 23
The sum is : 35
```

User-defined Functions **with argument** and **with return value** (Practical)

```
'''Program to add two numbers using function
with argument and return statement
'''
def add(a,b):
    c=a+b
    return c

x=int(input("Enter value of a : " ))
y=int(input("Enter value of b : " ))
z=add(x,y)
print("The sum is : ",z)
```

In the case of returning value, the calculated value is sent outside the function by a returning statement. This returned value will be hold by a variable used in calling statement.

```
= RESTART: C:/Users/KV
Enter value of a : 12
Enter value of b : 23
The sum is : 35
```

User-defined Functions with multiple return values (Practical)

```
'''Program which return multiple values'''  
  
def calc(a,b):  
    add=a+b  
    sub=a-b  
    mul=a*b  
    div=a/b  
    return add,sub,mul,div  
  
x=int(input("Enter value of a : "))  
y=int(input("Enter value of b : "))  
p,q,r,s=calc(x,y) ←  
print("The sum is : ",p)  
print("The difference is : ",q)  
print("The multiplication is : ",r)  
print("The quotient is : ",s)
```

In python a function may return multiple values. In multiple returning it returns a sequence of values to the calling statement. These returned values may be used as per the requirement.

```
= RESTART: C:\Users\KVBBKSer  
Enter value of a : 14  
Enter value of b : 7  
The sum is : 21  
The difference is : 7  
The multiplication is : 98  
The quotient is : 2.0
```

User-defined Functions with multiple return values (Practical)

```
'''Program which return multiple values'''
```

```
def calc(a,b):  
    add=a+b  
    sub=a-b  
    mul=a*b  
    div=a/b  
    return add,sub,mul,div
```

```
x=int(input("Enter value of x : "))  
y=int(input("Enter value of y : "))
```

```
result = calc(x,y)  
print("The answers are : ")  
for i in result:  
    print(i)
```

Last program may also be written as follows.

```
= RESTART: C:\Users\KV  
Enter value of x : 23  
Enter value of y : 4  
The answers are :  
27  
19  
92  
5.75
```

Result will be the tuple here.

Parameters and Arguments in Functions

- When we write header of any function then the one or more values given to its parenthesis () are known as *parameter*.
- These are the values which are used by the function for any specific task.
- While *argument* is the value passed at the time of calling a function.
- In other words the arguments are used to invoke a function.
- Formal parameters are said to be parameters and actual parameters are said to be arguments.

Parameters and Arguments in Functions . . .

```
'''Program which return multiple values'''
```

```
def calc(a,b):  
    add=a+b  
    sub=a-b  
    mul=a*b  
    div=a/b  
    return add,sub,mul,div
```

These are the *parameters* .

```
x=int(input("Enter value of x : "))  
y=int(input("Enter value of y : "))
```

```
result = calc(x,y)  
print("The answers are : ")  
for i in result:  
    print(i)
```

These are the *arguments* .

Types of Arguments

- Python supports 4 types of arguments-
 1. Positional Arguments
 2. Default Arguments
 3. Keyword Arguments
 4. Variable Length Arguments

```
'''Program to add two numbers using function  
with argument and return statement  
'''
```

```
def add(a,b):  
    c=a+b  
    return c
```

```
x=int(input("Enter value of a : " ))  
y=int(input("Enter value of b : " ))  
z=add(x,y)  
print("The sum is : ",z)
```

```
= RESTART: C:/Users/KV  
Enter value of a : 12  
Enter value of b : 23  
The sum is : 35
```

1. Positional Arguments

- These are the arguments which are passed in correct positional order in function.
- If we change the position of the arguments then the answer will be changed.

```
>>> def subtract(a,b):  
        print(a-b)  
  
>>> subtract(200,100)  
100  
>>> subtract(100,200)  
-100
```

2. Default Arguments

- These are the arguments through which we can provide default values to the function.
- If we don't pass any value to the function then it will take a pre defined value.

```
>>> def wish(name="Sanjeev") :  
        print("Hello ",name,"! Happy Holi!")  
  
>>> wish("Pavan")  
Hello Pavan ! Happy Holi!  
>>> wish()  
Hello Sanjeev ! Happy Holi!
```

This is point to remember that the default argument should be given after non default argument.

3. Keyword Arguments

- If a function have many arguments and we want to change the sequence of them then we have to use keyword arguments.
- Biggest benefit of keyword argument is that we need not to remember the position of the argument.
- For this whenever we pass the values to the function then we pass the values with the argument name. e.g.

```
>>> def wish(name,msg) :  
    print("Hello! ",name,"! ",msg,"!")  
  
>>> wish(name="Ram",msg="Good Evening")  
Hello! Ram ! Good Evening !  
>>> wish(msg="Good Evening",name="Ram")  
Hello! Ram ! Good Evening !  
>>> wish(name="Ram","Good Evening")  
SyntaxError: positional argument follows keyword argument  
>>> wish("Ram","Good Evening")  
Hello! Ram ! Good Evening !
```

If you used one argument with keyword then others must also be called with keywords otherwise an error will be raised.

4. Variable Length Arguments

- As we can assume by the name that we can pass any number of arguments according to the requirement. Such arguments are known as variable length arguments.
- We use (*) asterik to give Variable length argument.

```
>>> def sum(*n):
    s=0
    for i in n:
        s=s+i
    print("The sum = ",s)

>>> sum()
The sum = 0
>>> sum(10)
The sum = 10
>>> sum(10,20)
The sum = 30
>>> sum(10,20,30)
The sum = 60
```

You can notice here that every time the number of arguments are different and the answer is calculated for each number of arguments.

Passing ARRAYS /LISTS to Function

- In python we use list as array. Well we have to import numpy module to use array in python.
- We will pass here list to the function. As we know that list is better than array.

```
#Program to print the average of integers using list
```

```
def list_avg(lst):  
    l=len(lst)  
    sum=0  
    for i in lst:  
        sum+=i  
    return sum/l
```

```
a=input("Input Integers")  
a=a.split()  
for i in range(len(a)):  
    a[i]=int(a[i])  
  
avg=list_avg(a)  
  
print("Average is : ")  
print(round(avg,2))
```

```
RESTART: C:/Users/KVBBKServe  
Input Integers1 3 4 5 6 7 8 9  
Average is :  
5.38
```

Scope of Variable

- Scope of variable means the part of program where the variable will be visible. It means where we can use this variable.
- We can say that scope is the collection of variables and their values.
- Scope can of two types -
- Global (module)
 - All those names which are assigned at top level in module or directly assigned in interpreter.
- Local (function)
 - Those variables which are assigned within a loop of function.

Using main() as a Function

- Main function is not necessary in python.
- For the convenience of Non-python programmers, we can use it as follows-

```
def hello():  
    print("Hello! World!")  
  
def main():  
    print("This is the main function")  
    hello()  
  
main()
```

```
= RESTART: C:/Users/KVBBKSe  
This is the main function  
Hello! World!
```

Flow of Execution at the Time of Function Call

- The execution of any program starts from the very first line and this execution goes line by line.
- One statement is executed at a time.
- Function doesn't change the flow of any program.
- Statements of function doesn't start executing until it is called.
- When function is called then the control is jumped into the body of function.
- Then all the statements of the function gets executed from top to bottom one by one.
- And again the control returns to the place where it was called.
- And in this sequence the python program gets executed.

RECURSION

- In recursion, function calls itself until the condition is not satisfied.

```
# to calculate power of an inputed number using recursion

def power(x,n):
    if n==0:
        return 1
    else:
        return x*power(x,n-1)

a=int(input("Enter the value of x : "))
n=int(input("Enter the value of n : "))

print("The result is : ",power(a,n))
```

```
= RESTART: C:/Users/KVBB
Enter the value of x : 2
Enter the value of n : 3
The result is : 8
```

RECURSION. . .

- Recursion is one of the most powerful tools of the programming language. When a function calls itself within its body then this is known as recursion.
- There are two conditions to use recursion -
 - There must be a terminating condition.
 - There must be an if condition in recursive routine.
 - Every recursive problem has a special feature its reversal in the order of execution.
 - As many times there is recursive call then every time a new memory is allocated to the local variables of recursive function.
 - During recursion each value is pushed in a stack.

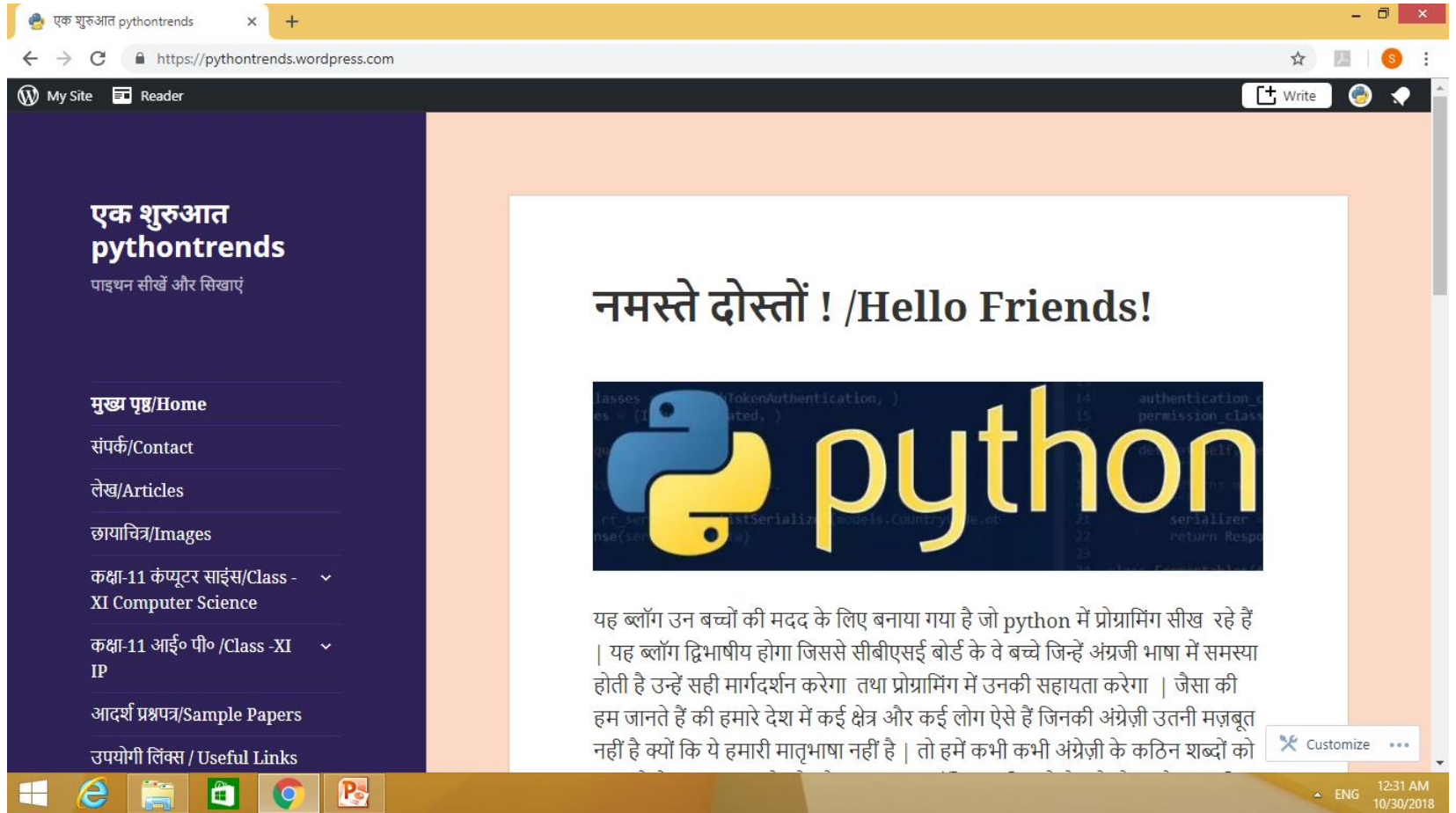
Drawbacks of RECURSION

- It uses more storage as each values is pushed in to stack.
- If recursive call is not checked carefully then your computer may go out of memory.
- It is less efficient in terms of time and speed.

धन्यवाद

और अधिक पाठ्य-सामग्री हेतु निम्न लिंक पर क्लिक करें -

www.pythontrends.wordpress.com




The screenshot shows a web browser displaying a WordPress site. The browser's address bar shows the URL <https://pythontrends.wordpress.com>. The site's header includes the title 'एक शुरुआत pythontrends' and the subtitle 'पाठ्यन सीखें और सिखाएं'. A navigation menu on the left lists: 'मुख्य पृष्ठ/Home', 'संपर्क/Contact', 'लेख/Articles', 'छायाचित्र/Images', 'कक्षा-11 कंप्यूटर साइंस/Class - XI Computer Science', 'कक्षा-11 आई० पी० /Class -XI IP', 'आदर्श प्रश्नपत्र/Sample Papers', and 'उपयोगी लिंक्स / Useful Links'. The main content area features a large heading 'नमस्ते दोस्तों ! /Hello Friends!' above a Python logo. Below the logo, there is a paragraph of text in Hindi. The browser's taskbar at the bottom shows the Windows logo, Edge, File Explorer, Word, Chrome, and PowerPoint icons. The system tray in the bottom right corner displays 'ENG', '12:31 AM', and '10/30/2018'.

एक शुरुआत
pythontrends
पाठ्यन सीखें और सिखाएं

मुख्य पृष्ठ/Home
संपर्क/Contact
लेख/Articles
छायाचित्र/Images
कक्षा-11 कंप्यूटर साइंस/Class - XI Computer Science
कक्षा-11 आई० पी० /Class -XI IP
आदर्श प्रश्नपत्र/Sample Papers
उपयोगी लिंक्स / Useful Links

नमस्ते दोस्तों ! /Hello Friends!



यह ब्लॉग उन बच्चों की मदद के लिए बनाया गया है जो python में प्रोग्रामिंग सीख रहे हैं | यह ब्लॉग द्विभाषीय होगा जिससे सीबीएसई बोर्ड के वे बच्चे जिन्हें अंग्रेजी भाषा में समस्या होती है उन्हें सही मार्गदर्शन करेगा तथा प्रोग्रामिंग में उनकी सहायता करेगा | जैसा की हम जानते हैं की हमारे देश में कई क्षेत्र और कई लोग ऐसे हैं जिनकी अंग्रेजी उतनी मज़बूत नहीं है क्यों कि ये हमारी मातृभाषा नहीं है | तो हमें कभी कभी अंग्रेजी के कठिन शब्दों को

Customize ...

ENG 12:31 AM 10/30/2018