What is Python?

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

It is used for:

- web development (server-side),
- software development,
- mathematics,
- system scripting.

What can Python do?

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.

Why Python?

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-oriented way or a functional way.

string

A **string** is any series of characters that are interpreted literally by a script. For example, "hello world" and "LKJH019283" are both examples of strings. Strings are used for storing text/characters. In most programming languages, the string can be either only letters or be letters and numbers (alphanumeric). Many languages also support a string as numbers only, but often is classified as an integer if it's only numbers.

The integer numbers (e.g. 2, 4, 20) have type int, and the ones with a fractional part (e.g. 5.0, 1.6) have type float.

Division (/) always returns a float. To do floor division and get an integer result you can use the // operator; to calculate the remainder you can use %:

Command:

```
>>> 17 / 3  # classic division returns a float
5.66666667
>>> 17 // 3  # floor division discards the fractional part and returns a int
5
>>> 17 % 3  # the % operator returns the remainder of the division
2
>>> 5 * 3 + 2  # floored quotient * divisor + remainder
17

>>> 5 ** 2  # 5 squared
25
>>> 2 ** 7  # 2 to the power of 7
128
```

Random number generation

Area of circle

$$A = \pi r^2$$

```
r=float(input("radius of circle:"))
A=3.14*r**2
print("Area of circle=", A)
```

Area of square

$$A = l \times b$$

```
l=float(input("length:"))
b=float(input("breadth:"))
A=l*b
print("Area of square=", A)
```

Volume of sphere

$$V = \frac{4}{3}\pi r^3$$

```
r=float(input("radius of circle:"))
V=(4/3)*3.14*r**3
print("Area of circle=", V)
```

Value of π

```
import math
print("Value of pi=", math.pi)
```

$res = x^y$

```
import math
x=int(input("Enter no:"))
y=int(input("Enter power:"))
res=math. pow(x,y)
print("Answer=", res)
```

$res = \sqrt{x}$

```
import math
x=int(input("Enter no:"))
res=math.sqrt(x)
print("Answer=", res)
```

$res = x!$

```
import math
x=int(input("Enter no:"))
res=math.factorial(x)
print("Answer=", res)
```

 # Bisection method
 # Solution of equation

```
def f(x):
    return x**3-5*x+1
def bisection(a,b,n):
    i=1
    condition=True
    while condition:
        x=(a+b)/2

        if f(a) * f(x) < 0:
            b=x
        else:
            a=x
        print("iteration=", i, "x=", x, "f(x)=", f(x))
        if i==n:
            condition=False
        else:
            condition=True
            i=i+1
        print("required root is", x)
```

```
a=float(input("first approximation root:"))
b=float(input("second approximation root:"))
n=float(input("number of iteration:"))
if f(a)*f(b)>0:
    print("given approximate root do not bracket the root.")
    print("try gain with different values.")
else:
    bisection(a,b,n)
# first approximation root:1
# second approximation root:0
# number of iteration: 5
# required root is 0.21875
```

For Trigonometric Eq.

```
import math
def f(x):
    return x*math.sin(x)-1
def bisection(a,b,n):
    i=1
    condition=True
    while condition:
        x=(a+b)/2

        if f(a) * f(x) < 0:
            b=x
        else:
            a=x
        print("iteration=", i, "x=", x, "f(x)=", f(x))
        if i==n:
            condition=False
        else:
            condition=True
            i=i+1
        print("required root is", x)

a=float(input("first approximation root:"))
b=float(input("second approximation root:"))
n=float(input("number of iteration:"))
if f(a)*f(b)>0:
    print("given approximate root do not bracket the root.")
    print("try gain with different values.")
else:
    bisection(a,b,n)
# first approximation root:1
# second approximation root:1.5
# number of iteration:7
```

```
# required root is 1.11328125
```

Or

```
import math
def f(x):
   return x-math.tan(x)
def bisection(a,b,n):
   i=1
   condition=True
   while condition:
      x=(a+b)/2

      if f(a) * f(x) < 0:
         b=x
      else:
         a=x
      print("iteration=", i, "x=", x, "f(x)=", f(x))
      if i==n:
         condition=False
      else:
         condition=True
         i=i+1
      print("required root is", x)

a=float(input("first approximation root:"))
b=float(input("second approximation root:"))
n=float(input("number of iteration:"))
if f(a)*f(b)>0:
   print("given approximate root do not bracket the root.")
   print("try gain with different values.")
else:
   bisection(a,b,n)
```

```
input:
first approximation root:-1
second approximation root:-2
number of iteration:5
```

```
Output:
iteration= 1 x= -1.5 f(x)= 12.601419947171719
required root is -1.5
iteration= 2 x= -1.75 f(x)= -7.27037992250933
required root is -1.75
iteration= 3 x= -1.625 f(x)= -20.05586276236962
required root is -1.625
iteration= 4 x= -1.5625 f(x)= 118.97000572254261
```

required root is -1.5625
iteration= 5 x= -1.59375 f(x)= -45.15211040673973
required root is -1.59375

Or (Simple form)

```
a=float(input("first approximation root:"))
b=float(input("second approximation root:"))
def f(x):
    return x**2-4
x1=(a+b)/2
if f(a)*f(b)<0:
    while abs(f(x1))>0.0001:
        if f(a)*f(x1)>0:
            a=x1
        else:
            b=x1
        x1=(a+b)/2
    print("required root is", x1)
else:
    print("the limits are not valid")
```

Prob:
Input: A **function of x**, for example $x^3 - x^2 + 2$.
And two values: **a** = -200 and **b** = 300 such that f(a)*f(b) < 0, i.e., f(a) and f(b) have opposite signs.
Output: The value of root is: -1.0025
OR any other value with allowed deviation from root.

Solution:

```
def f(x):
    return x*x*x - x*x + 2
a=float(input("first approximation root:"))
b=float(input("second approximation root:"))

def bisection(a,b):

    if (f(a) * f(b) >= 0):
        print("You have not assumed right a and b\n")
        return

    c = a
    while ((b-a) >= 0.01):
        # Find middle point
        c = (a+b)/2
         # Check if middle point is root
```

```
        if (f(c) == 0.0):
            break
         # Decide the side to repeat the steps
        if (f(c)*f(a) < 0):
            b = c
        else:
            a = c
    print("The value of root is : ","%.4f"%c)

# Driver code
# Initial values assumed
# a =-200
# b = 300
bisection(a, b)
```

Or

```
def f(x):
    return x**3-5*x-9
# Bisection method
def bisection(x0,x1,e):
    step = 1
    condition = True
    while condition:
        x2 = (x0 + x1)/2
        print("iteration=", step, "x1=", x1, "f(x1)=", f(x1))
        if f(x0) * f(x2) < 0:
            x1 = x2
        else:
            x0 = x2

        step = step + 1
        condition = abs(f(x2)) > e

    print('Required Root is : ', x2)

# input
x0 = float(input('First Guess: '))
x1 = float(input('Second Guess: '))
e = float(input('Tolerable Error: '))

# Checking Correctness of initial guess values and bisecting
if f(x0) * f(x1) > 0.0:
    print('Given guess values do not bracket the root.')
    print('Try Again with different guess values.')
else:
    bisection(x0,x1,e)
# First Guess: 2
# Second Guess: 3
```

Tolerable Error: 0.001
Required Root is : 2.855224609375

Newton Raphson

Prob: Using the Newton Rphson method solve the equation $x^3 - 2x - 5 = 0$ correct to three decimal places.

Program:

```
def f(x):
    return x**3-2*x-5

# derivative of f(x)
def g(x):
    return 3*x**2 - 5

def newtonRaphson(x0,e,N):
    step = 1
    flag = 1
    condition = True
    while condition:
        if g(x0) == 0.0:
            print('Divide by zero error!')
            break

        x1 = x0 - f(x0)/g(x0)
        print("iteration=", step, "x1=", x1, "f(x1)=", f(x1))
        x0 = x1
        step = step + 1

        if step > N:
            flag = 0
            break

        condition = abs(f(x1)) > e

    if flag==1:
        print("Required root is: ", x1)
    else:
        print("Not Convergent.")

x0 = float(input("Enter Guess: "))
e =float(input("Tolerable Error: "))
N =int(input("Maximum Step: "))
```

```
newtonRaphson(x0,e,N)

# Enter Guess: 2
# Tolerable Error: 0.001
# Maximum Step: 10
# Required root is:  2.094513521371825
```

Or

```
def f(x):
    return x-math.tan(x)

# derivative of f(x)
def g(x):
    return - math.tan(x)**2

def newtonRaphson(x0,e,N):
    step = 1
    flag = 1
    condition = True
    while condition:
        if g(x0) == 0.0:
            print('Divide by zero error!')
            break

        x1 = x0 - f(x0)/g(x0)
        print("iteration=", step, "x1=", x1, "f(x1)=", f(x1))
        x0 = x1
        step = step + 1

        if step > N:
            flag = 0
            break

        condition = abs(f(x1)) > e

    if flag==1:
        print("Required root is: ", x1)
    else:
        print("Not Convergent.")

x0 = float(input("Enter Guess: "))
e =float(input("Tolerable Error: "))
N =int(input("Maximum Step: "))

newtonRaphson(x0,e,N)
```

or

```
from sympy import*
x=Symbol('x')
f=x**3-2*x-5
f_prime=f.diff(x)
f=lambdify(x,f)
f_prime=lambdify(x, f_prime)
f_prime(2)
```

```
from sympy import*
x=Symbol('x')
f=x**3-2*x-5
f_prime=f.diff(x)
f=lambdify(x,f)
f_prime=lambdify(x, f_prime)

x=float(input("enter the value of x: "))
n=int (input("enter the required correct decimal place: "))
i=1
condition= True
while condition:
    g=str(x)
    x_n=x-(f(x)/f_prime(x))
    print("iteration=", i, "x=", x, "f(x)=", f(x))
    m=str(x_n)
    if m[0:n+2]==g[0:n+2]:
        condition=False
    else:
        condition=True
        x=x_n
        i=i+1
x=str(x)
s=x[0:n+2]
print("required root is ", s)
```

input:
enter the value of x: 2
enter the required correct decimal place: 3
output:
iteration= 1 x= 2.0 f(x)= -1.0
iteration= 2 x= 2.1 f(x)= 0.06100000000000083
iteration= 3 x= 2.094568121104185 f(x)= 0.0001857231732707021
required root is 2.094

Multiplication of list numbers

```
n=int(input("Enter limit:"))
m=1
for i in range(1,n+1):
    m=m*i
print("Answer=", m)
```

1. sum & average of list numbers

```
n=10
res=sum(range(1, n+1))
print("sum of first",n, "number is: ", res)
average=res/n
print("Average of", n, "number is: ", average)
```

Or

$$Sum = 1 + 2 + 3 + 4 + \cdots = \sum_{n=0}^{\infty} n$$

```
n=int (input ("Enter Limit"))
sum=0
for i in range (1, n+1):
    sum=sum+i
print ("sum = ", sum)
avg=sum/n
print ("average = ", avg)
```

$$Sum = \sum_{n=0}^{\infty} n^x$$

```
n=int(input("Enter Limit:"))
x=int(input("Enter power:"))
sum=0
for i in range(1,n+1,1):
    sum=sum+i**x
print ("sum = ", sum)
```

Or (Using sum() method)

```
L = [4, 5, 1, 2, 9, 7, 10, 8]
count = sum(L)
avg = count/len(L)
print("sum = ", count)
```

```
sum =  46
print("average = ", avg)
average =  5.75
```

Or (**Using sum() and statistics.mean()**)

```
import statistics
L = [4, 5, 1, 2, 9, 7, 10, 8]
sum1 = sum(L)
avg = statistics.mean(L)
print("sum = ", sum1)
sum =  46
print("average = ", avg)
average =  5.75
```

2. largest of a given list of numbers and its location in the list

```
list1 = [10, 20, 4, 45, 99]
list1.sort()
print("Largest element is:", list1[-1])
Largest element is: 99
```

Or

```
list1 = [10, 20, 4, 45, 99]
print("Largest element is:", max(list1))
Largest element is: 99
```

Or

```
heights = [100, 2, 300, 10, 11, 1000]
max_height = max(heights)
print("Largest element is:", max_height)
Largest element is: 1000
```

Or

```
heights = [100, 2, 300, 10, 11, 1000]
def my_max(x, y):
 if x < y:
  return y
 else:
   return x
max_height = reduce(my_max, heights)
print("Largest element is:", max_height)
Largest element is: 1000
```

3a. Sort a List of Numbers in Ascending Order

```
numbers = [1, 3, 4, 2]
print(numbers.sort())
None
print(numbers)
```

[1, 2, 3, 4]

Or **(Sort a List of Alphabets In Ascending Order)**

```
strs = ["geeks", "code", "ide", "practice"]
strs.sort()
print(strs)
```
['code', 'geeks', 'ide', 'practice']

3b. Sort a List of Numbers in the Descending Order

```
list1 = [1, 3, 4, 2]
list1.sort(reverse=True)
print(numbers)
```
[4, 3, 2, 1]

Or

```
list1 = [(1, 2), (3, 3), (1, 1)]
list1.sort()
print(list1)
```
[(1, 1), (1, 2), (3, 3)]
```
list1.sort(reverse=True)
print(list1)
```
[(3, 3), (1, 2), (1, 1)]

4. Binary Search

< (less than), > (greater than), == (equal to), <= (less than or equal to), >= (greater than or equal to) and != (not equal to).

```
def binary_search(item_list,item):
        first = 0
        last = len(item_list)-1
        found = False
        while( first<=last and not found):
                mid = (first + last)//2
                if item_list[mid] == item :
                        found = True
                else:
                        if item < item_list[mid]:
                                last = mid - 1
                        else:
                                first = mid + 1
        return found

print(binary_search([1,2,3,5,8], 6))
print(binary_search([1,2,3,5,8], 5))
```

Or

```
def BinarySearch(list1,key):
    low=0
    high=len(list1)-1
    Found=False
    while low<=high and not Found:
        mid=(low+high)//2
        if key==list1[mid]:
            Found=True
        elif key>list1[mid]:
            low=mid+1
        else:
            high=mid-1

    if Found==True:
        print("key is Found")
    else:
        print("key is not Found")


list1=[23,1,4,2,3]
list1.sort()
print(list1)
key=int(input("enter the key:"))
BinarySearch(list1,key)
```

5. Factorial of a Number Using Recursive approach

The "==" operator is known as the equality operator. The operator will return "true" if both the operands are equal.
However, it should not be confused with the "=" operator or the "is" operator. "=" works as an assignment operator. It assigns values to the variables.

e.g., Suppose, x=1

y=2

z=2

(x==y) will return false as both are assigned different values. (y==z) will return true as both are assigned the same values.

```
num=7
factorial=1
if num < 0:
   print("Sorry, factorial does not exist for negative numbers")
elif num == 0:
   print("The factorial of 0 is 1")
else:
   for i in range(1,num + 1):
      factorial = factorial*i
   print("The factorial of",num,"is",factorial)
```

The factorial of 7 is 5040

Or

```
n=int(input("Enter Limit:"))
f=1
for i in range(1,n+1):
   f=f*i
print ("factorial = ", f)
```

factorial = 720

Or

```
import math
n=7
print(math.factorial (n))
```

5040

6a. sum of a power series e.g. sine

The plus-equals operator "+=" provides a convenient way to add a value to an existing variable and assign the new value back to the same variable.

$$sinx = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1}$$

```
import math
def sin(x,n):
    sine=0
    for i in range(n):
        sign=(-1)**i
        pi=22/7
        y=x*(pi/180)
        sine+=  ((y**(2.0*i+1))/math.factorial(2*i+1))*sign  / sine=sine+
((y**(2.0*i+1))/ math.factorial(2*i+1))*sign
    return sine

x=30
n=10
print(round(sin(x,n),2))
```

0.5

Or

```
n=int(input("Enter Limit:"))
x=float(input("Enter Degree:"))
x=x*3.14/180
p=1
f=1
s=x
sine= -1
for i in range(3,n+1,2):
    x=x*sine
    p=pow(x,i)
    f=f*i*(i-1)
    s=s+(p/f)
print (round(s,2))
```

6b. sum of a cosine series

$$cosx = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n}$$

```
import math
def cosine(x,n):
    cosx=1
    sign= -1
    for i in range(2,n,2):
        pi=22/7
        y=x*(pi/180)
        cosx = cosx + (sign*(y**i))/math.factorial(i)
        sign = -sign
```

```
x=30
n=10
print(round(cosine(x,n),2))
```
0.87

Or

```
n=int(input("Enter Limit"))
x=float(input("Enter Degree"))
x=x*3.14/180
p=1
f=1
s=1
sine=1
for i in range(2,n+1,2):
    sine= sine*(-1)
    p=pow(x,i)
    f=f*i*(i-1)
    s=s+sine*(p/f)
print (s)
```

6c. sum of a exponential series

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots = \sum_{n=0}^{\infty} \frac{x^n}{n!}$$

```
def exponential(n, x):
    sum=1.0
    for i in range(n, 0, -1):
        sum = 1 + x * sum / i
    print ("e^x =", sum)
n=10
x=1.0
exponential(n, x)
```
e^x = 2.7182818011463845

Or

```
n=int(input("Enter Limit:"))
x=float(input("Enter Power of e:"))
p=1
f=1
s=1
for i in range(1,n+1):
    p=p*x
    f=f*i
    s=s+(p/f)
print (s)
```

Or

```
n=int(input("Enter Limit:"))
x=float(input("Enter Power of e:"))
p=1
f=1
s=1
for i in range(1,n+1):
    p=pow(x,i)
    f=f*i
    s=s+(p/f)
print (s)
```

Or

```
import math
x=int(input("Enter no:"))
e=math.exp(x)
print("Value of pi=", e)
```