Defination:

Python is highlevel, interactive, interpreted and object oriented programming language.

History:

Python is created by Guido Van Rossum and first released in 1991. python, like other languages has gone through a number of versions. python 0.9.0 was first released in 1991. In addition to exception handling, python includes classes, lists and strings. More importantly, it included lambda, map, filter and reduce, which aligned it heavily in relation to functional programming.

In 1994, python 1.0 was released. In 2000, python 2.0 was released. This version of python included list comprehen-sions, a full garbage collector, and it supported unicode. python 3.0 was the next version and was released in December of 2008

Difference between python 2.0 and 3.0

Python 2.0	python 3.0
* It was released in 2000	* It was released in 2008
* print statement is print "hello"	* print statement is print("hello")
* To store unicode string value you require to define them with "u".	* In python 3.0, default storing of strings is unicode.
* The syntax of python 2.0 was comparatively difficult to understand	* The syntax is simpler and easily understandable.
* In python 2.0, the xrange() is used for iterations	* The new Range() function introduced to perform iterations.

* It should be enclosed in notations

* It should be enclosed in parianthesis.

Features of python:

1. Easy to learn and use
2. Expressive Language
3. Interpreted Language
4. Cross-platform Language
5. Free and open source
6. object-oriented Language.
7. Extensible
8. Large standard Library
9. GUI programming support (Graphical user interfaces)
10. Integrated.
11. Case sensitive.

Applications of python:

1. Web Applications
2. E-commerce application
3. Scientifical applications
4. statical applications
5. Artificial Intelligence applications
6. Windows applications
7. IoT
8. Bussiness applications etc.

Identifiers:

A python identifier is a name used to identify a variable, function, class, module or other object.

An identifier starts with a letter A to Z or a to z or an underscore(_) followed by zero or more letters, underscores and digits (0 to 9)

The variable start with "_" in python is a private data member. variable is a reference of the memory location.

Operators:

To perform operations between two operands is known as operators.

1. Arithmetic operator
2. Relational operator
3. Logical operator
4. Assignment operator
5. Bitwise operator
6. Membership operator
7. Identity operator
8. Multiple assignment operator.

Arithmetic operator:

+ Addition of two operands
- subtraction
* Multiplication
/ division (Ex: 5/2 = 2.50000000)
%. remainder operator (Ex: 5%2 = 1)
// modulo division operator
it gives the coefficient value
Ex: 5//2 = 2
** Exponentiation operator
Exponentiation operator is right associative

Ex: ① 5**2 = 25
② 4**3 = 64

Ex:
>>> 5>3
True
>>> 5<3
False

Relational operator:

<, <=, >, >=, ==, !

Relational operators return the boolean values Either True (or) False.

Logical operator:

In python Logical operators are and, or, not We do not use the operator symbols like in C and C++.

Assignment operator:

Python does not support increment and decrement operators.

Assignment operators are

+=
-=
*=
/=
%=

Bitwise operator:

&, |, ^, ~, <<, >>

Examples

① >>> a=5 ⇒ 0101
>>> b=2 ⇒ 0010

5&2 (a&b)	5|2 (a|b)	5^2 (a^b)
00000101	00000101	00000101
00000010	00000010	00000010
00000000	00000111	00000111

~a = ~(00000101) = 11111010

a>>1 ⇒ 00000101 >>1
00000010

a<<1 ⇒ 00000101 <<1
00001010

Membership operator:

Membership operator returns either True or False, if a value/variable found in the list, its returns True otherwise it returns False.

Membership operators are "in" and "not in"

Identity operator:

Identity operators compare the memory location of two objects.

Identity operators are "is" and "is not".

Example

```
>>> a = 10        a
                 | 10 |
                 91911772

>>> b = 20        b
                 | 20 |
                 881177667

>>> id(a) ⇒ 91911772
>>> id(b) ⇒ 881177667
```

>>> id(a) gives the address of variable 'a'.

if id(a) is id(b)

Multiple assignment operator:

Assigns multiple values or the same value to multiple variables. In python, use the = operator to assign values to variables.

Ex:
```
>>> a = b = c = 10
>>> a,b,C = 10,20,30
>>> a,b,C = 10,"vasavi",20.30
```

String operations:

```
>>> a = "vasavi"
>>> a
```

It calls the memory the content will be printed as it is

```
⇒ "vasavi"
>>> print(a)
vasavi
```

→ print the string

Slicing:

slicing is a mechanism in python that is used to extracting a characters from the object.

In the above example if we want to print first 3 characters then we use the slicing operator.

Ex: a[0:3]

vas

a[1:3]

as

a[-2]

→ v

0 1 2 3 4 5

V a s a v i

-6 -5 -4 -3 -2 -1

Syntax:

object-name[start:end-1]

If we want to print the string three times than we use

>>> a *3

Vasavi, Vasavi, Vasavi

If we want to concatinate the string

a + 'Eng'

⇒ Vasavi Eng

Comments in python:

single comment #

Multiple comment """ """

""" is known as documentation.

String operations:

1. len()

len is used to find the length of the string

Ex: a = "Vasavi"

Syntax: len(object_name);

len(a)

⇒6

2. captalize()

captalize is used to charge first character into upper case.

Ex: a.captalize()

⇒ Vasavi

3. upper()

upper is used to change the entire string into captal letters.

Ex: a.upper()

VASAVI

4. split()

split is used to split the string into words

Ex: a = "Hello, world".

a.split(',')

⇒ "Hello", "world" ;

* To print the the list of keywords we have to import keyword package

There are two methods to import keyword.

1. import keyword

>>> print(keyword.kwargs)

The above statement print all the keywords in the python.

2. import keyword as k

Here k is alias name

>>>print(k.kwlist)

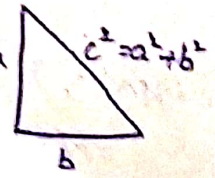
it also prints all the keywords by using alias names.

* iskeyword( )

iskeyword( ) return the boolean values either true or false

Expression:

Pythogorous $\Rightarrow c^2=a^2+b^2$



$c^2=a^2+b^2$

$c=\sqrt{a^2+b^2}$

To get the value of c we have to import math package

Program

```
import math
a,b =float(input("Enter a and b values"))
c = math.sqrt(a**2 + b**a)
print(c)
```

(or)

```
import math as m
a = 3
b = 4
c = m.sqrt(a*a+b*b)
print(c)
```

output format:

```
print("vasavi")
print("%.2f\n%.2f".format(a,b)) (or) print("%.2f\n%.2f"%(a,b))
print(f"{a}\n{b}")
```

Input format:

```
a=datatype(input("Enter a value"))
```

Adding two numbers:

```
a=10
b=20
c=a+b
print(c)
```

output

30

Program

Input employee details and calculate

1. hra = 20%.
2. da = 25%.
3. pf = 15%.
4. Net salary = (salary + hra + DA) - PF

Program

```
empid=int(input("enter employee id"))
empname = input("enter employee name)
salary = float(input("enter salary"))
hra = (salary * 20)/100 ⇒ salary * 0.2
DA = (salary *25)/100
Pf = (salary *15) /100
net = salary + hra + Da - Pf
Print("employee details")
```

```
print(empid)
print("\n" empname)
print("\n" salary)
print("\n" hra)
print("\n" da)
print("\n" pf)
print("\n" net)
```

swapping of two numbers using bitwise operator

Program

```
a = 20
b = 30
print(a)
print(b)
a = a^b
b = a^b
a = a^b
print("after swapping")
print(a)
print(b)
```

output

```
20
30
after swapping
30
20
```

5x1 Programs

①
```
a = 123
b = 'vasavi'
c = 5000.50
print(f"{a} {b} {c}".format(a,b,c))
```

Output

```
vasavi 5000.500000000 123
```

②
```
a = 3.41768
b = 44.3678
print("%.2f %.2f" %(a,b))
```

output

```
3.42
44.37
```

UNIT-2

Conditions

if: if is a conditional statement if the condition is true statements executed. The condition is executed boolean value (True (or) False)

Syntax:

```
if condition:
    statements
```

(indentation)

if else:

Syntax:

```
if condition:
    true statements
else:
    false statements
```

Nested if:

Syntax:

```
if condition:
    true statements
elif condition:
    fa True statements.
else:
    false statements
```

Example

①

```
a=10
b=5
if a>b:
    print('a is greater than b')
```

output:

a is greater than b

② Biggest of two numbers.

```
a=10
b=50
if condition a>b:
    print('a is greater than b')
else:
    print('b is greater than a')
```

output:

b is greater than a

③ Even or odd number

```
a = int(input("Enter a value"))
if a%2==0:
    print('a is even number')
else:
    print('a is odd')
```

output:

Enter a value 50

a is even number.

④ Biggest of three numbers.

```
a=int(input("Enter a value"))
b=int(input("Enter b value"))
c=int(input("Enter c value"))
if a>b and a>c:
    print('a is big')
elif b>c:
    print('b is big')
else:
    print('c is big')
```

output

enter a value 10
enter b value 20
enter c value 15
b is big

Shorthand Form of if statement:

This is single line statement without indentation

Syntax: ① if condition : statement

Ex: a = 10 , b = 20
if a == b : print("equal")

Form ②

syntax: True statement if condition else false statement

It is also known as conditional expression.

Ex: ① a = 50, b = 10
print('A') if a < b else print('B')

Conditional expression is also known as single line conditional statement.

Above syntax the if condition is true true statement is executed if it is false, false statement is executed

Example ②

a = 50
b = 50
print('A') if a > b else print("equal") if a == b else print('B')

Pass keyword:

pass is a keyword it is used to pass control to the interpreter control to the next statement.

Syntax:
```
if Condition:
    pass
elif Condition:
    statement
```

write a python program to find given number is even or odd using Bitwise operator.

```
n=int(input("Enter n value"))
if n&1==0
    print("Even")
else
    print("Odd")
```

write a python program to find given number is even or odd using conditional expression.

```
a = int(input("Enter value"))
print("Even") if a%2==0 else print("odd")
```

Write a python program to display the employee detail

5000 ← HRA - 15%, DA - 20%, PF - 10%

5001-10000 ← HRA - 20%, DA - 25%, PF - 15%

```
float HRA, DA, PF, NET,
a = input("Enter name")
b = int(input("Enter employee id"))
Basic = int(input("Enter basic salary"))
if basic<=5000:
    HRA = Basic * 0.15
    DA = Basic * 0.2
    PF = Basic * 0.1
```

```
elif Basic >5000 and Basic <=10000:
    HRA = Basic * 0.2
    DA = Basic * 0.25
    PF = Basic * 1.5
print("Employee details")
print(a)
print(b)
print(HRA) NET = Basic + HRA + DA - PF
print(NET) salary %.2f" % format(NET))
```

write a python program to print student id, name, m1, m2, m3, m4, m5, total, Grade.

below 35 — fail
35-49 — 3rd class
50-59 — 2nd class
>60 — 1st class

```
stud_id = input("Enter student id")
stud_name = input("Enter student name")
m1 = int(input("Enter m1"))
m2 = int(input("Enter m2"))
m3 = int(input("Enter m3"))
m4 = int(input("Enter m4"))
m5 = int(input("Enter m5"))
int tot = m1+m2+m3+m4+m5
float per = (tot/500) tot/5
print(stud_id)
print(stud_name)
print(m1)
print(m2)
print(m3)
print(m4)
print(m5)
print(total)
print(per)
```

```
if per<35 :
    print("fail")
elif per>=35 and per<=49:
    print("Third class")
elif per>=50 and per<=59:
    print("second class")
else:
    print("First class")
```

Loops:

While: while is a iterative statement that is used execute statements until condition false.

Syntax:

```
while condition :
    statements
    ---
else :
    statements
```

In the above syntax if the condition is true. True statements are executed repeatedly if condition is false else statement is executed.

Note: else statement is optional

Ex:

```
i=1
while ic=10:
    print(i)
    i=i+1
```

write a program to print event numbers between 1-20 using while loop.

```
i=1
while(i<=20):
    if i%2==0:
        print(i)
    i+=1
```

write a program to print the multiplication table

```
a=int(input("Enter a"))
i=1
while(i<=10):
    r=n*i
    print(a,'*',i,"=",r)
```

write a program to check the given number is prime or not

```
n=int(input("Enter n value"))
c=0
i=1
while(i<=n):
    if n%i==0:
        c+=1
if(c==2):
    prime('prime')
else:
    print('not prime')
```

Write a program to print reverse of a given number

```
n=int(input("Enter n value"))
rev=0
while(n>0):
    x=n%10
    n=n//10
    rev=rev*10+x
print(rev)
```

Write a program to print the biggest digit in the given number.

```
n=int(input('Enter n value'))
big=0
while(n>0):
    x=n%10
    n=n//10
    if x>big:
        big=x
print(big)
```

Write a program to find the factorial of a given number

```
n=int(input('Enter n value'))
i=1; fact=1
while(i<=n):
    fact=fact*i
    i+=1
print('Factorial is', fact)
```

write a program to print following pattern

```
i=1
while i<=4:
    j=1
    while j<=i:
        print('*', end=" ")
        j+=1
    print()
    i+=1
i=3
while i>=1:
    j=1
    while j<=i:
        print('*', end=" ")
        j+=1
    print()
    i-=1
```

write a python program to print n fibannoci series

```
n = int(input("enter n"))
a=0
b=1
i=1
while(i<=n):
    c=a+b
    a=b
    b=c
    i+=1
    print(b, end=" ")
```

a, b = b, a+b

range():

It is a built in function which returns sequence of numbers

Syntax: range(start, stop, step)

step - increment (or) decrement

default values of start and step are 0 and 1 respectively

It contains 3 parameters

1. start
2. stop
3. step

Here start is a optional by default zero. start means starting number and stop is required. That is traversing upto n-1 and step is optional by default "one" That is increment or decrement operations.

Example ① range(5)

o/p

0
1
2
3
4

② range(2,5)

o/p:

2
3
4

③ rang(5, 1, -2)

op

5
3

④ range(2,10,2)

op

2
4
6
8

for:

for is a iterative structure that is given to the sequence (such as string, list, tuple, dictionary and sets) of elements

Syntax: for val in sequence:

```
        statements
        ---
    else:
        statements
```

Example

①
```
for val in 'Vasavi':
    print(val)
```

o/p
```
V
a
S
a
v
i
```

②
```
for i in range(5):
    print(i)
```

o/p:
```
0
1
2
3
4
```

③
```
for i in range(5, 0, -1)
    print i
```

o/p
```
5
4
3
2
1
```

Reverse of a given string using for

```
a = 'vasavi'
l = len(a) - 1
for i in range(l, -1, -1):
    print(a[i])
```

o/p
i
v
a
s
a
v

Factorial of a given number

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

Multiplication table.

```
n = int(input("Enter n"))
for i in range(1,11):
    r = n * i
    print(n, "*", i, "=", r)
```

print

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

```
n = int(input("Enter n"))
for i in range(1, n+1)
    print('*', end = " ")
print()
```

```
for i in range(n-1, 0, -1)
    print('*', end = " ")
```

pair

```
n = int(input("Enter n"))
for i in range(1, n+1):
    for j in range(1, i+1):
        print('*', end = " ")
    print()
for i in range(n-1, 0, -1):
    for j in range(1, i+1):
        print('*', end = " ")
    print()
```

write a python program fabannocci series whose values do not exceed four million. find the sum of the even valued items.

```
f1 = 0; f2 = 1
sum = 0
while True:
    f1, f2 = f2, f1 + f2
    if f2 >= 4000000:
        break
    if f2 % 2 == 0:
        sum += f2
print(sum)
```

dp:

4613732

break and continue statement

break :

break statement in python is used to teaminate iterative state structure

Example

```
i=1
while i<=10:
    print(i)
    if i==5:
        break
    i+=1
```

o/p:

```
1
2
3
4
5
```

Continue:

Continue statement is a jump statement. It is jump depends on the condition and continue iterative structure.

Ex:

```
i=0
while i<10:
    i+=1
    if i==5 or i==6:
        continue
    print(i)
```

o/p

```
1
2
3
4
7
8
9
10
```

The given number is prime or not using for loop.

```
n = int(input("Enter n value"))
count = 0
for i in range(1, n+1):
    if n%i == 0:
        count += 1

if count == 2:
    print("prime number")

else:
    print("not prime number")
```

The given number is perfect or not

```
n = int(input("Enter n value"))
sum = 0
for i in range(1, n):
    if n%i == 0:
        sum += i

if sum == n:
    print('perfect number')

else:
    print('not perfect number')
```

The given number is armstrong or not

```
n = int(input("enter n value"))
t = n
sum = 0
for i in range(n, 0, -1):
    p = i%10
    i = i/10
    sum = sum + p*p*p
```

```
if sum==t:

    n= int(input("enter n value"))
    t=n
    sum=0
    for i in range(n,0,1*10):
            p=i%.10
            sum+=p*p*p
            i-=1
    if sum==t:
            print("Armstrong")
    else:
            print("not armstrong")
```

Functions:

A function is a self contained block of statements which is a reusable code.

Syntax: def fun-name(parameter1, parameter2 - - -):
    statements
    - - - -
    - - - -
    - - -
    return;

* In above syntax function name is defined along with parameters. Without parameters also define a function. Just enclose paranthesis.
* The statements are logical statements.
* return is optional.

Example ① def printmsg(s): ← Formal parameter
    print(s)
    return;
printmsg('welcome') ← Actual parameter

② def printmsg():
    print('Hello')
printmsg()

③ def sum(a,b):
    return a+b
print(sum(3,4))
print(sum(3.5,4.5))
print(sum('Hello','python'))

O/P:
7
8.0
Hello python

Types of formal parameters:

1. Required parameter
2. Keyword parameter
3. Default parameter
4. Variable-length parameter

Required parameters:

Ex:
```
def printmsg(a):
    print(s)
printmsg('welcomd)
printmsg( ) // causes error
```

Here the position argument is missing

keyword parameters:

```
def printmsg(s):
    print(s)
printmsg(s="welcome")
printmsg(p='python') # keyword error
```

Here the parameters which are used in functional declaration are same with positional arguments

default parameters:

Ex:
```
def sum(a=10, b=10):
    return a+b
print(sum( ))
print(sum(5))
```

o/p
```
20
15
```

```
def sum(a=10, b) // invalid statement.
```

variable-length parameters.

Ex: def printmsg(*valist):
    for i in valist:
        print(i)

o/p: 10
welcome
10.44

printmsg(10, 'welcome', 10.44)

Here * is used to take multiple parameters in the function call.

1. prime using function

```
def isprime(n):
    count=0;
    for i in range(1, n+1):
        if(n%i==0):
            count+=1

    if count==2:
        print('prime');
    else:
        print('not prime');
a=int(input("enter integer"))
isprime(a)
```

2. Find the prime digits in given numbers.

```
def primedigit(n):
    while(n>0):
        x=n%10
        count=0
        for i in range(1,x+1):
            if(x%i==0)
                count+=1
        if(count==2):
```

```
        print(x)
        n = n//10
a = int(input("Enter number "))
primedigit(a)
```

```
def isprime(n):
    c=0
    for i in range(1, n+1):
        if n%i==0:
            c+=1
    if c==2:
        return True
    else:
        return False
a = int(input("Enter a"))
while a>0:
    x = a%10
    isprime
    if isprime(x):
        print(x)
    a = a//10
```

Print the even digits in the given number.

```
def even(n):
    if n%2==0:
        return True
    else:
        return False
a = int(input("enter a"))
while a>0:
    x = a%10
    if even(x):
        print(x)
    a = a//10
```

Recursion:

Complex code is reduced to simple

A function call itself only is known as recursion

Advantage:

A complex code is converted to the small code.

Disadvantage:

It is occupied large memory.

Syntax:
```
def fun_name(arguments):
    statement
    condition
    fun_name(...)
```

Example
```
def fact(n):
    if n==1:
        return 1
    else:
        return n*fact(n-1)
n = int(input("enter n"))
print(fact(n))
```

Write a python program to generate fibinoci series using recursion

```
def fib(n):
    if n==0:
        return 0
    elif n==1:
        return 1
    else:
        return fib(n-1)+fib(n-2)

a=int(input("enter a"))
for i in range(a):
    print(fib(i))
```

Anonymous functions:

Anonymous function is a single line (inline function) It is also called lambda function

Syntax:

lambda arguments---: Expression

Example:

① `t=lambda x,y: x+y`
`print(t(3,4))`

O/P: 7

② `print((lambda x,y: x+y)(3,4))`

O/P: 7

③ `print((lambda x: x**2)(2))`

O/P 4

lambda function supporting following python functions.

* map()
* filter()
* reduce()

Local, global and non-local variables

Local variables:

Local variable is defined within the scope (or) block Those variables does not access outside world

Global

Global variable declared outside the block or scope. These variable are called global variables. But these global variables does not process in the local block.

The python interpreter raises error that is reference variable.

If we want to process the global variable into local using keyword "global"

```
x = 50 #global variable
def f():
    t=10 #local variable
    global x
    x = x+10
    print(x)
    print(t)
```

o/p:

60
10

write a program to calculate mean

```
def mean(*val):
    sum=0
    t=len(val)
    for i in val:
        sum+=i
    return sum/t
print(mean(10,20,30,40))
```

write a program to calculate mode

```
def mode(*val):
    t=len(val)
    for i in range(0,t):
        c=0; d=0
        for j in range(0,t):
            if val[i]=val[j]:
                c+=1
                if c>=d:
                    d=c
                    k=val[i]
    return k
print(mode(1,2,3,1,2,1,2,2))
```

Op: 2

Write a python program implement gcd function.

```
def gcd(a, b):
    r = a%b
    if(r==0):
        return b
    else:
        return gcd(b, r)
lcm = lambda x,y: x*y//gcd(x, y)
a = int(input("enter a"));
b = int(input("enter b"));
print(gcd(a,b));
print(lcm(a,b));
```

Nested function:

Syntax:
```
def f1():
    statements
    def f2():
        statements
```

Example
```
def myfun():
    def msg():
        print('welcome')
    msg()  // calling msg in myfun
myfun()
```

o/p: welcome

non-local variables:

Ex: ①
```
def f1():
    x='Global'
    def f2():
        x='local'
        print(x)
    print(x)
    return f2
f1()
```
o/p: Global

②
```
def f1():
    x='Global'
    def f2():
        nonlocal x
        x='Local'
        print('Inner function', x)
    f2()
    print('outer function:', x)
f1()
```
o/p:
Inner function Local
outer function Local

* Non-local is a variable that is defined in nested functions.
* Actually, the nonlocal is either converting into global or local.

Decorator:

Decorator is a simple function already exsited function is changed to new name is called decorator.

Ex: ①
```
def my-fun():
    print('vasavi')
msg = my-fun
my-fun()
msg()
```

o/p: vasavi
vasavi

②
```
def sum(x, y):
    return x+y
my-sum = sum
print(my-sum(2,5))
print(sum(2,5))
```

o/p
7
7