PYTHON NOTES

INTRODUCTION

What is Python?

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics.

It was created by Guido van Rossum, and released in 1991.

Why Python?

It is used for:

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

INPUT AND OUTPUT

Print Hello World.

```
>>> print("Hello World")
Hello World
```

How to declare a variable:

For Integer

```
>>> a=5
```

For Float

```
>>> b=5.5
```

For String

```
>>> c='Hello'  or  "Hello"
```

How to read user input and print it:

For Integer:

```
a=int(input("Enter The Integer="))
print(a)
```

Output –

```
Enter The Integer=10
```

```
10
```

For Float:

```
a=float(input("Enter The Float="))
print(a)
```

Output –

```
Enter The Float=5.5
5.5
```

For String:

```
a=input("Enter The String=")
print(a)
```

Output –

```
Enter The String=Hello
Hello
```

For Integer or Float:

```
a=eval(input("Enter The Number="))
print(a)
```

Output–

```
Enter The Number=5.5
5.5
```

Formatting output using String modulo operator(%):

```
a=20
b=3.14

print("Integer=%d,Float=%f"%(a,b))

#For printing float up to any decimal place just %.nf (Where n is place upto)
print("Round Of Float upto 2 decimal place=%.2f"%(b))
```

Output–

```
Integer=20,Float=3.140000

Round Of Float upto 2 decimal place=3.14
```

Formatting output using the format method:

```
a=240
b=3.14
c="Hello"

print("Integer={},Float={},String={}".format(a,b,c))


#Printing values of according to postion
#                                               0,1,2
print("String={2},Integer={0},Float={1}".format(a,b,c))
```

Output–

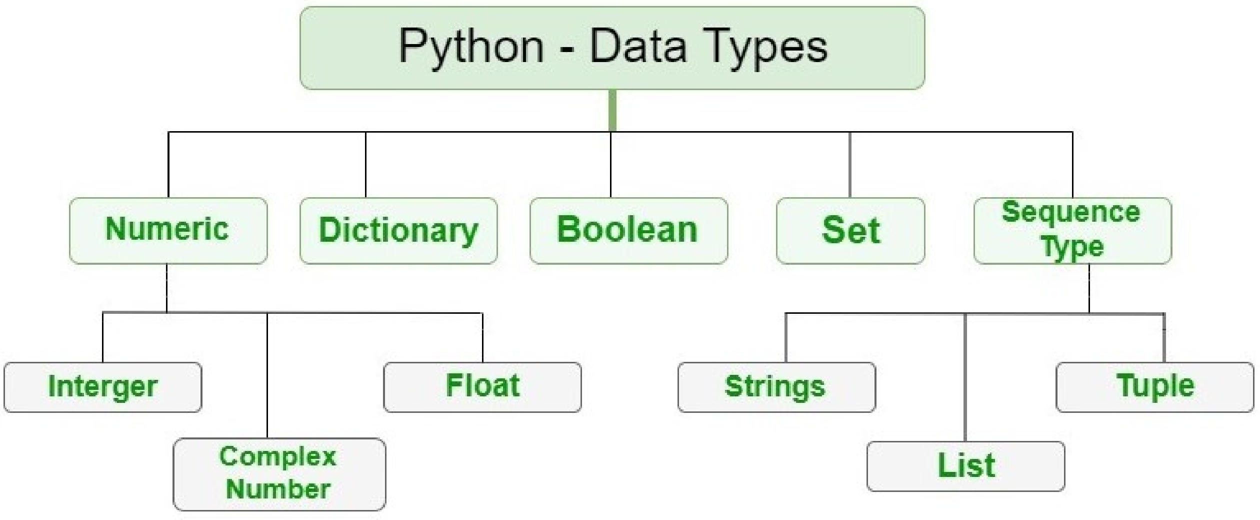
```
Integer=240,Float=3.14,String=Hello

String=Hello,Integer=240,Float=3.14
```

DATA TYPES

Data types are the classification or categorization of data items. It represents the kind of value that tells what operations can be performed on a particular data. Since everything is an object in Python programming, data types are actually classes and variables are instance (object) of these classes.

Python has five standard data types –

- Numeric (Integer, Float, Complex Number)
- Sequence Type (String, Lists, Tuple)
- Boolean
- Set
- Dictionary



Note – type() function is used to determine the type of data type.

Program to show the Type of variable.

```
a=20
b=3.14
c=3+4j
d="Hello"
e=[123,"Python",3.14]
f={1,1,2,3}
g=(1,2,3)
h={1:"Hello",2:"World"}

print("Type of a =",type(a))
print("Type of b =",type(b))
print("Type of c =",type(c))
print("Type of d =",type(d))
print("Type of e =",type(e))
print("Type of f =",type(f))
print("Type of g =",type(g))
print("Type of h =",type(h))
```

Output–

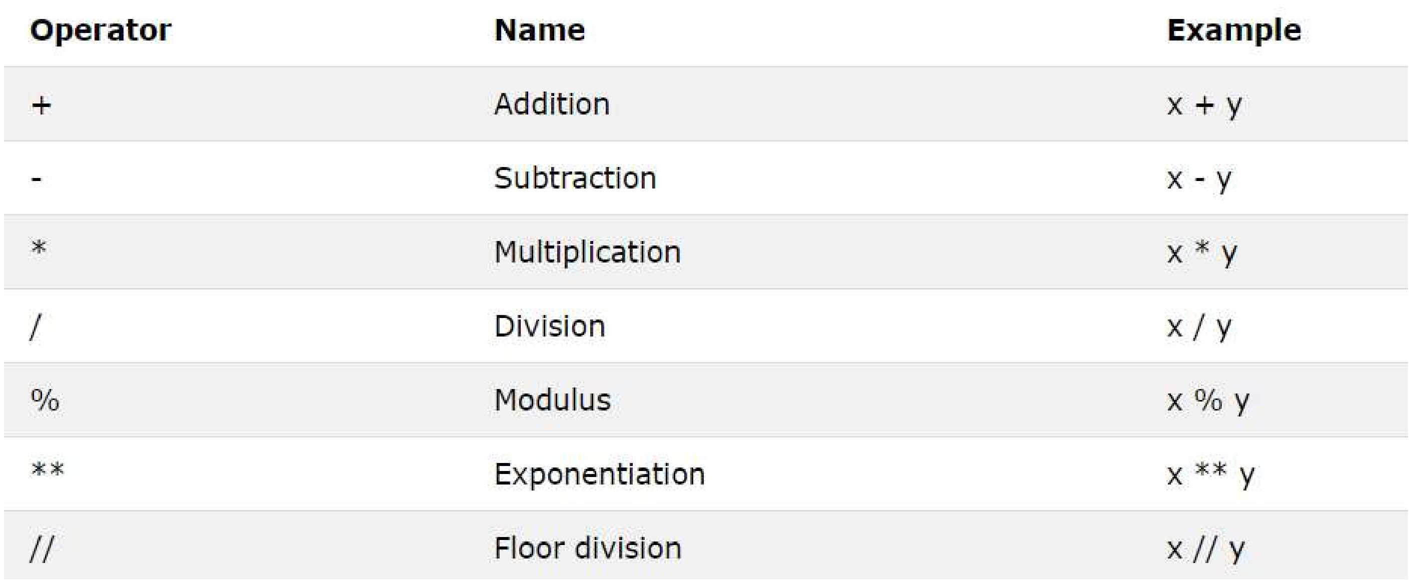
```
Type of a = <class 'int'>

Type of b = <class 'float'>

Type of c = <class 'complex'>

Type of d = <class 'str'>

Type of e = <class 'list'>

Type of f = <class 'set'>

Type of g = <class 'tuple'>

Type of h = <class 'dict'>
```

OPERATORS

Python divides the operators in the following groups:

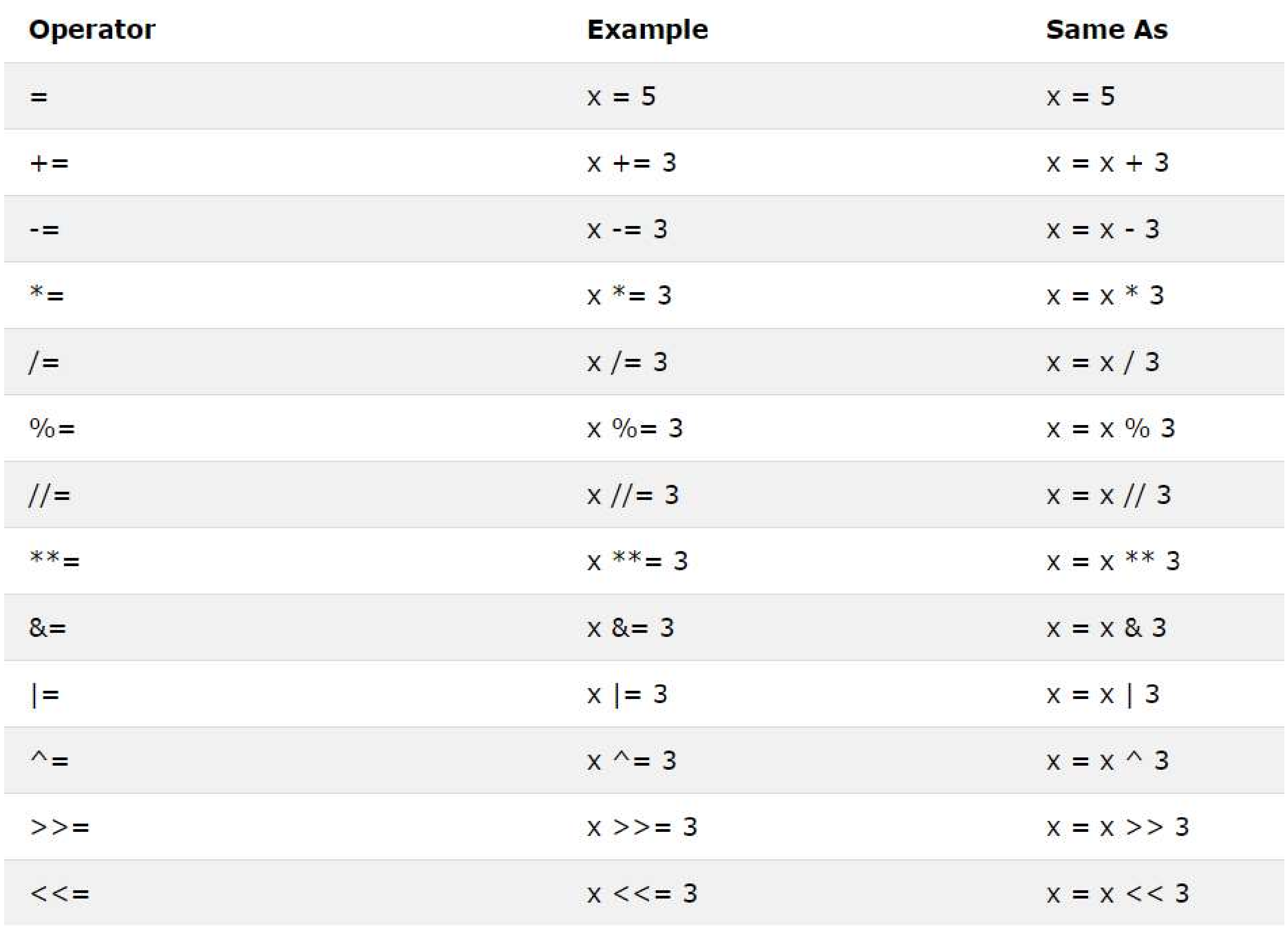
- Arithmetic operators
- Assignment operators
- Comparison operators
- Logical operators
- Identity operators
- Membership operators
- Bitwise operators

Python Arithmetic Operators



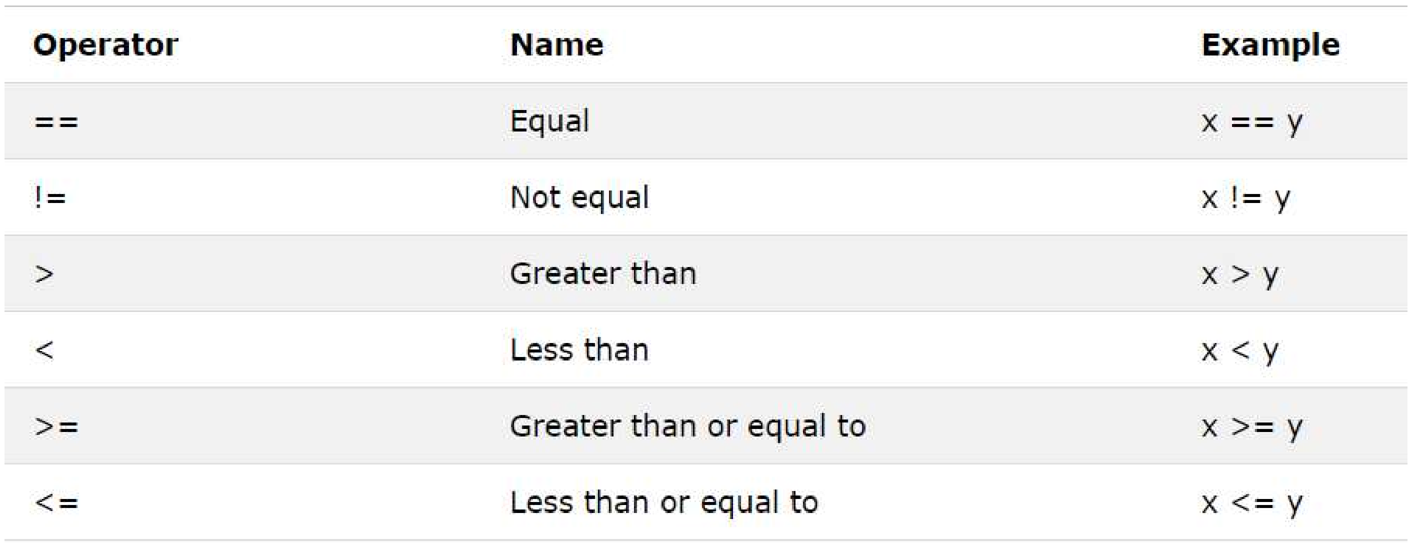
Operator	Name	Example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y

Python Assignment Operators



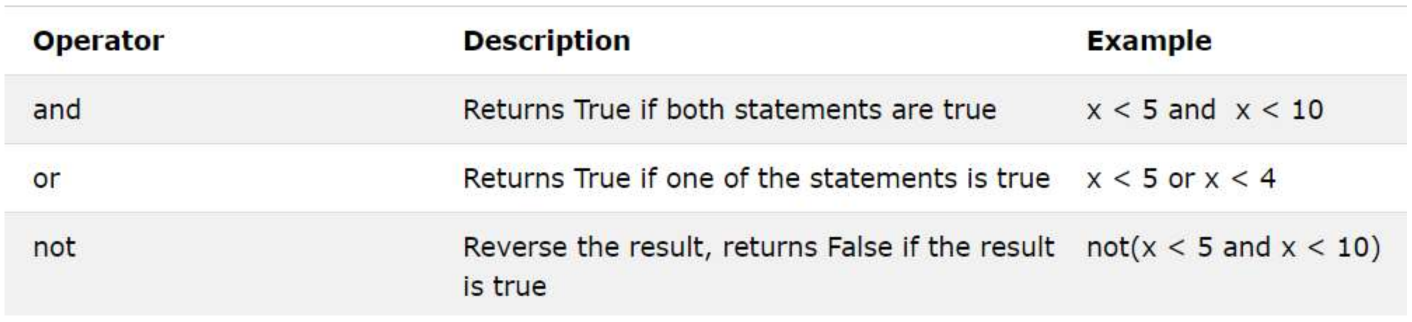
Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

Python Comparison Operators



Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Python Logical Operators



Operator	Description	Example
and	Returns True if both statements are true	x < 5 and x < 10
or	Returns True if one of the statements is true	x < 5 or x < 4
not	Reverse the result, returns False if the result is true	not(x < 5 and x < 10)

Python Identity Operators

Operator	Description	Example
is	Returns True if both variables are the same object	x is y
is not	Returns True if both variables are not the same object	x is not y

Python Membership Operators

Operator	Description	Example
in	Returns True if a sequence with the specified value is present in the object	x in y
not in	Returns True if a sequence with the specified value is not present in the object	x not in y

Python Bitwise Operators

Operator	Name	Description
&	AND	Sets each bit to 1 if both bits are 1
|	OR	Sets each bit to 1 if one of two bits is 1
^	XOR	Sets each bit to 1 if only one of two bits is 1
~	NOT	Inverts all the bits
<<	Zero fill left shift	Shift left by pushing zeros in from the right and let the leftmost bits fall off
>>	Signed right shift	Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off

Program to show the use of operators

```
# Arithmetic Operators
print("Arithmetic Operators")
print(5+4)
print(5-4)
print(5*4)
print(5/2)
print(5%2)
print(5//2)
print(2**3)


# Bitwise Operators
print("Bitwise Operators")
print(5|6)
print(5&6)
print(1^2)
print(~1)
print(13>>1)
print(13<<1)
```

Output–

```
Arithmetic Operators

9

1

20

2.5

1

2

8
```

Bitwise Operators

7

4

3

-2

6

26

IF-ELSE

Program to find Greatest of Two Numbers.

```
a=2
b=5
if(a>b):
    print("Greatest of Two Numbers = {}".format(a))
else:
    print("Greatest of Two Numbers = {}".format(b))
```

Program to find Greatest of Five Numbers.

```
a=2
b=5
c=3
d=7
e=10

if(a>b):
    m=a
else:
    m=b
if(c>m):
    m=c
if(d>m):
    m=d
if(e>m):
    m=e

print("Greatest of Five Numbers = {}".format(m))
```

Program to implement elif.

```
rupees=95
if(rupees>100):
    print("Greater than 100")
elif(rupees<100):
    print("Less than 100")
else:
    print("Equal to 100")
```

Program to check given year is a leap year or not.

```
year=int(input())
if(year%400==0 or year%100!=0 and year%4==0):
    print("Leap Year")
else:
    print("Not a Leap Year")
```

LOOPS

Loops are used to execute a set of statements repeatedly until the particular condition is satisfied

Types of Loop in Python-

1. **while loop**
2. **for loop**

While Loop

Print 1 to 10 using while loop.

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

Print 1 to 10 using while loop.

```
i=10
while(i>=1):
    print(i)
    i-=1
```

Important Programs Related to While Loop

1.Program to check given number is Palindrome or not.

```
n=int(input())
x=n
palindrome=0
while(n>0):
    palindrome=palindrome*10+n%10
    n=n//10
if(palindrome==x):
    print("Palindrome")
else:
    print("Not Palindrome")
```

2.Program to Reverse a given number.

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

3.Program to find the sum of digits of a number.

```
n=int(input())
s=0
while(n>0):
    s+=n%10
    n=n//10
print(s)
```

4.Program to chech given number is Armstrong or not.

```
n=int(input())
x=n
count=0
armstrong=0
while(n>0):
    count+=1
    n=n//10
n=x
while(n>0):
    armstrong+=(n%10)**count
    n=n//10
if(armstrong==x):
    print("Armstrong")
else:
    print("Not Armstrong")
```

For Loop

Print 1 to 10 using for loop.

```
for i in range(1,11):
    print(i)
```

Print 10 to 1 using for loop.

```
for i in range(10,0,-1):
    print(i)
```

Break - Break moves out of the loop and executes the next statement after the loop.

Program to show use of break.

```
for i in range(1,11):
    if(i==5):
        break
    print(i)
```

Output-

```
1

2

3

4
```

Continue-Continue skips the current executing loop and moves to the next loop.

Program to show use of continue.

```
for i in range(1,11):
    if(i>=5 and i<=8):
        continue
    print(i)
```

Output-

```
1

2

3

4

9

10
```

Important Programs Related to For Loop

1.Progran to find Factorial of a number.

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

2.Program for Fibonacci Series.

```
n=int(input())
f0=0
f1=1
if(n==0):
    print(f0)
elif(n==1):
    print(f0,f1)
else:
    print(f0,f1,end=" ")
    for i in range(2,n+1):
        f2=f0+f1
        f0=f1
        f1=f2
        print(f2,end=" ")
```

3.Program to find check given number is Prime or not.

```
n=int(input())
val=True
if(n==1):
    val=False
else:
    for i in range(2,n//2+1):
        if(n%i==0):
            val=False
            break
if(val==True):
    print("Prime Number")
else:
    print("Not Prime Number")
```

4.Program to check given number is Perfect number or not.

```
n=int(input())
s=0
for i in range(1,n//2+1):
    if(n%i==0):
        s+=i
if(s==n):
    print("Perfect Number")
else:
    print("Not Perfect Number")
```

5.Program to check given number is Peterson number or not.

```
#Strong number is also known a Peterson number
n=int(input())
x=n
s=0
while(n>0):
    r=n%10
    f=1
    for i in range(1,r+1):
        f*=i
    s+=f
    n=n//10
if(s==x):
    print("Strong Number")
else:
    print("Not Strong Number")
```

Important Pattern Programs.

1. Pattern 1

```
"""

*
**
***
****
*****

"""

n=int(input())

for i in range(1,n+1):
    for j in range(1,i+1):
        print("*",end="")
    print()
```

2. Pattern 2

```
"""
*****
****
***
**
*

"""

n=int(input())

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

3. Pattern 3

```
"""
    *
   **
  ***
 ****
*****

"""

n=int(input())

for i in range(1,n+1):
    for j in range(1,(n-i)+1):
        print(" ",end="")
    for k in range(1,i+1):
        print("*",end="")
    print()
```

4. Pattern 4

```
"""

*****
 ****
  ***
   **
    *


"""

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

5. Pattern 5

```
"""
    *
   ***
  *****
 *******
*********

"""

n=int(input())

for i in range(1,n+1):
    for j in range(1,(n-i)+1):
        print(" ",end="")
    for k in range(1,(2*i-1)+1):
        print("*",end="")
    print()
```

6. Pattern 6

```
"""
    *
   * *
  * * *
 * * * *
* * * * *

"""

n=int(input())

for i in range(1,n+1):
    for j in range(1,(n-i)+1):
        print(" ",end="")
    for k in range(1,i+1):
        print("* ",end="")
    print()
```

7. Pattern 7

```python
"""
A
AB
ABC
ABCD
ABCDE

"""
n=int(input())
for i in range(1,n+1):
    for j in range(1,i+1):
        # chr() converts ASCII values to characters
        print(chr(64+j),end="")
    print()
```

KEYWORDS AND BUILT-IN FUNCTIONS

Keywords

As of Python 3.8, there are thirty-five keywords in Python.

False	await	else	import	pass
None	break	except	in	raise
True	class	finally	is	return
and	continue	for	lambda	try
as	def	from	nonlocal	while
assert	del	global	not	with
async	elif	if	or	yield

Program to check keywords.

```
import keyword

# to check for keyword returns boolean
print(keyword.iskeyword("as"))

# to see list of keyword
print(keyword.kwlist)
```

Some Built-in Functions in Python

Function	Description
abs()	Returns the absolute value of a number
bin()	Returns the binary version of a number
oct()	Converts a number into an octal
hex()	Converts a number into a hexadecimal value
ord()	Returns the ASCII value of character
chr()	Returns the character of ASCII value
len()	Returns the length of an object
map()	Returns the specified iterator with the specified function applied to each item
max()	Returns the largest item in an iterable
min()	Returns the smallest item in an iterable
round()	Rounds a numbers
sorted()	Returns a sorted list
sum()	Sums the items of an iterator

Learn more about Python Built-in Functions-

https://www.w3schools.com/python/python_ref_functions.asp

Program to show the use of built-in functions.

abs()

```
>>> print(abs(-5))

5

>>> print(abs(5))

5
```

bin()

```
>>> print(bin(4))
0b100
```

oct()

```
>>> print(oct(12))
0o14
```

hex()

```
>>> print(oct(20))
0o24
```

ord()

```
>>> print(ord("a"))
97
```

chr()

```
>>> print(chr(97))
a
```

len()

```
>>> st="Python"
>>> print(len(st))
6
```

map()

```
>>> a,b,c=map(int,"1 2 3".split())
>>> print(a,b,c)
1 2 3
```

max()

```
>>> l=[6,5,8,1]
>>> print(max(l))
8
```

min()

```
>>> l=[6,5,8,1]
>>> print(min(l))
1
```

round()

```
>>> print(round(3.14))
3
>>> print(round(3.14,1))
3.1
```

sorted()

```
>>> st="cdba"
>>> print(sorted(st))
['a', 'b', 'c', 'd']
```

sum()

```
>>> l=[5,6,4]
>>> print(sum(l))
15
```

STRINGS

Strings in python are surrounded by either single quotation marks, or double quotation marks.

'hello' is the same as "hello".

Note - In python, the string data types are immutable.

Indexing

Zero indexing   # 0 1 2 3 4 5

Negative indexing #-6-5-4-3-2-1

"PYTHON"

Character at 0 is same as Character at -6 i,e P

……

Character at 5 is same as Character at -1 i,e N

Program for String Concatenation.

```
first_name="Tony"
last_name="Stark"
print(first_name+last_name)
```

Output-

```
TonyStark
```

Program for String Slicing.

```
>>> st="I love PYTHON"

>>> print(st[0])

I

>>> print(st[2])

l

>>> print(st[2:6])

love

>>> print(st[7:])

PYTHON

>>> print(st[:6])

I love

>>> print(st[::2])

Ilv YHN
```

Reversing String

```
>>> print(st[::-1])

NOHTYP evol I
```

Some String Methods in Python

Method	Description
capitalize()	Converts the first character to upper case
count()	Returns the number of times a specified value occurs in a string
index()	Searches the string for a specified value and returns the position of where it was found
join()	Joins the elements of an iterable to the end of the string
upper()	Converts a string into upper case
lower()	Converts a string into lower case
title()	Converts the first character of each word to upper case

Learn more about Python String Methods-

https://www.w3schools.com/python/python_ref_string.asp

Program to show the use of String Methods.

capitalize()

```
>>> st="i love python"

>>> st.capitalize()

'I love python'
```

count()

```
>>> txt = "I love apples, apple are my favorite fruit"

>>> txt.count("apple")

2
```

index()

```
>>> st="i love python"
>>> st.index("p")
7
```

join()

```
>>> l=["I","Love","Python"]
>>> " ".join(l)
'I Love Python'
```

upper()

```
>>> st="i love python"
>>> st.upper()
'I LOVE PYTHON'
```

lower()

```
>>> st="I LOVE PYTHON"
>>> st.lower()
'i love python'
```

title()

```
>>> st="i love python"
>>> st.title()
'I Love Python'
```

Program to Sort a String.

```
st="deabc"
st="".join(sorted(st))
print(st)
```

Important Programs related to Strings.

1. Program to count vowels and consonants in String.

```
st=input()
countvow=0
countcon=0
for i in st:
    if i in "AEIOUaeiou":
        countvow+=1
    else:
        countcon+=1
print("Vowels =",countvow)
print("Consonants =",countcon)
```

2.Program to reverse a sentence.

```
st=input()
rev=""
i=len(st)-1
while(i>=0):
    while(i>=0 and st[i]==" "):
        i-=1
    j=i
    if(i<0):
        break
    while(i>=0 and st[i]!=" "):
        i-=1
    if(rev==""):
        rev=rev+st[i+1:j+1]
    else:
        rev=rev+" "+st[i+1:j+1]
print("Reversed String =",rev)
```

3. Program for anagram of strings.

```
st1=input()
st2=input()

st1="".join(sorted(st1))
st2="".join(sorted(st2))

if(st1==st2):
    print("Anagram")
else:
    print("Not Anagram")
```

4.Program for anagram of strings (Efficient Algorithm Complexity=O(n)).

```
st1=input()
st2=input()

l=[0]*256
val=True
for i in st1:
    l[ord(i)]+=1

for i in st2:
    l[ord(i)]-=1

for i in l:
    if(i!=0):
        val=False
        break
if(val==True):
    print("Anagram")
else:
    print("Not Anagram")
```

LISTS

- Lists are used to store multiple items in a single variable.
- Lists can store different Data Types.
- Lists can store another list.
- Lists are Mutable.

Example-

```
>>> l=[1,5,"Python",[4,10]]
```

Indexing

Indexing of list is same as Strings.

Access List Items

```
>>> l=["apple","mango","orange"]

>>> print(l[0])

apple

>>> print(l[1])

mango

>>> print(l[2])

orange
```

Access Items of List that contains another List

```
Index          0                  1
         0           1      0          1
```

```
>>> l=[["Python","Java"],["HTML","CSS"]]
>>> print(l[0][0])
Python
>>> print(l[0][1])
Java
>>> print(l[1][0])
HTML
>>> print(l[1][1])
CSS
```

Change Item Value

```
>>> l=["apple","mango"]
>>> l[0]="orange"
>>> print(l)
['orange', 'mango']
```

Program for List concatenation

```
l1=[1,2,3,4]
l2=["Python","Java"]
l3=l1+l2
print(l3)
```

Output-

```
[1, 2, 3, 4, 'Python', 'Java']
```

List Comprehension

List comprehension offers a shorter syntax when you want to create a new list based on the values of an existing list.

Syntax

```
newlist = [expression for item in iterable if condition == True]
```

Program to take list from user.

```
l=[int(i) for i in input().split()]
print(l)
```

Program to take list from user(Alternate).

```
l=list(map(int,input().strip().split()))
print(l)

"""
For Taking value upto n

l=list(map(int,input().strip().split()))[:n]

"""
```

Some List Methods in Python

Method	Description
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the first item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

Learn more about Python Lists Methods-

https://www.w3schools.com/python/python_ref_list.asp

Program to show the use of Lists Methods.

append()

```
>>> l=["Python","Java"]

>>> l.append("Kotlin")

>>> print(l)

['Python', 'Java', 'Kotlin']
```

clear()

```
>>> l=["Python","Java"]
>>> l.clear()
>>> print(l)
[]
```

count()

```
>>> l=[1,2,1,4,6,4,4,5,6]
>>> print(l.count(4))
3
```

extend()

```
>>> l1=[1,2,3,4]
>>> t1=(9,10)
>>> l1.extend(t1)
>>> print(l1)
[1, 2, 3, 4, 9, 10]
```

index()

```
>>> l=["Python","Java","Kotlin"]
>>> print(l.index("Java"))

1
```

insert()

```
>>> l=["Python","Java","Kotlin"]

>>> l.insert(2,"Javascript")

>>> print(l)

['Python', 'Java', 'Javascript', 'Kotlin']
```

pop()

```
>>> l=["Python","Java","Kotlin"]

>>> print(l.pop())

Kotlin
```

remove()

```
>>> l=["Python","Java","Kotlin"]

>>> l.remove("Java")

>>> print(l)

['Python', 'Kotlin']
```

reverse()

```
>>> l=["Python","Java","Kotlin"]
>>> l.reverse()
>>> print(l)
['Kotlin', 'Java', 'Python']
```

sort()

For Increasing Order

```
>>> l=[2,5,4,3,6,10,9]
>>> l.sort()
>>> print(l)
[2, 3, 4, 5, 6, 9, 10]
```

For Decreasing Order

```
>>> l=[2,5,4,3,6,10,9]
>>> l.sort(reverse=True)
>>> print(l)
[10, 9, 6, 5, 4, 3, 2]
```

Making 3*3 Matrix

```
>>> l=[[0]*3]*3
>>> print(l)
[[0, 0, 0], [0, 0, 0], [0, 0, 0]]
```

Program to take n*n matrix from user.

```python
l=[]
n=int(input("Enter order of matrix = "))
for i in range(n):
    l1=list(map(int,input().strip().split()))[:n]
    l.append(l1)
print(l)
```

Output-

```
Enter order of matrix = 3
1 2 3
4 5 6
7 8 9
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

Important Programs related to Lists or Arrays.

1.Find the Smallest and largest element in an array.

```
arr=list(map(int,input().strip().split()))
minimum=arr[0]
maximum=arr[0]
for i in arr:
    if(minimum>i):
        minimum=i
    if(maximum<i):
        maximum=i
print("Smallest =",minimum)
print("Largest =",maximum)
```

2. Calculate the sum of elements in an array.

```
l=list(map(int,input().strip().split()))
s=0
for i in l:
    s+=i
print(s)
```

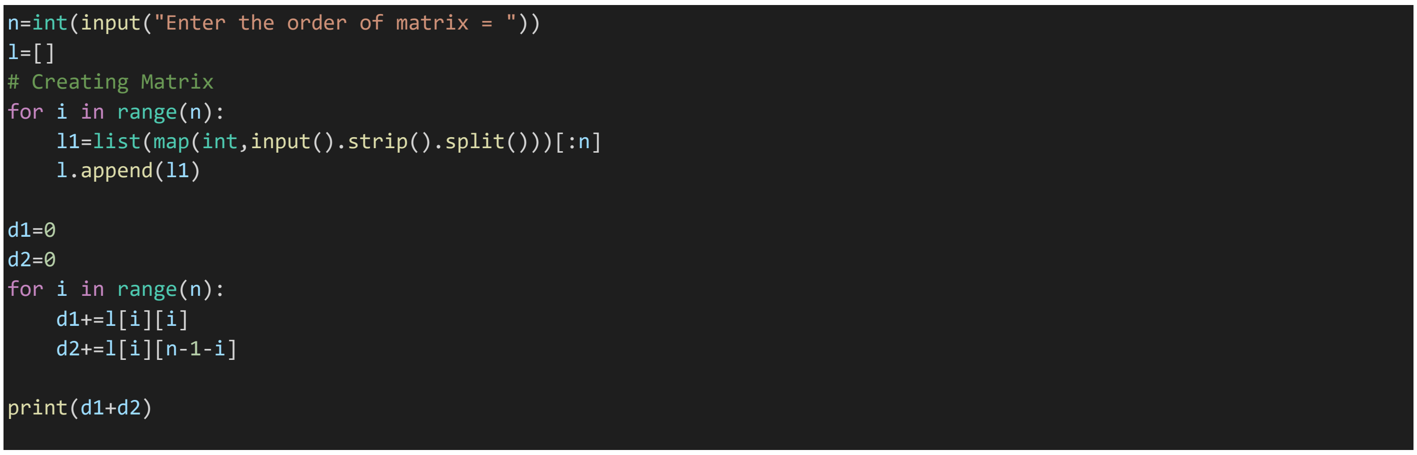
3. Reverse an Array(Efficient Algorithm)

```
l=list(map(int,input().strip().split()))

for i in range(len(l)//2):
    l[i],l[len(l)-1-i]=l[len(l)-1-i],l[i]

print(l)
```

4. Array Rotation upto k th Position (Efficient Algorithm).

```
l=list(map(int,input().strip().split()))
l1=[0]*len(l)
k=int(input())
for i in range(len(l)):
    l1[(i+k)%len(l)]=l[i]
print(l1)
```

5. Program of find diagonal sum in matrix.



```
n=int(input("Enter the order of matrix = "))
l=[]
# Creating Matrix
for i in range(n):
    l1=list(map(int,input().strip().split()))[:n]
    l.append(l1)

d1=0
d2=0
for i in range(n):
    d1+=l[i][i]
    d2+=l[i][n-1-i]

print(d1+d2)
```

Pass by value and Pass by Reference in Lists.

Pass by Value: The method parameter values are copied to another variable and then the copied object is passed, that's why it's called Pass by Value.

Pass by Reference: An alias or reference to the actual parameter is passed to the method, that's why it's called Pass by Reference.

Pass by Reference in list.

```
>>> l1=["Python","Java","Javascript"]

>>> l2=l1

>>> l2[1]="Kotlin"

>>> print(l1)
```

```
['Python', 'Kotlin', 'Javascript']
```

Pass by Value in list.

```
>>> l1=["Python","Java","Javascript"]
>>> l2=list(l1)
>>> l2[1]="Kotlin"
>>> print(l1)
['Python', 'Java', 'Javascript']
>>> print(l2)
['Python', 'Kotlin', 'Javascript']
```

TUPLES

- In someways a tuple is similar to a list in terms of indexing, nested objects and repetition.
- A tuple is immutable.

Example-

```
t= ("apple", "banana", "cherry")
                    or
t= "apple", "banana", "cherry"
```

Tuple Methods in Python

Method	Description
count()	Returns the number of times a specified value occurs in a tuple
index()	Searches the tuple for a specified value and returns the position of where it was found

Program to show the use of Tuples Methods.

count()

```
>>> t=(2,3,1,2,3,4,2,2)
>>> t.count(2)
4
```

index()

```
>>> t=(2,3,1,2,3,4,2,2)
>>> t.index(3)
1
```

SET

- A set is well-defined collection of distinct elements.
- A set is a collection which is both *unordered* and *unindexed*.
- Sets are written with curly brackets.

Example-

s= {"apple", "banana", "cherry"}

Some Sets Methods in Python

Method	Description
add()	Adds an element to the set
clear()	Removes all the elements from the set
difference()	Returns a set containing the difference between two or more sets
intersection()	Returns a set, that is the intersection of two other sets
isdisjoint()	Returns whether two sets have a intersection or not
issubset()	Returns whether another set contains this set or not
issuperset()	Returns whether this set contains another set or not
pop()	Removes an element from the set
remove()	Removes the specified element
symmetric_difference()	Return a set that contains all items from both sets, except intersection
union()	Return a set containing the union of sets

Learn more about Python Set Methods-

https://www.w3schools.com/python/python_ref_set.asp

Program to show the use of Set Methods.

add()

```
>>> s={1,2,3}

>>> s.add(4)

>>> print(s)

{1, 2, 3, 4}
```

clear()

```
>>> s={1,2,3}

>>> s.clear()

>>> print(s)

set()
```

difference()

```
>>> s={1,2,3}

>>> s1={1,2,3}

>>> s2={3,4,5}

>>> print(s1.difference(s2))

{1, 2}
```

intersection()

```
>>> s1={1,2,3}
>>> s2={3,4,5}
>>> s1.intersection(s2)
{3}
```

isdisjoint()

```
>>> s1={1,2,3}
>>> s2={3,4,5}
>>> s1.isdisjoint(s2)
False
```

issubset()

```
>>> s1={1,2,3,4,5,6,7,8,9}
>>> s2={3,4,5}
>>> s2.issubset(s1)
True
```

issuperset()

```
>>> s1={1,2,3,4,5,6,7,8,9}
>>> s2={3,4,5}
>>> s1.issuperset(s2)
True
```

pop()

```
>>> s1={1,2,3}
>>> s1.pop()
1
```

remove()

```
>>> s1={1,2,3}
>>> s1.remove(2)
>>> print(s1)
{1, 3}
```

symmetric_difference()

```
>>> s1={1,2,3}
>>> s2={3,4,5}
>>> s1.symmetric_difference(s2)
```

```
{1, 2, 4, 5}
```

union()

```
>>> s1={1,2,3}
>>> s2={3,4,5}
>>> s1.union(s2)
{1, 2, 3, 4, 5}
```

DICTIONARIES

- Dictionaries are used to store data values in key:value pairs.
- A dictionary is a collection which is ordered*, changeable and does not allow duplicates.

Example-

```
word={1:"One",2:"Two",3:"Three"}
```

Accessing Values with Keys

```
>>> word={1:"One",2:"Two",3:"Three"}

>>> word[2]

'Two'
```

If Dictionary has two same keys with different values last value is considered.

```
>>> word={1:"One",2:"Two",3:"Three",1:"Four"}

>>> word

{1: 'Four', 2: 'Two', 3: 'Three'}
```

Adding Items

Adding an item to the dictionary is done by using a new index key and assigning a value to it

```
>>> word={1:"One",2:"Two",3:"Three"}

>>> word[4]="Four"

>>> print(word)

{1: 'One', 2: 'Two', 3: 'Three', 4: 'Four'}
```

Some Dictionary Methods in Python

Method	Description
clear()	Removes all the elements from the dictionary
fromkeys()	Returns a dictionary with the specified keys and value
items()	Returns a list containing a tuple for each key value pair
keys()	Returns a list containing the dictionary's keys
pop()	Removes the element with the specified key
popitem()	Removes the last inserted key-value pair
setdefault()	Returns the value of the specified key. If the key does not exist: insert the key, with the specified value
values()	Returns a list of all the values in the dictionary

Program to show the use of Dictionary Methods.

clear()

```
>>> word={1:"One",2:"Two",3:"Three"}

>>> word.clear()

>>> print(word)
```

```
{}
```

fromkeys()

```
>>> k=("Key1","Key2","Key3")
>>> val=0
>>> res=dict.fromkeys(k,val)
>>> print(res)
{'Key1': 0, 'Key2': 0, 'Key3': 0}
```

items()

```
>>> word={1:"One",2:"Two",3:"Three"}
>>> print(word.items())
dict_items([(1, 'One'), (2, 'Two'), (3, 'Three')])
```

keys()

```
>>> word={1:"One",2:"Two",3:"Three"}
>>> print(word.keys())
dict_keys([1, 2, 3])
```

pop()

```
>>> word={1:"One",2:"Two",3:"Three"}

>>> word.pop(2)

'Two'
```

popitem()

```
>>> word={1:"One",2:"Two",3:"Three"}

>>> word.popitem()

(3, 'Three')
```

setdefault()

If key is present

```
>>> word={1:"One",2:"Two",3:"Three"}

>>> word.setdefault(1,"Four")

'One'
```

If key is not present

```
>>> word.setdefault(4,"Four")

'Four'
```

values()

```
>>> word={1:"One",2:"Two",3:"Three"}
>>> word.values()

dict_values(['One', 'Two', 'Three'])
```

Important Programs related to Lists or Arrays.

1. Find the Frequency of elements in array.

```
def frequency(arr):
    freq={}
    for i in arr:
        if (i in freq):
            freq[i]+=1
        else:
            freq[i]=1
    return freq

l=list(map(int,input().strip().split()))

print(frequency(l))
```

TYPE CONVERSION

OR

TYPE CASTING

Type Casting – Type casting is when you assign a value of oneprimitive data type to another type.

There are two types of Type Conversion in Python:

1. **Implicit Type Conversion –** Automatic Type Conversion
2. **Explicit Type Conversion –** Manual Type Conversion

Example of Implicit Type conversion-

```
>>> n1=2

>>> n2=2.5

>>> n1+=n2

>>> print(n1)

4.5

>>> print(type(n1))

<class 'float'>
```

Example of Explicit Type Conversion-

From int() to

1.float()

```
>>> n=2

>>> n=float(n)

>>> print(n)

2.0

>>> print(type(n))

<class 'float'>
```

2.complex()

```
>>> n=2

>>> n=complex(n)

>>> print(n)

(2+0j)

>>> print(type(n))

<class 'complex'>
```

3.str()

```
>>> n=2

>>> n=str(n)

>>> print(n)

2

>>> print(type(n))

<class 'str'>
```

From float() to

1.int()

```
>>> n=2.5

>>> n=int(n)

>>> print(n)

2

>>> print(type(n))

<class 'int'>
```

2.complex()

```
>>> n=2.5

>>> n=complex(n)

>>> print(n)
```

```
(2.5+0j)
>>> print(type(n))
<class 'complex'>
```

3.str()

```
>>> n=2.5
>>> n=str(n)
>>> print(n)
2.5
>>> print(type(n))
<class 'str'>
```

Note: You cannot convert complex numbers into another number type.

From str() to

Note-for int(), float() or complex() string should be a number.

1.int()

```
>>> s="1"
>>> s=int(s)
>>> print(s)
```

```
1
>>> print(type(s))
<class 'int'>
```

2.float()

```
>>> s="1.5"
>>> s=float(s)
>>> print(s)
1.5
>>> print(type(s))
<class 'float'>
```

3.complex()

```
>>> s="1"
>>> s=complex(s)
>>> print(s)
(1+0j)
>>> print(type(s))
<class 'complex'>
```

4.list()

```
>>> s="Python"

>>> s=list(s)

>>> print(s)

['P', 'y', 't', 'h', 'o', 'n']

>>> print(type(s))

<class 'list'>
```

5.tuple()

```
>>> s="Python"

>>> s=tuple(s)

>>> print(s)

('P', 'y', 't', 'h', 'o', 'n')

>>> print(type(s))

<class 'tuple'>
```

6.set()

```
>>> s="apple"

>>> s=set(s)

>>> print(s)
```

```
{'p', 'e', 'l', 'a'}
>>> print(type(s))
<class 'set'>
```

From list() to

1.tuple()

```
>>> l=["Python","Java","Javascript"]
>>> l=tuple(l)
>>> print(l)
('Python', 'Java', 'Javascript')
>>> print(type(l))
<class 'tuple'>
```

2.set()

```
>>> l=["Python","Java","Javascript"]
>>> l=set(l)
>>> print(l)
{'Java', 'Javascript', 'Python'}
>>> print(type(l))
<class 'set'>
```

From tuple() to

1.list()

```
>>> t=("Python","Java","Javascript")
>>> t=list(t)
>>> print(t)
['Python', 'Java', 'Javascript']
>>> print(type(t))
<class 'list'>
```

2.set()

```
>>> t=("Python","Java","Javascript")
>>> t=set(t)
>>> print(t)
{'Java', 'Javascript', 'Python'}
>>> print(type(t))
<class 'set'>
```

From set() to

1.list()

```
>>> s={1,2,2,3}

>>> s=list(s)

>>> print(s)

[1, 2, 3]

>>> print(type(s))

<class 'list'>
```

2.tuple()

```
>>> s={1,2,2,3}

>>> s=tuple(s)

>>> print(s)

(1, 2, 3)

>>> print(type(s))

<class 'tuple'>
```

USER DEFINED FUNCTION

- A function is a block of code which only runs when it is called.
- You can pass data, known as parameters, into a function.

Creating a Function

In Python a function is defined using the def keyword:

Example

```
def my_function():
  print("Hello from a function")

my_function()
```

Return Values

To let a function return a value, use the return statement:

Example

```
def my_function(x):
  return 5 * x

print(my_function(3))
print(my_function(5))
print(my_function(9))
```

The pass Statement

function definitions cannot be empty, but if you for some reason have a function definition with no content, put in the pass statement to avoid getting an error.

Example

```
def myfunction():
  pass
```

Recursion

Function calling itself.

Important Programs related to Recursion.

1.Print 1 to 100 using Recursion.

```
def num(n):
    if(n>100):
        return
    print(n)
    num(n+1)

num(1)
```

2.Print 100 to 1 using Recursion.

```
def num(n):
    if(n<1):
        return
    print(n)
    num(n-1)

num(100)
```

Important Programs.

1. Fibonacci Series using recursion.

```python
def fibo(n):
    if(n<=1):
        return n
    else:
        return fibo(n-1)+fibo(n-2)

n=int(input())
print(fibo(n))
```

2.Program for HCF iterative.

```python
a,b=map(int,input().split())

for i in range(1,min(a,b)+1):
    if(a%i==0 and b%i==0):
        hcf=i

print(hcf)
```

3.Program for HCF recursive.

```python
def hcf(a,b):
    if(b==0):
        return a
    else:
        return hcf(b,a%b)


a,b=map(int,input().split())
print(hcf(a,b))
```

4.Program for LCM iterative.

```python
def lcm(a,b):
    greatest=max(a,b)

    while(True):
        if(greatest%a==0 and greatest%b==0):
            lcm=greatest
            break
        greatest+=1
    return lcm
```

```
a,b=map(int,input().split())
print(lcm(a,b))
```

5.Program for LCM recursive.

```
def hcf(a,b):
    if(b==0):
        return a
    else:
        return hcf(b,a%b)
a,b=map(int,input().split())
lcm=int((a*b)/hcf(a,b))

print(lcm)
```