1.Operators in Python

Operators	Type	Type
+,-,*,/,//,%	Arithmetic Operator	Performs operations like addition, subtraction, multiplication, and division.
<, <=,>,>=,==,!=	Relational Operator	In Python Comparison of Relational operators compares the values. It either returns True or False according to the condition.
&&,!,//	Logical Operator	Python Logical operators perform Logical AND, Logical OR, and Logical NOT operations. It is used to combine conditional statements.
&,/,<<,>>,~,^	Bitwise Operator	Python Bitwise operators act on bits and perform bit-by-bit operations. These are used to operate on binary numbers.
=,+=,-=,=,%=	Assignment Operator	Python Assignment operators are used to assign values to the variables.

Logical Operators

Operators	Description	Syntax
and	Logical AND: True if both the operands are true	x and y
or	Relational OperatoLogical OR: True if either of the operands is true	x or y
not	Logical NOT: True if the operand is false	not x

The precedence of Logical Operators in Python is as follows:

- Logical not
- logical and
- logical or

Assignment Operators

Operators	Description	Syntax
=	Assign the value of the right side of the expression to the left side operand	x=y+z
+=	Add AND: Add right-side operand with left-side operand and then assign to left operand	x+=y is same as x=x+y
-=	Subtract AND: Subtract right operand from left operand and then assign to left operand	x-=y is same as x=x-y
*=	Multiply AND: Multiply right operand with left operand and then assign to left operand	x*=y is same as x=x * y
/=	Divide AND: Divide left operand with right operand and then assign to left operand	a/=b is same as a=a/y
%=	Modulus AND: Takes modulus using left and right operands and assign the result to left operand	a%=b is same as a=a%y
//=	Divide(floor) AND: Divide left operand with right operand and then assign the value(floor) to left operand	a//=b is same as a=a//y

Identity Operators in Python

- In Python, **is** and **is not** are identity operators both are used to check if two values are locaed on the same part of memory.

Operators	
is	True if the operands are identical
is not	True if the operands are not identical

Membership Operators in Python

- In Python, in and not in are the membership operators that are used to test whether a **value or variable is in a sequence**.

Operators	
in	True if the value is found in the sequence
not in	True if the value is not found in the sequence

In [1]:
```
x = 24
y = 20
list = [10, 20, 30, 40, 50]

if (x not in list):
    print("x is NOT present in given list")
else:
    print("x is present in given list")

if (y in list):
    print("y is present in given list")
else:
    print("y is NOT present in given list")
```

```
x is NOT present in given list
y is present in given list
```

Ternary Operator

- The Python ternary operator determines if a condition is true or false and then returns the appropriate value in accordance with the result.
- The ternary operator is useful in cases where we need to assign a value to a variable based on a simple condition, and **we want to keep our code more concise — all in just one line of code**.
- It’s particularly handy when you want to avoid writing multiple lines for a simple if-else situation.

In [2]:
```
# Program to demonstrate conditional operator
a, b = 10, 20

# Copy value of a in min if a < b else copy b
min = a if a < b  else b

print(min)
```

```
10
```

In [3]:
```
a, b = 10, 20

if a != b:
    if a > b:
        print("a is greater than b")
    else:
        print("b is greater than a")
else:
    print("Both a and b are equal")
```

```
b is greater than a
```

In [4]:
```
a, b=10, 20

print('Both a and b are equal' if a==b else 'a is greater than b' if a > b else 'b is gr
```

```
b is greater than a
```

Python Ternary Operator using Tuples

- In this example, we are using tuples to demonstrate ternary operator. We are using tuple for selecting an item
- if **[a<b] is true** it return 1, **so element with 1 index will print.**
- else if **[a<b] is false** it return 0, **so element with 0 index will print.**

In [5]:
```
a, b =10,20

print((b,a) [a<b])
```

```
10
```

- Here,output is 10 because at 'index 1' number '10' is present.

Python Ternary Operator using Dictionary

In [6]:
```
a, b =10, 20

print({True: a, False : b} [a < b])
```

```
10
```

Python Ternary Operator using Lambda

- This is a list containing two lambda functions.
- When the condition a < b is True, the first lambda function lambda: 'b' will be selected.
- When the condition a < b is False, the second lambda function lambda: 'a' will be selected.
- ():This calls the lambda function selected from the list.

In [7]:
```
a, b =10, 20

print((lambda : b ,lambda : a)  [a < b]())
```

```
10
```

In [8]:
```
a=5
b=7

# [statement_on_True] if [condition] else [statement_on_false]
print(a,"is greater") if (a>b) else print(b,"is Greater")
```

```
7 is Greater
```

2.List Comprehension

A Python list comprehension consists of brackets containing the expression, which is executed for each element along with the for loop to iterate over each element in the Python list.

- **Syntax: newList = [expression(element) for element in oldList if condition]**
- **Parameter:**
  - **expression:** Represents the operation you want to execute on every item within the iterable.
  - **element:** The term “variable” refers to each value taken from the iterable.
  - **iterable:** specify the sequence of elements you want to iterate through.(e.g., a list, tuple, or string).
  - **condition:** (Optional) A filter helps decide whether or not an element should be added to the new list.
- **Return:** The return value of a list comprehension is a new list containing the modified elements that satisfy the given criteria.

In [9]:
```python
# user input
l_numbers = int(input("Enter the Lower no. of list:"))
u_numbers = int(input("Enter the Upper no. of list:"))

l=[]
for num in range(l_numbers,u_numbers+1):
    l.append(num)
print(l)
#Double of numbers
print('Double of numbers: ',[x*2 for x in l])
```

```
Enter the Lower no. of list:1
Enter the Upper no. of list:20
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
Double of numbers:  [2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 3
6, 38, 40]
```

In [10]:
```python
# user input
l_numbers = int(input("Enter the Lower no. of list:"))
u_numbers = int(input("Enter the Upper no. of list:"))

l=[]
for num in range(l_numbers,u_numbers+1):
    l.append(num)
print(l)
#Print square of numbers
print('Square of numbers: ',[x**2 for x in l])
```

```
Enter the Lower no. of list:1
Enter the Upper no. of list:10
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Square of numbers:  [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

In [11]:
```python
#Printing even numbers with help of list comprehension

print('Even numbers from range of numbers: ',[i for i in range(11) if i%2==0])
```

```
Even numbers from range of numbers:  [0, 2, 4, 6, 8, 10]
```

In [12]:
```python
# Matrix making through list comprehension

[[j for j in range(3)] for i in range(3)]
```

Out[12]:
```
[[0, 1, 2], [0, 1, 2], [0, 1, 2]]
```

In [13]:
```python
#Storing strings in list with help of List comprehension
print([character for character in 'Hello All Welcome to Jupyter Notebook!!!'])

#Returing strings if from List comprehension
print(''.join([character for character in 'Hello All Welcome to Jupyter Notebook!!!']))
```

```
['H', 'e', 'l', 'l', 'o', ' ', 'A', 'l', 'l', ' ', 'W', 'e', 'l', 'c', 'o', 'm', 'e',
' ', 't', 'o', ' ', 'J', 'u', 'p', 'y', 't', 'e', 'r', ' ', 'N', 'o', 't', 'e', 'b',
'o', 'o', 'k', '!', '!', '!']
Hello All Welcome to Jupyter Notebook!!!
```

In [14]:
```python
# Printing List comprehension without after removing spaces from strings
print([character for character in 'Hello All Welcome to Jupyter Notebook!!!' if characte

#Returing strings if from List comprehension
print('\n',''.join([character for character in 'Hello All Welcome to Jupyter Notebook!!!
```

```
['H', 'e', 'l', 'l', 'o', 'A', 'l', 'l', 'W', 'e', 'l', 'c', 'o', 'm', 'e', 't', 'o',
'J', 'u', 'p', 'y', 't', 'e', 'r', 'N', 'o', 't', 'e', 'b', 'o', 'o', 'k', '!', '!',
'!']
 HelloAllWelcometoJupyterNotebook!!!
```

In [15]:
```python
st='Hello All Welcome to Jupyter Notebook!!!'

for word in st:
    print(word[::-1],end='')
```

```
Hello All Welcome to Jupyter Notebook!!!
```

In [16]:
```python
print('Hello All Welcome to Jupyter Notebook!!!')
print()
# Reversing word of string on its place
l=[]
for word in st.split(' '):
    l.append(word[::-1])


print('By traditional for loop: ',' '.join(l))
print()
print('By List Comprehension  : ',' '.join([word[::-1] for word in st.split(' ')]))
```

```
Hello All Welcome to Jupyter Notebook!!!

By traditional for loop:  olleH llA emocleW ot retypuJ !!!koobetoN

By List Comprehension  :  olleH llA emocleW ot retypuJ !!!koobetoN
```

In [1]:
```python
# Squaring using List Comprehension
print('Squaring using List Comprehension:',[i**2 for i in range(11)])
print()

# Squaring using Lambda function
print('Squaring using Lambda function:',list(map(lambda x: x**2,range(11))))
print()

print('Squaring using Lambda function + List Comprehension:',list(map(lambda x: x**2,[x
print()
```

```
Squaring using List Comprehension: [0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

Squaring using Lambda function: [0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

Squaring using Lambda function + List Comprehension: [0, 1, 4, 9, 16, 25, 36, 49, 64,
81, 100]
```

In [2]:
```
# List comprehension using If-Else

[f'Even number : {num}' if num%2==0 else f'Odd number : {num}' for num in range(1,20)]
```

Out[2]:
```
['Odd number : 1',
 'Even number : 2',
 'Odd number : 3',
 'Even number : 4',
 'Odd number : 5',
 'Even number : 6',
 'Odd number : 7',
 'Even number : 8',
 'Odd number : 9',
 'Even number : 10',
 'Odd number : 11',
 'Even number : 12',
 'Odd number : 13',
 'Even number : 14',
 'Odd number : 15',
 'Even number : 16',
 'Odd number : 17',
 'Even number : 18',
 'Odd number : 19']
```

In [3]:
```
# Nested If-Else

[num for num in range(101) if num%5==0 if num%10==0]
```

Out[3]:
```
[0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
```

In [4]:
```
import pandas as pd
names=['Ram','Shyam','Ghanshyam']
ages =[23,24,25]

#Using list comprehension making tuples of name and age
print([(name,age) for name, age  in zip(names,ages)])

pd.DataFrame({(name,age) for name, age  in zip(names,ages)},columns=['Names','Ages'])
```

```
[('Ram', 23), ('Shyam', 24), ('Ghanshyam', 25)]
```

Out[4]:

	Names	Ages
0	Ram	23
1	Shyam	24
2	Ghanshyam	25

In [5]:
```
#Calulating length of each words
words = ["apple", "banana", "cherry", "orange"]
[len(word) for word in words]
```

Out[5]:
```
[5, 6, 6, 6]
```

In [6]:
```
noprimes = [j for i in range(2, 8) for j in range(i*2, 50, i)]
primes = [x for x in range(2, 50) if x not in noprimes]
print (primes)
```

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47]
```

In [7]:
```
print('Using traditional for loop extracting prime nos.: ')
for i in range(2,40):
    for j in range(2,i):
        if i%j==0:
            break
    else:
        print(i,end=' ')
print()
print()
print('\rUsing List Comprehension extracting prime nos.: ',[i for i in range(2, 40) if a
```

```
Using traditional for loop extracting prime nos.: 
2 3 5 7 11 13 17 19 23 29 31 37 


Using List Comprehension extracting prime nos.:  [2, 3, 5, 7, 11, 13, 17, 19, 23, 29,
31, 37]
```

In [8]:
```
string = "my phone number is : 11122 !!"
print('Extracting phone number from the string using list comprehension: ',''.join([x fo
```

```
Extracting phone number from the string using list comprehension:  11122
```

In [9]:
```
print('Doing squaring of numbers,after which are divisble by 5 and are odd nos.: ',list(
```

```
Doing squaring of numbers,after which are divisble by 5 and are odd nos.:  [25, 225, 6
25]
```