**Task 1: Arithmetic Operators**

1. Create two variables a and b with numeric values.
2. Calculate the sum, difference, product, and quotient of a and b.
3. Print the results.

Solution:

a=20  
b=15  
print ("Sum of two numbers:", a+b) #Sum of two numbers: 35  
print ("Difference of two numbers:", a-b) #Difference of two numbers: 5  
print ("Product of two numbers:", a\*b) #Product of two numbers: 300  
print ("Division of two numbers:", a/b) #Division of two numbers: 1.33333

**Task 2: Comparison Operators**

1. Compare the values of a and b using the following comparison operators: <, >, <=, >=, ==, and !=.
2. Print the results of each comparison.

Solution:

a=20  
b=15  
print("comparing whether a is greater than b:", a>b)  
print("comparing whether a is less than b:", a<b)  
print("comparing whether a is greater than equal to b:", a>=b)  
print("comparing whether a is less than equal to b:", a<=b)  
print("comparing whether a is equal to b:", a==b)  
print("comparing whether a is not equal to b:", a!=b)

output:

comparing whether a is greater than b: True  
comparing whether a is less than b: False  
comparing whether a is greater than equal to b: True  
comparing whether a is less than equal to b: False  
comparing whether a is equal to b: False  
comparing whether a is not equal to b: Tru

**Task 3: Logical Operators**

1. Create two boolean variables, x and y.
2. Use logical operators (and, or, not) to perform various logical operations on x and y.
3. Print the results.

Solution:

x=True  
y=False  
print ("and:", x and y)  
print ("or:", x or y )  
print ("not:", not x)

output:

and: False  
or: True  
not: False

**Task 4: Assignment Operators**

1. Create a variable total and initialize it to 10.
2. Use assignment operators (+=, -=, \*=, /=) to update the value of total.
3. Print the final value of total.

Solution:

total=10  
total += 10  
total -= 5  
total \*= 3  
total /= 3  
print ("Final value of total:", total)

output:

Final value of total: 15.0

**Task 5: Bitwise Operators (Optional)**

1. If you are comfortable with bitwise operators, perform some bitwise operations on integer values and print the results. If not, you can skip this task.

Solution:

a=5  
b=6  
print("bitwise and:", a&b)  
print("bitwise or:", a|b)  
print("bitwise xor:", a^b)  
print("bitwise not:", ~a)  
print("leftshift:", a<<1)  
print("rigthshift:", b>>1)

output:

bitwise and: 4  
bitwise or: 7  
bitwise xor: 3  
bitwise not: -6  
leftshift: 10  
rigthshift: 3

**Task 6: Identity and Membership Operators**

1. Create a list my\_list containing a few elements.
2. Use identity operators (is and is not) to check if two variables are the same object.
3. Use membership operators (in and not in) to check if an element is present in my\_list.
4. Print the results.

Solution:

my\_list=["hello", "everyone", "how", "was", "your", "day"]  
new\_list=my\_list  
print(my\_list is new\_list)  
print(my\_list is not new\_list)  
print("hello" in my\_list)  
print("is" not in my\_list)

output:

True  
False  
True  
True

**List Methods:**

**append():** The append() method adds an element to the end of the list.

list1 = ["I am", "learning", "python"]  
list1.append("programming")  
print(list1)

output:

['I am', 'learning', 'python', 'programming']

**extend():** The extend() method adds the specified list elements or any iterable to the end of the current list.

list1=["ben","john","jack","tessa"]  
list2=["clay","22"]  
list1.extend(list2)  
print(list1)

output:

['ben', 'john', 'jack', 'tessa', 'clay', '22']

**insert():** The insert method inserts the specific value at specific position.

list1=["ben","john","jack","tessa"]  
list1.insert(2,"clay")  
print(list1)

output:

['ben', 'john', 'clay', 'jack', 'tessa']

**pop():** The pop() method removes the element at the specified position. And by default it removes last element from the list.

list1=["ben","john","jack","tessa"]  
list1.pop(2)  
print(list1)

output:

['ben', 'john', 'tessa']

**remove():** The remove() method removes the first occurrence of the element with the specified value.

list1=["ben","tessa","john","jack","tessa"]  
list1.remove("tessa")  
print(list1)

output:

['ben', 'john', 'jack', 'tessa']

**clear():** The clear() method removes all the elements from a list.

list1=["ben","tessa","john","jack","tessa"]  
list1.clear()  
print(list1)

output:

[]

**count():** The count() method returns the number of elements with the specified value.

list1=["ben","tessa","john","jack","tessa"]  
x=list1.count("tessa")  
print(x)

output:

2

**copy():** Returns a copy of the list

list1= ["ben", "john", "jack"]  
x = list1.copy()  
print(x)

output:

['ben', 'john', 'jack']

**index():** The index() method returns the position at the first occurrence of the specified value.

num = [4, 55, 64, 32, 16, 32]  
  
x = num.index(32)  
print(x)

output:

3

**sort():** sorts in the order by default it is done in ascending order.

num = [4, 55, 64, 32, 16, 32]  
num.sort()  
print(num)

output:

[4,16,32,32,55,64]

And we can also sort in descending order.

num = [4, 55, 64, 32, 16, 32]  
num.sort(reverse=True)  
print(num)

output:

[64, 55, 32, 32, 16, 4]

**reverse():** it reverse the order of the list.

num = [4, 55, 64, 32, 16, 32]  
num.reverse()  
print(num)

output:

[32, 16, 32, 64, 55, 4]

**Access list items:**

List items are indexed and you can access them by referring to the index number.

num = [4, 55, 64, 32, 16, 32]  
print(num[2])

output:

64

num = [4, 55, 64, 32, 16, 32]  
print(num[-2])

output:

16

**Range of Indexes:**

You can specify a range of indexes by specifying where to start and where to end the range.

When specifying a range, the return value will be a new list with the specified items.

num = [4, 55, 64, 32, 16, 32]

print(num[2:5])

output:

[64,32,16]

len(): It returns the length of the list.

num = [4, 55, 64, 32, 16, 32]

print(len(num))

output:

6

**Tuple Methods:**

Creating a tuple:

Tuple items are ordered, unchangeable, and allow duplicate values**.**

We can create a tuple by using ().

thistuple = ("python", "sql", "java")

print(thistuple)

output:

('python', 'sql', 'java')

**Access tuple items:**

We can access tuple items by referring to the index number, inside square brackets.

thistuple = ("python", "sql", "java")  
print(thistuple[1])

output:

sql

thistuple = ("python", "sql", "java")  
print(thistuple[-1])

output:

java

thistuple = ("python", "sql", "java")  
print(thistuple[:2])

output:

('python', 'sql')

**Length of a tuple:**

By using len() function we can find the length of a tuple.

thistuple = ("python", "sql", "java")

print(len(thistuple))

output:

3

**count():** The count() method returns the number of elements with the specified value.

thistuple = ("python", "sql", "java")  
print(thistuple.count("sql"))

output:

1

**Loop through a tuple:**

We can loop through the tuple items by using a for loop.

thistuple = ("apple", "banana", "cherry")  
for x in thistuple:  
  print(x)

output:

apple

banana

cherry

Membership in tuple:

num = (10,22,13,22,45)

if 13 in num:

print (True)

Concatenation:

Combining two tuples into a single tuple.

tuple1 = (10,20,30,40,50)

tuple2 = (70,60,80)

result=tuple1+tuple2

print(result)

output:

(10, 20, 30, 40, 50, 70, 60, 80)