In [4]:
```
#Dictionary Collection or Data type in Python
dict1 ={'name':'shankar','age':26}
print(dict1['name']) # Using key we are retrieving value from a dictionary
print(dict1)# using print function displaying the content of dict1 dictionary
print(type(dict1)) #using type function we getting the type of varaiable dict1
print(dir(dict1)) #using dir function we can get the attributes and functions define in the class <'dict'>
```

```
shankar
{'name': 'shankar', 'age': 26}
<class 'dict'>
['__class__', '__class_getitem__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc__', '__eq__', '__format
__', '__ge__', '__getattribute__', '__getitem__', '__getstate__', '__gt__', '__hash__', '__init__', '__init_subclass__',
'__ior__', '__iter__', '__le__', '__len__', '__lt__', '__ne__', '__new__', '__or__', '__reduce__', '__reduce_ex__', '__rep
r__', '__reversed__', '__ror__', '__setattr__', '__setitem__', '__sizeof__', '__str__', '__subclasshook__', 'clear', 'cop
y', 'fromkeys', 'get', 'items', 'keys', 'pop', 'popitem', 'setdefault', 'update', 'values']
```

In [15]:
```
dict2 = {"brand":"maruti","model":"ertiga","year":2020}
print(dict2)
print(dict2["model"])# Retreiving value
print(dict2.keys()) #Get all the keys from a dictionary
print(dict2.values())#Get all the values from a dictionary
print(dict2.get("year"))#Retreving value
print(dict2["year"])
print(dict2.items()) #This will all display key-value pairs
#adding new key-value pair
dict2["color"] = "white"
print(dict2)

dict2.popitem() #deletes the last item
print(dict2)
dict2.pop("model") #deletes the specified element
print(dict2)
del dict2["brand"] #deletes the specified element
print(dict2)
print("**************************")
#dict2.pop() #We will get error because pop function expects one argument
print(dict2)
del dict2 #It has deleted entire dict2
print(dict2)#This we will get error as dict2 doesn't exist
```

```
{'brand': 'maruti', 'model': 'ertiga', 'year': 2020}
ertiga
dict_keys(['brand', 'model', 'year'])
dict_values(['maruti', 'ertiga', 2020])
2020
2020
dict_items([('brand', 'maruti'), ('model', 'ertiga'), ('year', 2020)])
{'brand': 'maruti', 'model': 'ertiga', 'year': 2020, 'color': 'white'}
{'brand': 'maruti', 'model': 'ertiga', 'year': 2020}
{'brand': 'maruti', 'year': 2020}
{'year': 2020}
**************************
{'year': 2020}
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[15], line 23
     21 print(dict2)
     22 del dict2
---> 23 print(dict2)

NameError: name 'dict2' is not defined
```

In [25]:

```
dict1 = {"brand":"maruti","model":"ertiga","year":2020}
print(dict1)
for x,y in dict1.items():
    print(x+" -- "+str(y))
print("****************************")
for x,y in dict1.items():
    print(x,y)
print("****************************")
print(dict1.fromkeys("model"))
dict2 = dict1.copy()
print(dict2)
```

```
{'brand': 'maruti', 'model': 'ertiga', 'year': 2020}
brand -- maruti
model -- ertiga
year -- 2020
****************************
brand maruti
model ertiga
year 2020
****************************
{'m': None, 'o': None, 'd': None, 'e': None, 'l': None}
{'brand': 'maruti', 'model': 'ertiga', 'year': 2020}
```

In [27]:
```
#Operators
#Airthmetic Operators
a=10
b=20
resOfSum=a+b
print(resOfSum)
resOfSub=a-b
print(resOfSub)
resOfMul=a*b
print(resOfMul)
resOfDiv=a/b
print(resOfDiv)
resOfMod=a%b
print(resOfMod)
resOfExp = a**b
print(resOfExp)
```

```
30
-10
200
0.5
10
100000000000000000000
```

In [34]:
```
#Comparison Operators
a=21
b=10
c=0
resOfEqEq=(a==b)
print(a==b)
print(resOfEqEq)
print(a>b)
print(a<b)
print(a!=b)
print(a>=b)
print(b<=a)
print("*******************a==b*************************")
if(a==b):
    #if block
    print("a is equal to b")
else:
    #else block
    print("a is not equal to b")
print("*******************a!=b*************************")
if(a!=b):
    #if block
    print("a is not equal to b")
else:
    #else block
    print("a is equal to b")
print("*******************a>=b*************************")
if(a>=b):
    #if block
    print("a is greater than equal to b")
else:
    #else block
    print("a is not greater than equal to b")
print("*******************a<=b*************************")
if(a<=b):
    #if block
    print("a is less than equal to b")
else:
    #else block
    print("a is not less than equal to b")
print("*******************a<b*************************")
if(a<b):
    #if block
    print("a is less than b")
else:
    #else block
    print("a is not less than b")
print("*******************a>b*************************")
if(a>b):
    #if block
    print("a is greater than b")
else:
    #else block
    print("a is not greater than b")
```

```
False
False
True
False
True
True
True
*******************a==b*************************
a is not equal to b
*******************a!=b*************************
a is not equal to b
*******************a>=b*************************
a is greater than equal to b
*******************a<=b*************************
a is not less than equal to b
*******************a<b*************************
a is not less than b
*******************a>b*************************
a is greater than b
```

In [40]:

```
#Assignment Operators
a=10 # = is an assignment operator
# +=
a+=5 # a=a+5  -- a=10+5=15
print(a)
# -=
a-=5 #a=a-5  -- a=15-5=10
print(a)
# *=
a*=5 #a=a*5  -- a=10*5=50
print(a)
# /=
a/=5 #a=a/5  -- a=50/5=10.0
print(a)
# %=
a%=5 #a=a%5  -- a=10.0%5=0  -- we will get Remainder
print(a)
# **=
a=2
a**=5 #a=a**5  -- a=2**5=32
print(a)
```

```
15
10
50
10.0
0.0
32
```

In [44]:

```
x=10
binOfX=bin(x)# bin() is converting your decimal number 10 referred by x into binary form
print(binOfX)
print(oct(10))
print(hex(10))
binNum="1011"
print(int(binNum,2)) #here base 2 indicates the binary number
```

```
0b1010
0o12
0xa
11
```

In [55]:

```
#Bitwise operator
A=10
B=7
'''
    A= 0000 0000 0000 1010
    B= 0000 0000 0000 0111
  A&B= 0000 0000 0000 0010
'''
print(A&B)
print(bin(A))
print(bin(B))
print(bin(A&B))
print("**************Bitwise OR Operator")
'''
    A= 0000 0000 0000 1010
    B= 0000 0000 0000 0111
  A|B= 0000 0000 0000 1111
'''
print(A|B)
print(bin(A|B))
print("**************Bitwise XOR Operator")
'''
    A= 0000 0000 0000 1010
    B= 0000 0000 0000 0111
  A^B= 0000 0000 0000 1101
```

```
'''
print(A^B)
print(bin(A^B))
print("**************One's complement Operator")
A=10
print(~A)
print(bin(A))
print(bin(~A))
print("**************Left Shift Operator**********")
'''
    A= 0000 0000 0000 1010
 A<<2= 0000 0000 0010 1000  = 2 raise to pow 5 + 2 raise to pow 3= 32+8=40
'''

print(A<<2)
print(bin(A))
print(bin(A<<2))
print("**************Right Shift Operator**********")
'''
    A= 0000 0000 0000 1010
 A>>2= 0000 0000 0000 0010  = 2 raise to pow 1 =2
'''

print(A>>2)
print(bin(A))
print(bin(A>>2))
```

```
2
0b1010
0b111
0b10
*************Bitwise OR Operator
15
0b1111
*************Bitwise XOR Operator
13
0b1101
*************One's complement Operator
-11
0b1010
-0b1011
*************Left Shift Operator**********
40
0b1010
0b101000
*************Right Shift Operator**********
2
0b1010
0b10
```

In [62]:

```
#Logical Operators
x=True
y=False
#y=True
print('x and y is ',x and y)
print('x or y is ',x or y)
print('not x is ', not x)
print('not y is ', not y)
```

```
x and y is  False
x or y is  True
not x is  False
not y is  True
```

In [72]:

```
#Membership Operators
X=[1,2,3,4,5,6,7,8,9]
numToBeSearched = 6
print(numToBeSearched in X)
print(numToBeSearched not in X)
numToBeSearched = 11
print(numToBeSearched in X)
print(numToBeSearched not in X)
```

```
listOfBooks=["Java","Strategy",5,6,8,"Statistics","Marketing"]
#itemToBeSearched="Operations"
itemToBeSearched="Statistics"
print(itemToBeSearched in listOfBooks)
print(itemToBeSearched not in listOfBooks)
print("***************************************")
strHi ="Hi How are you ?"
print('are' in strHi)
print('a' not in strHi)
print('a' in strHi)
```

```
True
False
False
True
True
False
*************************************
True
False
True
```

In [74]:
```
#Identity Operator
A=10
B=10
print(A is B)
print(A is not B)
B=5
print(A is not B)
```

```
True
False
True
```

In [80]:
```
#Conditional Statement
X=15
Y=10
if(X<Y):
    print('X is less than Y')
    print('Happy Learning')
elif(X>Y):
    print('X is greater than Y')
    print('OK Happy Learning')
elif(X==Y):
    print('X is equal to Y')
    print('OK OK Happy Learning')
```

```
else:
    print('Good Afternoon All !!! Bye All See you tomorrow')
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[80], line 4
      2 X=15
      3 Y='Hello'
----> 4 if(X<Y):
      5     print('X is less than Y')
      6     print('Happy Learning')

TypeError: '<' not supported between instances of 'int' and 'str'
```

In [ ]: