6.Bitwise XOR operator (^)

In [ ]:
```
""" XOR (Exclusive OR) Operator
        •       Symbol: ^
        •       Works bit by bit on binary values.
        •       Rule:
        •       If the two bits are different, the result is 1.
        •       If the two bits are same, the result is 0."""
```

In [4]:
```
1^0
```
Out[4]:
```
1
```

In [6]:
```
0^1
```
Out[6]:
```
1
```

In [8]:
```
1^1
```
Out[8]:
```
0
```

In [10]:
```
0^0
```
Out[10]:
```
0
```

In [9]:
```
2 ^ 2
```
Out[9]:
```
0
```

In [11]:
```
2 ^ 3
```
Out[11]:
```
1
```

In [18]:
```
s1={20,30,40}
s2={30,40,50}
s3=s1 ^ s2
print(s3,type(s3))
```
```
{50, 20} <class 'set'>
```

In [20]:
```
s1={2.3,3.0,4.0}
s2={3.0,4.0,5.0}
s3=s1 ^ s2
print(s3,type(s3))
```
```
{2.3, 5.0} <class 'set'>
```

In [22]:
```
s1={"A","B","C"}
s2={"A","B","C"}
s3=s1 ^ s2
print(s3,type(s3))
```

```
set() <class 'set'>
```

In [26]:
```
set() ^ {10,20,30,40,50}
```

Out[26]:
```
{10, 20, 30, 40, 50}
```

In [2]:
```
a,b=3,4
print(a,b)
```

```
3 4
```

In [4]:
```
a=a^b
b=a^b
a=a^b
print(a,b)
```

```
4 3
```

“Write a Python program that accepts two integer values and swaps them using bitwise XOR.”

In [15]:
```
a=int(input("Enter the value of a:"))
b=int(input("Enter the value of b:"))
print("----------------------------")
print("orginal value of a={}".format(a))
print("orginal value of b={}".format(b))
print("----------------------------")
a=a^b
b=a^b
a=a^b
print("swapped value of a={}".format(a))
print("swapped value of b={}".format(b))
print("----------------------------")
```

```
----------------------------
orginal value of a=100
orginal value of b=200
----------------------------
swapped value of a=200
swapped value of b=100
----------------------------
```

In [17]:
```
a=int(input("Enter the value of a:"))
b=int(input("Enter the value of b:"))
print("----------------------------")
print("orginal value of a={}".format(a))
print("orginal value of b={}".format(b))
print("----------------------------")
a=a^b
b=a^b
a=a^b
print(f"swapped value of a={a}")
print(f"swapped value of b={b}")
print("----------------------------")
```

```
----------------------------
orginal value of a=100
orginal value of b=200
----------------------------
swapped value of a=200
swapped value of b=100
----------------------------
```

MEMBERSHIP OPERATORS:-

In [ ]:
```
"""Membership Operators in Python
Purpose: Membership operators are used to check whether a value is present in an it
Iterable Objects: These are objects that contain more than one value. Examples incl

Sequence types (like strings and tuples)
Lists
Sets
Dictionaries

Non-Iterable Objects: These contain only a single value. Examples include:
int, float, bool, complex, and NoneType

Types of Membership Operators in Python:
1. in operator – Returns True if the specified value is found in the iterable.
2. not in operator – Returns True if the specified value is not found in the iterab
```

in operator

"""Syntax: value in iterable_object The in operator returns True if the specified value is present in the iterable object. The in operator returns False if the specified value is not present in the iterable object."""

not in operator

2. not in Operator Syntax: value not in iterable_object The not in operator returns True if the specified value is not present in the iterable object. The not in operator returns False if the specified value is present in the iterable object.

In [10]:
```
s="Python"
"P" in s
```

Out[10]:
```
True
```

In [12]:
```
"t" in s
```

Out[12]:
```
True
```

In [14]:
```
"n" in s
```

Out[14]:
```
True
```

In [16]:
```
"p" in s #asking small letter "p" in word "Python"
```

```
Out[16]:  False
```

```
In [18]:  "t" not in s
```

```
Out[18]:  False
```

```
In [22]:  "Y" not in s #ASKING CAPITAL LETTER "Y" IN WORD "Python"
```

```
Out[22]:  True
```

```
In [26]:  s="Python" #LETTER SEQUENCE NOT MATCHING TO WORD (STRING)
          "PTO" in s
```

```
Out[26]:  False
```

```
In [28]:  "PYN" not in s
```

```
Out[28]:  True
```

```
In [30]:  "Pyn" not in s
```

```
Out[30]:  True
```

```
In [32]:  "noh" in s[::-1] #REVERS LOGIC
```

```
Out[32]:  True
```

```
In [34]:  "noh" not in s
```

```
Out[34]:  True
```

```
In [36]:  "noh" not in s[::-1]
```

```
Out[36]:  False
```

```
In [40]:  lst=["KHAN", 10, "MRIIRS",2+3j]
```

```
In [42]:  10 in lst
```

```
Out[42]:  True
```

```
In [44]:  "MRIIRS" in lst
```

```
Out[44]:  True
```

```
In [46]:  "MRI" in lst
```

```
Out[46]:  False
```

```
In [48]:  "KHA" in lst[0]
```

```
Out[48]: True
```

```
In [50]: 2+3j in lst[-1] #Non iterable
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[50], line 1
----> 1 2+3j in lst[-1]

TypeError: argument of type 'complex' is not iterable
```

```
In [52]: 2+3j in str(lst[-1])
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[52], line 1
----> 1 2+3j in str(lst[-1])

TypeError: 'in <string>' requires string as left operand, not complex
```

```
In [56]: lst=["KHAN", 10, "MRIIRS",2+3j]
         lst[2][::-2] in lst[-2][::][::-2]
```

```
Out[56]: True
```

Identity Operators (Applicable in Python only)

Identity Operators (Applicable in Python only) Purpose: The purpose of identity operators is to compare the memory addresses (references) of two objects. In Python, there are two types of identity operators: 1. is operator 2. is not operator

```
In [ ]: """1. is Operator

Syntax:
object1 is object2

The is operator returns True if both objects refer to the same memory address.
The is operator returns False if both objects refer to different memory addresses.

2. is not Operator

Syntax:
object1 is not object2

The is not operator returns True if both objects refer to different memory addresse
The is not operator returns False if both objects refer to the same memory address.
```

```
In [65]: lst1=[10,20,30,40]
         lst2=lst1 # deep copy
         lst1 is lst2
```

```
Out[65]: True
```

```
In [67]: lst1 is lst2
```

```
Out[67]: True
```

```
In [77]: lst=[10,20,30,40]
         lst2=lst1.copy() #shallow coppy
         lst2=lst1.copy()
         print(id(lst1),id(lst2))
```

```
1456442424832 1456419429184
```

```
In [79]: lst1 is lst2
```

```
Out[79]: False
```

```
In [81]: lst1 is not lst2
```

```
Out[81]: True
```

'Extra Examples will cover in next session. Thank you,'