In [1]:
```
1 + 1
```

Out[1]: 2

In [2]:
```
2 - 1
```

Out[2]: 1

In [3]:
```
3 *4
```

Out[3]: 12

In [4]:
```
8/4
```

Out[4]: 2.0

In [5]:
```
8/5    # Float division
```

Out[5]: 1.6

In [6]:
```
8//4   # integer division
```

Out[6]: 2

In [7]:
```
8 + 9 - 7
```

Out[7]: 10

In [8]:
```
8 + 8 -   # syntax error
```

```
  Cell In[8], line 1
    8 + 8 -   # syntax error
              ^
SyntaxError: invalid syntax
```

In [10]:
```
5 + 5
```

Out[10]: 10

In [11]:
```
(5+5)*5
```

Out[11]: 50

In [12]:
```
5+5*5
```

Out[12]: 30

In [13]:
```
2 * 2 * 2 * 2  *  2
```

Out[13]: 32

In [14]:
```
2 * 5
```

Out[14]: 10

In [15]:
```
2 ** 5
```

Out[15]: 32

In [16]:
```
3 ** 4
```

Out[16]:
```
81
```

In [17]:
```
15 / 3
```

Out[17]:
```
5.0
```

In [18]:
```
10//3
```

Out[18]:
```
3
```

In [19]:
```
15 % 2 # Modulus
```

Out[19]:
```
1
```

In [20]:
```
10 % 2
```

Out[20]:
```
0
```

In [21]:
```
15 %% 2
```

```
  Cell In[21], line 1
    15 %% 2
       ^
SyntaxError: invalid syntax
```

In [22]:
```
-10 // 3
```

Out[22]:
```
-4
```

In [23]:
```
3 + 'nit'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[23], line 1
----> 1 3 + 'nit'

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

In [24]:
```
3 * 'nit'
```

Out[24]:
```
'nitnitnit'
```

In [25]:
```
3 * ' nit'
```

Out[25]:
```
' nit nit nit'
```

In [26]:
```
a,b,c,d,e = 15,7.8,'nit',8+9j,True

print(a)
print(b)
print(c)
print(d)
print(e)
```

```
15
7.8
nit
(8+9j)
True
```

In [27]:
```
print(type(a))
print(type(b))
print(type(c))
print(type(d))
print(type(e))
```

```
<class 'int'>
<class 'float'>
<class 'str'>
<class 'complex'>
<class 'bool'>
```

In [28]:
```
print('Max IT')
```

```
Max IT
```

In [29]:
```
"Max IT Technology"
```

Out[29]:
```
'Max IT Technology'
```

In [30]:
```
s1 = ' Max IT Technology'
s1
```

Out[30]:
```
' Max IT Technology'
```

In [31]:
```
a = 2
b = 3
a + b
```

Out[31]:
```
5
```

In [32]:
```
c = a + b
c
```

Out[32]:
```
5
```

In [33]:
```
a = 3
b = 'hi'
c = a + b
print(c)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[33], line 3
      1 a = 3
      2 b = 'hi'
----> 3 c = a + b
      4 print(c)

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

In [34]:
```
print('max it's "Technlogy"')   # \ has some special meaning to ignore the error
```

```
  Cell In[34], line 1
    print('max it's "Technlogy"')   # \ has some special meaning to ignore the error
                                ^
SyntaxError: unterminated string literal (detected at line 1)
```

In [35]:
```
print('max it\'s "Technlogy"')
```

```
max it's "Technlogy"
```

In [36]:
```
print('max it', 'Technology')
```

```
max it Technology
```

In [37]:
```
print("max it','Technology")
```

```
max it','Technology
```

In [38]:
```
'nit'  + ' nit'
```

Out[38]:
```
'nit nit'
```

In [39]:
```
'nit' ' nit'
```

Out[39]:
```
'nit nit'
```

In [40]:
```
5 * 'nit'
```

Out[40]:
```
'nitnitnitnitnit'
```

In [41]:
```
5 * ' nit'
```

Out[41]:
```
' nit nit nit nit nit'
```

In [42]:
```
print('c:\nit')   #\n -- new line # i will explain
```

```
c:
it
```

In [43]:
```
print(r'c:\nit')   # raw string # I will explain later
```

```
c:\nit
```

Variable Identifier Object

In [44]:
```
2
```

Out[44]:
```
2
```

In [45]:
```
x = 2
x
```

Out[45]:
```
2
```

In [46]:
```
x + 3
```

Out[46]:
```
5
```

In [47]:
```
y = 5
```

In [48]:
```
x + y
```

Out[48]:
```
7
```

In [49]:
```
x = 9
```

In [50]:
```
x + y
```

Out[50]:
```
14
```

In [51]:
```
x + 10
```

Out[51]: 19

In [52]:
```
_ + y   # _ understand the previous result
```

Out[52]: 24

In [53]:
```
_ +y
```

Out[53]: 29

In [54]:
```
_ + y
```

Out[54]: 34

In [55]:
```
# string variable
name = 'mit'
```

In [56]:
```
name
```

Out[56]: 'mit'

In [57]:
```
name + 'technology'
```

Out[57]: 'mittechnology'

In [58]:
```
name + ' technology'
```

Out[58]: 'mit technology'

In [59]:
```
'a' 'b'
```

Out[59]: 'ab'

In [60]:
```
name 'technology'
```

```
  Cell In[60], line 1
    name 'technology'
         ^
SyntaxError: invalid syntax
```

In [61]:
```
name
```

Out[61]: 'mit'

In [62]:
```
len(name)
```

Out[62]: 3

In [63]:
```
name[0]
```

Out[63]: 'm'

In [64]:
```
name[2]
```

Out[64]: 't'

In [65]:
```
name[-1]
```

Out[65]: 't'

In [66]:
```
name[-2]
```

Out[66]: 'i'

Slicing

In [67]:
```
name
```

Out[67]: 'mit'

In [68]:
```
name[0:1]
```

Out[68]: 'm'

In [69]:
```
name[0:2]
```

Out[69]: 'mi'

In [70]:
```
name[1:4]
```

Out[70]: 'it'

In [71]:
```
name[1:]
```

Out[71]: 'it'

In [72]:
```
name[:4]
```

Out[72]: 'mit'

In [73]:
```
name1 = 'fine'  # change the string fine to dine
name1
```

Out[73]: 'fine'

In [74]:
```
name1[0:1]
```

Out[74]: 'f'

In [75]:
```
name1[0:4]
```

Out[75]: 'fine'

In [76]:
```
name[0:1] ='d'   # want o change 1 st characer of naresh (n) - d
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[76], line 1
----> 1 name[0:1] ='d'

TypeError: 'str' object does not support item assignment
```

In [77]:
```
name1[0] = 'd'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[77], line 1
----> 1 name1[0] = 'd'

TypeError: 'str' object does not support item assignment
```

In [78]:
```
name1
```

Out[78]: 'fine'

In [79]:
```
name1[1:]
```

Out[79]: 'ine'

In [80]:
```
'd'+name1[1:]
```

Out[80]: 'dine'

In [81]:
```
len(name1)
```

Out[81]: 4

LIST

In [4]:
```
l = []
```

In [5]:
```
# list list list
nums = [10,20,30]
```

In [6]:
```
nums[0]
```

Out[6]: 10

In [7]:
```
nums[-1]
```

Out[7]: 30

In [8]:
```
nums[:1]
```

Out[8]: [10]

In [9]:
```
nums[1:]
```

Out[9]: [20, 30]

In [10]:
```
num1 = ['hi','hello']
```

In [11]:
```
num1
```

Out[11]: ['hi', 'hello']

In [12]:
```
num2 = ['hi',8.9, 34]   # we can assign multiple variable
num2
```

Out[12]: ['hi', 8.9, 34]

In [13]:
```
# can we have 2 list together
num3 = [nums, num1]
```

In [14]:
```
num3
```

Out[14]: [[10, 20, 30], ['hi', 'hello']]

In [15]:
```
num4 = [nums, num1 , num2]
```

In [16]:
```
num4
```

Out[16]:
```
[[10, 20, 30], ['hi', 'hello'], ['hi', 8.9, 34]]
```

In [17]:
```
nums
```

Out[17]:
```
[10, 20, 30]
```

In [18]:
```
nums.append(45)
```

In [19]:
```
nums
```

Out[19]:
```
[10, 20, 30, 45]
```

In [20]:
```
nums.remove(45)
```

In [21]:
```
nums
```

Out[21]:
```
[10, 20, 30]
```

In [22]:
```
nums.pop(1)
```

Out[22]:
```
20
```

In [23]:
```
nums
```

Out[23]:
```
[10, 30]
```

In [24]:
```
nums.pop()         #if you dont assign the index element then it will consider by default last ind
```

Out[24]:
```
30
```

In [25]:
```
nums
```

Out[25]:
```
[10]
```

In [26]:
```
num1
```

Out[26]:
```
['hi', 'hello']
```

In [27]:
```
num1.insert(2,'nit')  # insert the value as per index value i.e 2nd index we are assigning nit
```

In [28]:
```
num1
```

Out[28]:
```
['hi', 'hello', 'nit']
```

In [29]:
```
num1.insert(0, 1)
```

In [30]:
```
num1
```

Out[30]:
```
[1, 'hi', 'hello', 'nit']
```

In [31]:
```
# if you want to delete multiple values
num2
```

Out[31]:
```
['hi', 8.9, 34]
```

In [32]:
```
del num2[2:]
```

In [33]:
```
num2
```

Out[33]:
```
['hi', 8.9]
```

In [35]:
```
# if you need to add multiple values
num2.extend([29,15,20])
```

In [36]:
```
num2
```

Out[36]:
```
['hi', 8.9, 29, 15, 20]
```

In [37]:
```
num3
```

Out[37]:
```
[[10], [1, 'hi', 'hello', 'nit']]
```

In [38]:
```
nums
```

Out[38]:
```
[10]
```

In [39]:
```
min(nums)    #inbuild function
```

Out[39]:
```
10
```

In [40]:
```
max(nums)   #inbuild function
```

Out[40]:
```
10
```

In [45]:
```
num1
```

Out[45]:
```
[1, 'hi', 'hello', 'nit']
```

In [46]:
```
min(num1)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[46], line 1
----> 1 min(num1)

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

In [47]:
```
nums.sort()   # sort method
```

In [48]:
```
nums
```

Out[48]:
```
[10]
```

In [49]:
```
l = [1,2,3]
l
```

Out[49]:
```
[1, 2, 3]
```

In [50]:
```
l[0] = 100
l
```

Out[50]:
```
[100, 2, 3]
```

Tuple

In [55]:
```
# TUPLE TUPLE TUPLE
```

In [56]:
```
tup = (15,25,35)
tup
```

Out[56]:
```
(15, 25, 35)
```

In [57]:
```
tup[0]
```

Out[57]:
```
15
```

In [58]:
```
tup[0] = 10
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[58], line 1
----> 1 tup[0] = 10

TypeError: 'tuple' object does not support item assignment
```

SET

In [59]:
```
# SET SET SET
```

In [61]:
```
s = {}
```

In [62]:
```
s1 = {21,6,34,58,5}
```

In [63]:
```
s1
```

Out[63]:
```
{5, 6, 21, 34, 58}
```

In [64]:
```
s3 = {50,35,53,'nit',53}
```

In [65]:
```
s3
```

Out[65]:
```
{35, 50, 53, 'nit'}
```

In [66]:
```
s1[1]        #as we dont have proper sequencing thats why indexing not subscriptable
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[66], line 1
----> 1 s1[1]

TypeError: 'set' object is not subscriptable
```

Dictionary

In [67]:
```
# Dictionary Dictionary Dictionary
data = {1:'apple',2:'banana',4:'orange'}
data
```

Out[67]:
```
{1: 'apple', 2: 'banana', 4: 'orange'}
```

In [68]:
```
data[4]
```

Out[68]:
```
'orange'
```

In [70]:
```
data.get(2)
```

Out[70]:
```
'banana'
```

In [72]:
```
data.get(3)
```

In [73]:
```
print(data.get(3))
```

```
None
```

In [74]:
```
data.get(1,'Not Fount')
```

Out[74]:
```
'apple'
```

In [75]:
```
data.get(3, 'Not Found')
```

Out[75]:
```
'Not Found'
```

In [76]:
```
data[5] = 'five'
```

In [77]:
```
data
```

Out[77]:
```
{1: 'apple', 2: 'banana', 4: 'orange', 5: 'five'}
```

In [78]:
```
del data[5]
```

In [79]:
```
data
```

Out[79]:
```
{1: 'apple', 2: 'banana', 4: 'orange'}
```

In [81]:
```
t in the dictionary
= {'python':['vscode','pycharm'],'machine learning':'sklearn','datascience':['jupyter','spyder']}
```

In [82]:
```
prog
```

Out[82]:
```
{'python': ['vscode', 'pycharm'],
 'machine learning': 'sklearn',
 'datascience': ['jupyter', 'spyder']}
```

In [83]:
```
prog['python']
```

Out[83]:
```
['vscode', 'pycharm']
```

In [84]:
```
prog['machine learning']
```

Out[84]:
```
'sklearn'
```

In [85]:
```
prog['datascience']
```

Out[85]:
```
['jupyter', 'spyder']
```

In [87]:
```
2 + 3
```

Out[87]:
```
5
```

In [89]:
```
help()
```

```
Welcome to Python 3.11's help utility!

If this is your first time using Python, you should definitely check out
the tutorial on the internet at https://docs.python.org/3.11/tutorial/. (https://docs.python.org/
3.11/tutorial/.)

Enter the name of any module, keyword, or topic to get help on writing
Python programs and using Python modules.  To quit this help utility and
return to the interpreter, just type "quit".

To get a list of available modules, keywords, symbols, or topics, type
"modules", "keywords", "symbols", or "topics".  Each module also comes
with a one-line summary of what it does; to list the modules whose name
or summary contain a given string such as "spam", type "modules spam".

help> keywords

Here is a list of the Python keywords.  Enter any keyword to get more help.

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

help> q

You are now leaving help and returning to the Python interpreter.
If you want to ask for help on a particular object directly from the
interpreter, you can type "help(object)".  Executing "help('string')"
has the same effect as typing a particular string at the help> prompt.
```

Introduce to ID()

In [90]:
```
# variable address
num = 5
id(num)
```

Out[90]:
```
140718008144808
```

In [91]:
```
name = 'nit'
id(name)
```

Out[91]:
```
1913934794480
```

In [92]:
```
a = 10
id(a)
```

Out[92]:
```
140718008144968
```

In [94]:
```
b = a # thats why python is more memory efficient
```

In [95]:
```
id(b)
```

Out[95]:
```
140718008144968
```

In [96]:
```
id(10)
```

Out[96]:
```
140718008144968
```

In [97]:
```
k = 10
id(k)
```

Out[97]: 140718008144968

In [98]:
```
a = 20   # as we change the value of a address will change
id(a)
```

Out[98]: 140718008145288

In [100]:
```
id(b)
```

Out[100]: 140718008144968

In [101]:
```
PI = 3.14  # in math this is always constant but python we can change
PI
```

Out[101]: 3.14

```
PI = 3.15
PI
```

In [103]:
```
type(PI)
```

Out[103]: float

DATA TYPES & DATA STRUCTURES

```
1- NUMERIC || 2- LIST || 3- TUPLE || 4-SET || 5-STRING ||6-RANGE || 7-DICTIONARY
```

1. NUMERIC :- INT , FLOAT, COMPLEX, BOOL

In [44]:
```
w = 2.5
type(w)
```

Out[44]: float

In [46]:
```
w2 = 2 + 3j  # so hear j is reprsent as root of -1
type(w2)
```

Out[46]: complex

In [47]:
```
# convert float to integer
a = 5.6
b = int(a)
```

In [48]:
```
b
```

Out[48]: 5

In [49]:
```
type(b)
```

Out[49]: int

In [50]:
```
type(a)
```

Out[50]: float

In [52]:
```
k = float(b)
```

In [53]:
```
k
```

Out[53]:
```
5.0
```

In [54]:
```
print(a)
print(b)
print(k)
```

```
5.6
5
5.0
```

In [55]:
```
k1 = complex(b,k)
```

In [56]:
```
print(k1)
type(k1)
```

```
(5+5j)
```

Out[56]:
```
complex
```

In [57]:
```
b < k
```

Out[57]:
```
False
```

In [58]:
```
condition = b < k
condition
```

Out[58]:
```
False
```

In [59]:
```
b == k
```

Out[59]:
```
True
```

In [61]:
```
type(condition)
```

Out[61]:
```
bool
```

In [62]:
```
int(True)
```

Out[62]:
```
1
```

In [63]:
```
int(False)
```

Out[63]:
```
0
```

In [64]:
```
l = [1,2,3,4]
print(l)
type(l)
```

```
[1, 2, 3, 4]
```

Out[64]:
```
list
```

In [65]:
```
s = {1,2,3,4}
s
```

Out[65]:
```
{1, 2, 3, 4}
```

In [66]:
```
type(s)
```

Out[66]:
```
set
```

In [67]:
```
s1 = {1,2,3,4,4,3,11}   # duplicates are not allowed
s1
```

Out[67]:
```
{1, 2, 3, 4, 11}
```

In [68]:
```
t = {10,20,30}
t
```

Out[68]:
```
{10, 20, 30}
```

In [69]:
```
type(t)
```

Out[69]:
```
set
```

In [72]:
```
str = 'nit'   # we dont have character in python
type(str)
```

Out[72]:
```
str
```

In [73]:
```
st = 'n'
type(st)
```

Out[73]:
```
str
```

range()

In [31]:
```
r = range(0,10)
r
```

Out[31]:
```
range(0, 10)
```

In [32]:
```
type(r)
```

Out[32]:
```
range
```

In [33]:
```
# if you want to print the range
list(range(10,20))
```

Out[33]:
```
[10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
```

In [34]:
```
r1 = list(r)
r1
```

Out[34]:
```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [35]:
```
# if you want to print even number
even_number = list(range(2,10,2))
even_number
```

Out[35]:
```
[2, 4, 6, 8]
```

In [36]:
```
d = {1:'one',2:'two',3:'three'}
d
```

Out[36]:
```
{1: 'one', 2: 'two', 3: 'three'}
```

In [38]:
```
type(d)
```

Out[38]:
```
dict
```

In [40]:
```
# print the keys
d.keys()
```

Out[40]:
```
dict_keys([1, 2, 3])
```

In [41]:
```
d.values()
```

Out[41]:
```
dict_values(['one', 'two', 'three'])
```

In [42]:
```
d[2]
```

Out[42]:
```
'two'
```

In [43]:
```
d.get(2)
```

Out[43]:
```
'two'
```

Operator

1. Arithmetic Operator(+,-,*,/,%,%%,**,^) 2.Assignment operator(=) 3. Relational Operator 4. Logical Operator 5. Unary Operator

Arithmetic Operator

In [82]:
```
x1 , y1 = 10 , 5
```

In [83]:
```
x1 + y1
```

Out[83]:
```
15
```

In [84]:
```
x1 - y1
```

Out[84]:
```
5
```

In [85]:
```
x1 * y1
```

Out[85]:
```
50
```

In [86]:
```
x1 / y1
```

Out[86]:
```
2.0
```

In [87]:
```
x1 // y1
```

Out[87]:
```
2
```

In [88]:
```
x1 % y1
```

Out[88]:
```
0
```

In [89]:
```
x1 ** y1
```

Out[89]:
```
100000
```

In [90]:
```
x2 = 3
y2 = 3
x2 ** y2
```

Out[90]: 27

Assignment Operator

In [91]:
```
x = 2
```

In [92]:
```
x = x + 2
```

In [93]:
```
x
```

Out[93]: 4

In [94]:
```
x += 2
x
```

Out[94]: 6

In [95]:
```
x += 2
```

In [96]:
```
x
```

Out[96]: 8

In [97]:
```
x *= 2
```

In [98]:
```
x
```

Out[98]: 16

In [99]:
```
x -= 2
```

In [100]:
```
x
```

Out[100]: 14

In [101]:
```
x /= 2
```

In [102]:
```
x
```

Out[102]: 7.0

In [103]:
```
x // 2
```

Out[103]: 3.0

In [104]:
```
a, b = 5, 6
print(a)
print(b)
```

```
5
6
```

In [105]:
```
a = 5
b = 6
print(a)
print(b)
```

```
5
6
```

In [106]:
```
a
```

Out[106]: 5

In [107]:
```
b
```

Out[107]: 6

Unary Operator

In [ ]:
```
# Unary means 1  we are applying unary minus operator(-) on the operand n; the value of m becomes
```

In [108]:
```
n = 7 # negation
n
```

Out[108]: 7

In [109]:
```
m = -(n)
```

In [110]:
```
m
```

Out[110]: -7

In [111]:
```
-n
```

Out[111]: -7

Relational Operator

In [112]:
```
a = 5
b = 6
```

In [113]:
```
a < b
```

Out[113]: True

In [114]:
```
a > b
```

Out[114]: False

In [115]:
```
a == b    # a = b # we cannot use = operatro that means it is assigning
```

Out[115]: False

In [116]:
```
a != b
```

Out[116]: True

In [117]:
```
b = 5
```

In [118]:
```
a == b
```

Out[118]: True

In [119]:
```
a
```

Out[119]: 5

In [120]:
```
b
```

Out[120]: 5

In [121]:
```
a > b
```

Out[121]: False

In [122]:
```
a < b
```

Out[122]: False

In [123]:
```
a >= b
```

Out[123]: True

In [124]:
```
a <= b
```

Out[124]: True

In [125]:
```
b = 7
```

In [126]:
```
a != b
```

Out[126]: True

Logical Operator

In [127]:
```
a = 5
b = 4
```

In [128]:
```
a<8 and b< 5
```

Out[128]: True

In [129]:
```
a < 8 and b< 2
```

Out[129]: False

In [130]:
```
a < 8 or b < 2
```

Out[130]: True

In [131]:
```
a > 8 or b < 2
```

Out[131]: False

In [132]:
```
x = False
x
```

Out[132]: False

In [133]:
```
not x
```

Out[133]: True

In [134]:
```
x = not x
```

In [135]:
```
x
```

Out[135]: True

In [136]:
```
not x
```

Out[136]: False

Number System Conversion (bit-binary digit)

In [137]:
```
25
```

Out[137]: 25

In [138]:
```
bin(25)
```

Out[138]: '0b11001'

In [139]:
```
int(0b11001)
```

Out[139]: 25

In [140]:
```
bin(90)
```

Out[140]: '0b1011010'

In [141]:
```
int(0b1011010)
```

Out[141]: 90

In [142]:
```
oct(25)
```

Out[142]: '0o31'

In [143]:
```
int(0o31)
```

Out[143]: 25

In [144]:
```
0b11001
```

Out[144]: 25

In [145]:
```
int(0b11001)
```

Out[145]: 25

In [146]:
```
bin(7)
```

Out[146]: '0b111'

In [147]:
```
oct(25)
```

Out[147]: '0o31'

In [148]:
```
int(0o31)
```

Out[148]: 25

In [149]:
```
hex(25)
```

Out[149]: '0x19'

In [150]:
```
hex(16)
```

Out[150]: '0x10'

In [151]:
```
0xa
```

Out[151]: 10

In [152]:
```
0xb
```

Out[152]: 11

In [153]:
```
0xc
```

Out[153]: 12

In [154]:
```
hex(1)
```

Out[154]: '0x1'

In [155]:
```
hex(25)
```

Out[155]: '0x19'

In [156]:
```
0x19
```

Out[156]: 25

In [157]:
```
0x15
```

Out[157]: 21

Swap 2 = Variable in python

In [158]:
```
a = 5
b = 6
```

In [159]:
```
a = b
b = a
```

In [160]:
```
print(a)
print(b)
```

```
6
6
```

In [161]:
```
a1 = 7
b1 = 8
```

In [162]:
```
temp = a1      # Using thied variable
a1 = b1
b1 = temp
```

In [163]:
```
print(a1)
print(b1)
```

```
8
7
```

In [164]:
```
a2 = 5
b2 = 6
```

In [165]:
```
# swap variable formulas without using 3rd formula

a2 = a2 + b2  # 5+6=11
b2 = a2 - b2  # 11-6=5
a2 = a2 -  b2 # 11-5=6
```

In [166]:
```
print(a2)
print(b2)
```

```
6
5
```

In [167]:
```
0b110
```

Out[167]:
```
6
```

In [168]:
```
0b101
```

Out[168]:
```
5
```

In [169]:
```
print(0b110)
print(0b101)
```

```
6
5
```

In [170]:
```
print(0b101)
print(0b110)
```

```
5
6
```

In [171]:
```
#but when we use a2 + b2 then we get 11 that means we will get 4 bit which is 1 bit extra
print(bin(11))
print(0b1011)
```

```
0b1011
11
```

In [172]:
```
# XOR

print(a2)
print(b2)
```

```
6
5
```

In [173]:
```
#there is other way to work using swap variable also which is XOR because it will not waste extra
a2 = a2 ^ b2
b2 = a2 ^ b2
a2 = a2 ^ b2
```

In [174]:
```
print(a2,b2)
```

```
5 6
```

In [175]:
```
print(bin(12))
print(bin(13))
```

```
0b1100
0b1101
```

In [176]:
```
0b1101
```

Out[176]: 13

In [177]:
```
0b1100
```

Out[177]: 12

In [178]:
```
~12
```

Out[178]: -13

In [179]:
```
~46
```

Out[179]: -47

In [180]:
```
~54
```

Out[180]: -55

In [181]:
```
~10
```

Out[181]: -11

In [182]:
```
# bitwise and operator

#AND - Logical Operator ||| & - Bitwise AND operator
```

In [184]:
```
12 & 13
```

Out[184]: 12

In [1]:
```
12 | 13
```

Out[1]: 13

In [2]:
```
1 & 0
```

Out[2]: 0

In [3]:
```
1 | 0
```

Out[3]: 1

In [4]:
```
bin(13)
```

Out[4]: '0b1101'

In [5]:
```
print(bin(35))
print(bin(40))
```

```
0b100011
0b101000
```

In [6]:
```
35 & 40
```

Out[6]: 32

In [7]:
```
35 | 40
```

Out[7]: 43

In [8]:
```
12 ^ 13
```

Out[8]: 1

In [9]:
```
print(bin(25))
print(bin(30))
```

```
0b11001
0b11110
```

In [10]:
```
25^30
```

Out[10]: 7

In [11]:
```
bin(7)
```

Out[11]: '0b111'

In [12]:
```
bin(25)
```

Out[12]: '0b11001'

In [13]:
```
bin(30)
```

Out[13]: '0b11110'

In [14]:
```
0b00111
```

Out[14]: 7

In [15]:
```
bin(10)
```

Out[15]: '0b1010'

In [16]:
```
10<<1
```

Out[16]: 20

In [17]:
```
10<<2
```

Out[17]: 40

In [19]:
```
# BIT WISE LEFT SHIFT OPERATOR
# in left shift what we need to to we need shift in left hand side & need to shift 2 bits
#bit wise left operator bydefault you will take 2 zeros ( )
#10 binary operator is 1010 | also i can say 1010
10<<2
```

Out[19]: 40

In [20]:
```
10<<3
```

Out[20]: 80

In [21]:
```
bin(20)
```

Out[21]: '0b10100'

In [22]:
```
20<<4
```

Out[22]: 320

In [23]:
```
# Bitwise Rightshift operator
```

```
1. Left side we are gaining the bits
2. right side we are lossing bits
```

In [26]:
```
bin(10)
```

Out[26]: '0b1010'

In [27]:
```
10>>1
```

Out[27]: 5

In [28]:
```
10>>2
```

Out[28]: 2

In [29]:
```
10>>3
```

Out[29]: 1

In [30]:
```
bin(20)
```

Out[30]: '0b10100'

In [ ]: