In [3]:
```
1+1 # Addition
```

Out[3]: 2

In [4]:
```
2-1
```

Out[4]: 1

In [5]:
```
3*4
```

Out[5]: 12

In [6]:
```
8/4 # Division
```

Out[6]: 2.0

In [7]:
```
8/5 # Float division
```

Out[7]: 1.6

In [8]:
```
8//4 # Integer Division
```

Out[8]: 2

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

Out[9]: 10

In [10]:
```
8+8 - #Syntax error:
```

```
  Cell In[10], line 1
    8+8 - #Syntax error:
          ^
SyntaxError: invalid syntax
```

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

In [ ]:
```
(5+5)*5 #BODMAS
```

In [ ]:
```
2*2*2*2*2 # Exponention
```

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

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

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

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

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

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

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

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

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

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

In [ ]:
```
type(c)
```

In [ ]:
```
'Naresh IT'
```

In [ ]:
```
print('Max it')
```

In [ ]:
```
"max it technology"
```

In [ ]:
```
s1 = 'max it technology'
s1
```

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

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

In [ ]:
```
a = 3
b = 'hi'
type(b)
```

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

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

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

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

In [ ]:
```
# print the nit 2 times
'nit' + ' nit'
```

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

In [ ]:
```
#5 time print
5 * 'nit'
```

In [ ]:
```
5*' nit' # soace between words
```

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

In [ ]:
```
print(r'c:\nit') #raw string
```

Variable|| Identifier|| Object

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

Out[13]: 2

In [14]:
```
x = 2 #x is variable/identifier/objec, 2 is the value
x
```

Out[14]: 2

In [15]:
```
x + 3
```

Out[15]: 5

In [16]:
```
y = 3
y
```

Out[16]: 3

In [17]:
```
x + y
```

Out[17]: 5

In [18]:
```
x = 9
x
```

Out[18]: 9

In [19]:
```
x + y
```

Out[19]: 12

In [20]:
```
x + 10
```

Out[20]: 19

In [21]:
```
_ + y # _ understand the previous result of the
```

Out[21]: 22

In [22]:
```
_ + y
```

Out[22]: 25

In [23]:
```
_ + y
```

Out[23]: 28

In [24]:
```
_ + y
```

Out[24]: 31

In [25]:
```
y
```

Out[25]: 3

In [26]:
```
_+y
```

Out[26]: 6

In [27]:
```
_+y
```

Out[27]: 9

In [28]:
```
_+y
```

Out[28]: 12

In [29]:
```
# string variable
name = 'mit'
```

In [30]:
```
name
```

Out[30]: 'mit'

In [31]:
```
name + 'technology'
```

Out[31]: 'mittechnology'

In [32]:
```
name + ' technology'
```

Out[32]: 'mit technology'

In [33]:
```
name 'technology'
```

```
  Cell In[33], line 1
    name 'technology'
         ^
SyntaxError: invalid syntax
```

In [34]:
```
name
```

Out[34]: 'mit'

In [35]:
```
len(name)
```

Out[35]: 3

In [36]:
```
name[0] #python index begins with 0
```

Out[36]: 'm'

In [37]:
```
name[5]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[37], line 1
----> 1 name[5]

IndexError: string index out of range
```

In [38]:
```
name[7]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[38], line 1
----> 1 name[7]

IndexError: string index out of range
```

In [39]:
```
name[-1]
```

Out[39]:
```
't'
```

In [40]:
```
name[-2]
```

Out[40]:
```
'i'
```

In [41]:
```
name[-6]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[41], line 1
----> 1 name[-6]

IndexError: string index out of range
```

slicing

In [42]:
```
name
```

Out[42]:
```
'mit'
```

In [43]:
```
name[0:1] #to print 2 character
```

Out[43]:
```
'm'
```

In [44]:
```
name[0:2]
```

Out[44]:
```
'mi'
```

In [45]:
```
name[1:4]
```

Out[45]:
```
'it'
```

In [46]:
```
name[1:]
```

Out[46]:
```
'it'
```

In [47]:
```
name[:4]
```

Out[47]: 'mit'

In [48]:
```
name[3:9]
```

Out[48]: ''

In [49]:
```
name[4:9]
```

Out[49]: ''

In [50]:
```
name
```

Out[50]: 'mit'

In [51]:
```
name1 = 'fine' # change the string fine to dine
name1
```

Out[51]: 'fine'

In [52]:
```
name1[0:1]
```

Out[52]: 'f'

In [53]:
```
name1[0:1] = 'd' # i want to change 1st character of naresh (n) - t
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[53], line 1
----> 1 name1[0:1] = 'd'

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

In [54]:
```
name1[0] = 'd' #strings in python are immutable
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[54], line 1
----> 1 name1[0] = 'd'

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

In [55]:
```
name1
```

Out[55]: 'fine'

In [56]:
```
name1[1:]
```

Out[56]: 'ine'

In [59]:
```
x= 'd' + name1[1:] #i want to change fine to dine
x
```

Out[59]: 'dine'

In [61]:
```
name1.insert(2,'nit') #insert the value as per index values i.e 2nd index we are
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[61], line 1
----> 1 name1.insert(2,'nit')

AttributeError: 'str' object has no attribute 'insert'
```

Introduce to ID()

In [62]:
```
# variable address
num = 5
id(num)
```

Out[62]: 140722039499320

In [63]:
```
name = 'nit'
id(name) #Address will be different for both
```

Out[63]: 2691567032464

In [64]:
```
a = 10
id(a)
```

Out[64]: 140722039499480

In [65]:
```
b = a #thats why python is more memory efficient
```

In [66]:
```
id(b)
```

Out[66]: 140722039499480

In [67]:
```
id(10)
```

Out[67]: 140722039499480

In [68]:
```
k = 10
id(k)
```

Out[68]: 140722039499480

In [69]:
```
a = 20  # as we change the value of a then address will change
id(a)
```

Out[69]: 140722039499800

In [70]:
```
id(b)
```

Out[70]: 140722039499480

In [71]:
```
PI = 3.14  #in math this is alway constant but python we can chang
PI
```

Out[71]: 3.14

In [72]:
```
PI = 3.15
PI
```

Out[72]: 3.15

In [73]:
```
type(PI)
```

Out[73]: float

Arithmetic operator

In [76]:
```
x1, y1 = 10, 5
```

In [78]:
```
print(bin(x1))
print(bin(y1))
```

```
0b1010
0b101
```

In [79]:
```
x1 ^ y1
```

Out[79]: 15

In [80]:
```
x1  + y1
```

Out[80]: 15

In [81]:
```
x1 - y1
```

Out[81]: 5

In [82]:
```
x1 * y1
```

Out[82]: 50

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

Out[83]: 2.0

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

Out[84]: 2

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

Out[85]: 0

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

Out[86]: 100000

In [87]:
```
x2 = 3
y2 = 2
x2 ** y2
```

Out[87]: 9

Assignment operator

In [108…
```
x = 2
```

In [109…
```
x = x + 2 # if you want to increment by 2
```

In [110…
```
x
```

Out[110… 4

In [111…
```
x += 2
x
```

Out[111… 6

In [112…
```
x += 2
x
```

Out[112… 8

In [113…
```
x *= 2
x
```

Out[113… 16

In [114…
```
x
```

Out[114… 16

In [115…
```
x -= 2
```

In [116…
```
x
```

Out[116… 14

In [117…
```
x/=2
x
```

Out[117… 7.0

In [118…
```
x//=2
x
```

Out[118… 3.0

In [119…
```
a,b=5,6
```

In [120…
```
a
```

Out[120… 5

In [121…
```
b
```

Out[121… 6

Unary operator

In [126…
```
n = 7 #negattion
n
```

Out[126… 7

In [127…
```
m = -(n)
m
```

Out[127… -7

In [128…
```
n
```

Out[128… 7

In [129…
```
-n
```

Out[129… -7

Relational operator

In [130…
```
a = 5
b = 6
```

In [131…
```
a<b
```

Out[131… True

In [132…
```
a>b
```

Out[132… False

In [133…
```
# a = b # we cannot use = operatro that means it is assigning
a == b
```

Out[133… False

In [134…
```
a != b
```

Out[134… True

In [135…
```
# hear if i change b = 6
b = 5
```

In [136…
```
a == b
```

Out[136… True

In [137…
```
a
```

Out[137… 5

In [138…
```
b
```

Out[138… 5

In [139…
```
a >= b
```

Out[139… True

In [140…
```
a <= b
```

Out[140… True

In [141…
```
a < b
```

Out[141… False

In [142…
```
a>b
```

Out[142… False

In [143…
```
b = 7
```

In [144…
```
a != b
```

Out[144… True

LOGICAL OPERATOR

In [145…
```
a = 5
b = 4
```

In [146…
```
a < 8 and b < 5 #refer to the truth table
```

Out[146… True

In [147…
```
a < 8 and b < 2
```

Out[147… False

In [148…
```
a < 8 or b < 2
```

Out[148… True

In [149…
```
a>8 or b<2
```

Out[149… False

In [150…
```
x = False
x
```

Out[150… False

In [151…
```
not x
```

Out[151…
```
True
```