Python Basics

```
In [13]: 1 + 1 # ADDITION

Out[13]: 2

In [14]: 2-1 # subtraction

Out[14]: 1

In [15]: 3*4 #Multiplication

Out[15]: 12

In [16]: 4/2 #float division

Out[16]: 2.0

In [18]: 10//2  #integer division

Out[18]: 5

In [19]: 8 + 9 - 7

Out[19]: 10

In [22]: 8 + 8 - #syntax error

  Cell In[22], line 1
    8 + 8 - #syntax error
            ^
SyntaxError: invalid syntax

In [23]: 5 + 5 * 5

Out[23]: 30

In [24]: (5 + 5) * 5 # BODMAS (Bracket || Oders || Divide || Multiply || Add || Substact)

Out[24]: 50

In [25]: 2 * 2 * 2 * 2 * 2 # exponentaion

Out[25]: 32

In [28]: 2**5 # exponentation

Out[28]: 32

In [29]: 2*5
```

```
Out[29]:  10

In [30]:  2**5

Out[30]:  32

In [31]:  15 % 2 # Modulus

Out[31]:  1

In [32]:  10 % 2

Out[32]:  0

In [33]:  15 %% 2

  Cell In[33], line 1
    15 %% 2
        ^
SyntaxError: invalid syntax

In [35]:  3 + 'nit' # You can not add integer with string

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[35], line 1
----> 1 3 + 'nit'

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

In [36]:  3*'nit'

Out[36]:  'nitnitnit'

In [37]:  3*' nit'

Out[37]:  ' nit nit nit'

In [38]:  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 [39]:  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 [40]:
```
'Naresh IT'
```

Out[40]:
```
'Naresh IT'
```

In [41]:
```
print('Max it')
```

```
Max it
```

In [42]:
```
"max it technology"
```

Out[42]:
```
'max it technology'
```

In [44]:
```
s1 = 'max it technology'
s1
```

Out[44]:
```
'max it technology'
```

In [45]:
```
a = 2
b = 3
a + b
```

Out[45]:
```
5
```

In [46]:
```
c = a+b
c
```

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

In [47]:
```
a = 3
b = 'hi'
type(b)
```

Out[47]:
```
str
```

In [50]:
```
print('It's Naresh IT Technology")
```

```
  Cell In[50], line 1
    print('It's Naresh IT Technology")
                                      ^
SyntaxError: unterminated string literal (detected at line 1)
```

In [51]:
```
print('max it's"Technology"')
```

```
  Cell In[51], line 1
    print('max it's"Technology"')
                              ^
SyntaxError: unterminated string literal (detected at line 1)
```

```
In [53]: print('It\'s Naresh IT Technology')#\ has some special meaning to ignore the error
```

```
It's Naresh IT Technology
```

```
In [54]: print('max it',  'Technology')
```

```
max it Technology
```

```
In [55]: print("max it',  'Technology")
```

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

```
In [57]: # print the name 2 times
         'name' + ' name'
```

```
Out[57]: 'name name'
```

```
In [59]: 'name' ' name'
```

```
Out[59]: 'name name'
```

```
In [60]: 'amruta'
```

```
Out[60]: 'amruta'
```

```
In [61]: #print amruta 5 times
         5*'amruta'
```

```
Out[61]: 'amrutaamrutaamrutaamrutaamruta'
```

```
In [62]: #print amruta 5 times with space
         5*' amruta'
```

```
Out[62]: ' amruta amruta amruta amruta amruta'
```

```
In [63]: print('c:\nit') #\n -- new line
```

```
c:
it
```

```
In [64]: print(r'c:\nit') #raw string
```

```
c:\nit
```

Variable || Identifier || Object

```
In [65]: 5
```

```
Out[65]: 5
```

In [68]:
```
x = 10 #x is variable/identifier/objec, 10 is the value
x
```
Out[68]:
```
10
```
In [69]:
```
x + 3
```
Out[69]:
```
13
```
In [70]:
```
y = 3
y
```
Out[70]:
```
3
```
In [71]:
```
x = 9
x
```
Out[71]:
```
9
```
In [72]:
```
x + y
```
Out[72]:
```
12
```
In [74]:
```
_+ y # _ understand the previous result of the
```
Out[74]:
```
18
```
In [75]:
```
_+y
```
Out[75]:
```
21
```
In [76]:
```
_ + y
```
Out[76]:
```
24
```
In [78]:
```
# string variable
name = 'madras'
name
```
Out[78]:
```
'madras'
```
In [79]:
```
name + 'technology'
```
Out[79]:
```
'madrastechnology'
```
In [81]:
```
name + ' technology'
```
Out[81]:
```
'madras technology'
```
In [82]:
```
name 'technology'
```

```
  Cell In[82], line 1
    name 'technology'
         ^
SyntaxError: invalid syntax
```

```
In [83]: name , 'technology'
```

```
Out[83]: ('madras', 'technology')
```

```
In [84]: len(name)
```

```
Out[84]: 6
```

```
In [85]: name[0] #python index begins with 0
```

```
Out[85]: 'm'
```

```
In [86]: name[6] #python index begins with 0
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[86], line 1
----> 1 name[6]

IndexError: string index out of range
```

```
In [87]: name[5] #python index begins with 0
```

```
Out[87]: 's'
```

```
In [88]: name[2] #python index begins with 0
```

```
Out[88]: 'd'
```

```
In [89]: name[-1]
```

```
Out[89]: 's'
```

```
In [90]: name[-2]
```

```
Out[90]: 'a'
```

```
In [91]: name[-6]
```

```
Out[91]: 'm'
```

Slicing

```
In [92]: name
```

```
Out[92]: 'madras'
```

```
In [93]: name[0:1] #to print 2 character

Out[93]: 'm'

In [94]: name[0:2]

Out[94]: 'ma'

In [95]: name[1:4]

Out[95]: 'adr'

In [96]: name[1:]

Out[96]: 'adras'

In [97]: name[:4]

Out[97]: 'madr'

In [98]: name[3:9]

Out[98]: 'ras'
```

Convert String Fine to Dine

```
In [99]: name = 'Fine'
         name

Out[99]: 'Fine'

In [100… name[1:]

Out[100… 'ine'

In [101… 'D' + name[1:] # Replace F letter with D

Out[101… 'Dine'

In [102… name1 = 'D' + name[1:]
          name1

Out[102… 'Dine'

In [103… print(name)
          print(name1)

Fine
Dine
```

In [104…

```
name = 'Fine'
name[1:]
name1 = 'D' + name[1:]
print(name)
print(name1)
```

```
Fine
Dine
```

In [112…

```
## Convert String Glass to Grass
str = 'Glass'
str
str[0]
str[1]
str[2:]
str1 = str[0] + 'r' + str[2:]
print(str)
print(str1)
```

```
Glass
Grass
```

In [114…

```
# Convert string Change to Range

str2 = 'change'
str[0:2]
str[2:]

str3 = 'R' + str2[2:]

print(str2)
print(str3)
```

```
change
Range
```

In [125…

```
num1 = [1,2,'Accademy',50]
```

In [126…

```
num1.insert(3,'nit') #insert the value as per index values i.e 2nd index we are ass
num1
```

Out[126…

```
[1, 2, 'Accademy', 'nit', 50]
```

In [124…

```
num1.clear()
num1
```

Out[124…

```
[]
```

ID

In [127…

```
# variable address
num = 5
id(num)
```

```
Out[127…  2663796506992
```

```
In [128…  name = 'nit'
          id(name) #Address will be different for both
```

```
Out[128…  2663846873968
```

```
In [129…  a = 10
          id(a)
```

```
Out[129…  2663796507152
```

```
In [133…  b = a #thats why python is more memory efficient
          print(id(a))
          print(id(b))
          print(id(k))
```

```
2663796507152
2663796507152
2663796507152
```

```
In [132…  k = 10
          id(k)
```

```
Out[132…  2663796507152
```

```
In [134…  a = 20  # as we change the value of a then address will change
          id(a)
```

```
Out[134…  2663796507472
```

```
In [135…  id(b)
```

```
Out[135…  2663796507152
```

- what ever the variale we assigned the memory and we not assigned anywhere then we can use as garbage collection.
- VARIABLE - we can change the values
- ONSTANT - we cannot change the value -can we make VARIABLE as a CONSTANT
- **(note - in python you cannot make variable as constant)**

```
In [136…  PI = 3.14  #in math this is alway constant but python we can chang
          PI
```

```
Out[136…  3.14
```

```
In [137…  PI = 3.15  #in math this is alway constant but python we can chang
          PI
```

```
Out[137…  3.15
```

In [138…
```
type(PI)
```
Out[138…
```
float
```

Operators

- 1- ARITHMETIC OPERATOR (+ , -, , /, %, %%, *, ^
- 2- ASSIGNMEN OPERATOR (=)
- 3- RELATIONAL OPERATOR (>,<,>=,<=,!=)
- 4- LOGICAL OPERATOR (AND,OR,NOT)
- 5- UNARY OPERATOR ()

Arithmetic operator

In [140…
```
x1, y1 = 10, 5
print(x1)
print(y1)
```
```
10
5
```

In [145…
```
print(x1  + y1) # Addition
print(x1 - y1)  # Subtraction
print(x1 * y1) # Multiplication
print(x1 / y1)
print(x1 // y1)
print(x1 % y1)
print(x1 ** y1) # Expontial
```
```
15
5
50
2.0
2
0
100000
```

In [146…
```
x2 = 3
y2 = 2
x2 ** y2
```
Out[146…
```
9
```

Assignment operator

In [147…
```
x = 2
```

In [148…
```
x = x + 2 # if you want to increment by 2
```

```
x
```

Out[148… 4

In [149…
```
x += 2
x
```

Out[149… 6

In [150…
```
x *= 2
x
```

Out[150… 12

In [151…
```
x -=2
x
```

Out[151… 10

In [152…
```
x /= 2
x
```

Out[152… 5.0

In [153…
```
x //= 2
x
```

Out[153… 2.0

unary operator

- unary means 1 || binary means 2
-

Here we are applying unary minus operator(-) on the operand n; the value of m becomes -7, which indicates it as a negative value.

In [154…
```
n = 7 #negattion
n
```

Out[154… 7

In [155…
```
m = -(n)
m
```

Out[155… -7

In [156…
```
print(n)
print(-n)
```

```
7
-7
```

Relational operator

we are using this operator for comparing

In [162…
```
a = 5
b = 6
```

In [158…
```
a<b
```

Out[158…
```
True
```

In [159…
```
a>b
```

Out[159…
```
False
```

In [160…
```
# a = b # we cannot use = operatro that means it is assigning
```

In [163…
```
a==b
```

Out[163…
```
False
```

In [164…
```
a != b
```

Out[164…
```
True
```

In [165…
```
# hear if i change b = 6
b = 5
```

In [166…
```
a == b
```

Out[166…
```
True
```

In [167…
```
a >= b
```

Out[167…
```
True
```

In [168…
```
a <= b
```

Out[168…
```
True
```

In [169…
```
a < b
```

Out[169…
```
False
```

In [170…
```
a>b
```

Out[170…
```
False
```

In [171…
```
b = 7
```

In [172…
```
a != b
```

Out[172…
```
True
```

LOGICAL OPERATOR

- logical operator you need to understand about true & false table image.png
- 3 importand part of logical operator is --> AND, OR, NOT

In [173…
```
a = 5
b = 4
```

In [174…
```
a < 8 and b < 5 #refer to the truth table
```

Out[174…
```
True
```

In [175…
```
a < 8 and b < 2
```

Out[175…
```
False
```

In [176…
```
a < 8 or b < 2
```

Out[176…
```
True
```

In [177…
```
a>8 or b<2
```

Out[177…
```
False
```

In [178…
```
x = False
x
```

Out[178…
```
False
```

In [179…
```
not x  # you can reverse the operation
```

Out[179…
```
True
```

Swipe 2 Variables

In [180…
```
# Without using thord variable
a =5
b =6
print(a)
print(b)

a,b = b,a
```

```
print(a)
print(b)
```

```
5
6
6
5
```

In [181…

```
# Using third Variable
x=3
y=5
print(x)
print(y)
temp = x
x = y
y = temp
print(x)
print(y)
```

```
3
5
5
3
```

29th Jan

In [1]:

```
25
```

Out[1]: `25`

In [2]:

```
bin(25)
```

Out[2]: `'0b11001'`

In [3]:

```
bin(35)
```

Out[3]: `'0b100011'`

In [4]:

```
oct(25)
```

Out[4]: `'0o31'`

In [5]:

```
bin(7)
```

Out[5]: `'0b111'`

In [6]:

```
hex(256)
```

Out[6]: `'0x100'`

In [10]:

```
int(0x100)
```

Out[10]: `256`

In [7]:
```
0xa
```
Out[7]:
```
10
```
In [8]:
```
0xb
```
Out[8]:
```
11
```
In [9]:
```
0xc
```
Out[9]:
```
12
```
In [11]:
```
hex(9)
```
Out[11]:
```
'0x9'
```

List Data STructure

In [12]:
```

```
```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[12], line 1
----> 1 l

NameError: name 'l' is not defined
```
In [13]:
```
l = [10,10,20,30,89]
l
```
Out[13]:
```
[10, 10, 20, 30, 89]
```
In [ ]:
```
l.
```

29th Jan 2025

Bit Wise Number System

image discription

image discription

image discription

In [1]:
```
25
```
Out[1]:
```
25
```

```
In [4]: bin(25) # Binary meand base 2
Out[4]: '0b11001'

In [3]: int(0b11001)
Out[3]: 25

In [5]: bin(35)
Out[5]: '0b100011'

In [6]: int(0b100011)
Out[6]: 35

In [7]: oct(25)
Out[7]: '0o31'

In [8]: int(0o31)
Out[8]: 25

In [9]: bin(7)
Out[9]: '0b111'

In [11]: int(0b111)
Out[11]: 7

In [12]: hex(10)
Out[12]: '0xa'

In [13]: 0xa
Out[13]: 10

In [14]: 0xb
Out[14]: 11

In [15]: hex(256)
Out[15]: '0x100'

In [17]: int(0x100)
Out[17]: 256
```

```
In [18]: int(0x100)

Out[18]: 256
```