* PYTHON *

#Variable

```
In [2]: 1
Out[2]: 1

In [3]: 2
Out[3]: 2

In [4]: 1+3   #addition
Out[4]: 4

In [5]: 2 * 4   #Multiplication
Out[5]: 8

In [6]: 9 - 4     #subtraction
Out[6]: 5

In [7]: 25/4        #division
Out[7]: 6.25

In [8]: 25 // 4   #integer division
Out[8]: 6

In [9]: 3 ** 5    #exponential
Out[9]: 243

In [10]: 2 * ( 3 + 4 ) - 8        #BODMAS rule
Out[10]: 6

In [11]: 28 % 4   #Modulus
Out[11]: 0

In [12]: x = 2        # x is variable/object/identifier, 2 is the value
         x
Out[12]: 2

In [13]: x + 5
```

```
Out[13]:  7

In [14]:  x + 9

Out[14]:  11

In [15]:  y = 4
          y

Out[15]:  4

In [16]:  x + y

Out[16]:  6

In [17]:  x = 8
          x

Out[17]:  8

In [18]:  x + y

Out[18]:  12

In [19]:  x + 10

Out[19]:  18

In [20]:  y

Out[20]:  4

In [24]:  y = 6
          y

Out[24]:  6

In [26]:  name = 'Nit'    #string variable
          name

Out[26]:  'Nit'

In [27]:  name + 'technology'    #concatenate

Out[27]:  'Nittechnology'

In [28]:  name + ' technology'

Out[28]:  'Nit technology'

In [29]:  name

Out[29]:  'Nit'

In [35]:  len(name)     # gives length of string
```

```
Out[35]: 3
```

```
In [36]: com = 2 + 3j    #complex varable
         com
```

```
Out[36]: (2+3j)
```

```
In [37]: f = 34.9     #float variable
         f
```

```
Out[37]: 34.9
```

```
In [44]: b = True
         b
```

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

```
In [46]: import keyword
         keyword.kwlist
```

```
Out[46]: ['False',
          'None',
          'True',
          'and',
          'as',
          'assert',
          'async',
          'await',
          'break',
          'class',
          'continue',
          'def',
          'del',
          'elif',
          'else',
          'except',
          'finally',
          'for',
          'from',
          'global',
          'if',
          'import',
          'in',
          'is',
          'lambda',
          'nonlocal',
          'not',
          'or',
          'pass',
          'raise',
          'return',
          'try',
          'while',
          'with',
          'yield']
```

```
In [48]: len(keyword.kwlist)
```

```
Out[48]:  35
```

#Datatypes

```
In [ ]:  # *int  *float  *string  *bool  *complex

In [38]:  type(x)

Out[38]:  int

In [39]:  type(name)

Out[39]:  str

In [40]:  type(com)

Out[40]:  complex

In [41]:  type(f)

Out[41]:  float

In [43]:  type(b)

Out[43]:  bool

In [49]:  i = 34
          f = 112.56
          print(i)
          print(f)

         34
         112.56

In [50]:  i + f

Out[50]:  146.56

In [51]:  i - f

Out[51]:  -78.56

In [52]:  i * f

Out[52]:  3827.04

In [53]:  #bool
          True

Out[53]:  True

In [54]:  False

Out[54]:  False
```

```
In [55]: True + True

Out[55]: 2

In [59]: True + False     #by default True =1 and False =0

Out[59]: 1

In [57]: False + False

Out[57]: 0

In [58]: False + True

Out[58]: 1

In [60]: True / True

Out[60]: 1.0

In [61]: True // True

Out[61]: 1

In [63]: False / True

Out[63]: 0.0

In [64]: s = 'Hello'      #string
         s

Out[64]: 'Hello'

In [65]: s1 = "Hello"
         s1

Out[65]: 'Hello'

In [66]: s2 = '''Hello'''
         s2

Out[66]: 'Hello'

In [69]: s3 = '''Hello               # use triple quot
               Team'''
         s3

Out[69]: 'Hello \n           Team'

In [70]: c = 10 + 20j      # complex
         c

Out[70]: (10+20j)

In [74]: c.real
```

```
Out[74]:  10.0

In [75]:  c.imag

Out[75]:  20.0

In [76]:  c = 10 + 20j
          d = 20 + 50j
          print(c + d)

         (30+70j)

In [77]:  print(c - d)

         (-10-30j)

In [78]:  print(c * d)

         (-800+900j)

In [79]:  print(c / d)

         (0.41379310344827586-0.034482758620689655j)
```

#Type Casting ==Convert one datatype to other datatype

```
In [ ]:   # ----integer

In [80]:  int(35.67)   # only one parameter

Out[80]:  35

In [81]:  int(True)

Out[81]:  1

In [82]:  int(False)

Out[82]:  0

In [ ]:   int(true)     # give error because of case sensitve

In [83]:  int('10')

Out[83]:  10

In [ ]:   int(3 + 2j)     # give error

In [85]:  # -----Float

In [86]:  float(305000)

Out[86]:  305000.0
```

```
In [ ]:  float(30,40)    #error because at most one argument
```

```
In [ ]:  float(2+3j)    #error argument must be a string or a real no.  ,not complex
```

```
In [87]: float(True)
```

```
Out[87]: 1.0
```

```
In [88]: float(False)
```

```
Out[88]: 0.0
```

```
In [89]: float('4')
```

```
Out[89]: 4.0
```

```
In [ ]:  float('four')    #error
```

```
In [90]: # -----string
```

```
In [91]: str(8)
```

```
Out[91]: '8'
```

```
In [92]: str(7.8)
```

```
Out[92]: '7.8'
```

```
In [94]: str(4 + 7j)
```

```
Out[94]: '(4+7j)'
```

```
In [ ]:  str(True,False)    # error must be string not bool
```

```
In [95]: str(True)
```

```
Out[95]: 'True'
```

```
In [96]: str()
```

```
Out[96]: ''
```

```
In [97]: # ----bool
```

```
In [98]: bool(10)    # non-zero value return True
```

```
Out[98]: True
```

```
In [99]: bool(3.4)
```

```
Out[99]: True
```

```
In [100…  bool()
```

```
Out[100… False

In [101… bool('45')

Out[101… True

In [102… bool('Ten')

Out[102… True

In [103… bool(0)

Out[103… False

In [104… bool('0')

Out[104… True

In [105… bool(3 + 5j)

Out[105… True

In [106… # ----complex

In [107… complex(10)

Out[107… (10+0j)

In [108… complex(20,30)

Out[108… (20+30j)

  In [ ]: complex(20,10,40)      # give error

In [109… complex(10.3,56.3)

Out[109… (10.3+56.3j)

In [110… complex(True)

Out[110… (1+0j)

In [111… complex(False)

Out[111… 0j

In [112… complex('10')

Out[112… (10+0j)

  In [ ]: complex('10','20')      # error if first is string then it can't take other

In [114… complex(True,False)

Out[114… (1+0j)
```

In [115…
```
complex(False,True)
```
Out[115…
```
1j
```

#String Indexing

In [116…
```
# Forward indexing ----left to right(start from 0)
# Backward indexing ---right to left(start from -1)
```
In [117…
```
s = 'Python'
s
```
Out[117…
```
'Python'
```
In [118…
```
s[0]
```
Out[118…
```
'P'
```
In [119…
```
s[2]
```
Out[119…
```
't'
```
In [121…
```
s[5]
```
Out[121…
```
'n'
```
In [122…
```
s[-1]
```
Out[122…
```
'n'
```
In [123…
```
s[-2]
```
Out[123…
```
'o'
```
In [124…
```
s[-6]
```
Out[124…
```
'P'
```
In [125…
```
str = 'world'
str
```
Out[125…
```
'world'
```
In [126…
```
print(str[0])
print(str[1])
print(str[2])
print(str[3])
print(str[4])
```
```
w
o
r
l
d
```

In [127…
```
print(str[-1])
print(str[-2])
print(str[-3])
print(str[-4])
print(str[-5])
```
```
d
l
r
o
w
```

In [128…
```
len(str)
```
Out[128…
```
5
```

#string Slicing,id(),len()

In [129…
```
str = 'HelloPython'
str
```
Out[129…
```
'HelloPython'
```

In [130…
```
s = 'hello'
s
```
Out[130…
```
'hello'
```

In [131…
```
s[:]       # print all elements
```
Out[131…
```
'hello'
```

In [132…
```
s[0:4]     # print elements from 0 to (4-1) means last (n-1)
```
Out[132…
```
'hell'
```

In [133…
```
s[1:4]
```
Out[133…
```
'ell'
```

In [134…
```
s[4:4]
```
Out[134…
```
''
```

In [137…
```
str
```
Out[137…
```
'HelloPython'
```

In [138…
```
str[0:6]
```
Out[138…
```
'HelloP'
```

In [139…
```
s[2:]      #start from 2 till last
```

```
Out[139… 'llo'

In [140… s[:4]      #start from 0 to 3rd index

Out[140… 'hell'

In [141… str

Out[141… 'HelloPython'

In [142… str[::3]    #print start to last index print only 3rd index

Out[142… 'Hlyo'

In [143… str[::-1]   #print the element in reverse

Out[143… 'nohtyPolleH'

In [144… s[::-1]

Out[144… 'olleh'

In [145… str[::-2]      # print the element in reverse -2 index position

Out[145… 'nhyolH'

In [146… s[::-2]

Out[146… 'olh'

In [147… str

Out[147… 'HelloPython'

In [148… str[0:10:2]    #print index 0 to 9th index but print every 2nd index count skip

Out[148… 'Hloyh'

In [149… str[1:8:2]

Out[149… 'elPt'

In [150… str

Out[150… 'HelloPython'

  In [ ]: str[0] = 'd'    #give error string in python are immutable

In [153… len(str)

Out[153… 11

In [154… # ID() ---- variable address
```

In [155…
```
num = 5
id(num)
```
Out[155…
```
140707912528936
```
In [156…
```
name ='nit'
id(name)
```
Out[156…
```
1470582401920
```
In [157…
```
a = 10
id(num)
```
Out[157…
```
140707912528936
```
In [158…
```
b = a
id(b)
```
Out[158…
```
140707912529096
```
In [160…
```
k =13
id(k)
```
Out[160…
```
140707912529192
```
In [161…
```
a = 20
id(a)     #as we change value of a then address will change
```
Out[161…
```
140707912529416
```
In [ ]:
```

```