```
In [ ]: Generally we have
```

```
In [ ]: ################## Level-1#################
1) basic syntax
2) First program
3) conditional statements
4) functions
5) loops concept
6) Expetional handling
7) packages

############### Level-2####################
8)Strings
9)List
10)dictionary
11)tuples
12)sets
13) file handling sessions

############## Level-3#################
14)OOPS concept
15)EDA
16)ML
17)DL
18)NLP
19) FLASK
```

- integer type
- float
- string
- boolean
- complex conjugate

$Integer\text{-}type$:

```
In [1]: a=100
type(a)
```

```
Out[1]: int
```

in math different number system will avialable

- binary
- decimal
- octal
- hexa

binary

- bi mean two : base=2
- base=2 , two digits required

- the two digits are 0 and 1
- The representation is 0b
- example: 0b10001 or 0B10001

In [2]:
```
0b111  # here the input is binary format

# the output is decimal format
```

Out[2]:
```
7
```

In [3]:
```
0b101
```

Out[3]:
```
5
```

In [ ]:
```
# STLD  DLD

2^2    2^1     2^0

4       2       1
0       0       0==========0
0       0       1==========1
0       1       0==========2
0       1       1==========3
1       0       0==========4
1       0       1==========5
1       1       0==========6
1       1       1==========7

ON=1   OFF=0
```

In [ ]:
```
0B1111

8   4   2    1
1   1   1    1==== 15

1011

8 + 2+1=11
```

In [4]:
```
# THREE DIGITS(2^2) ARE THERE:       4  2  1
# FOUR DIGITS(2^3) ARE THERE:     8  4  2  1
# Five digits(2^4) are there:  16 8  4  2  1
# Six digits(2^5) are there:32 16 8  4  2  1

0b11101
```

Out[4]:
```
29
```

```
The history saving thread hit an unexpected error (OperationalError('datab
ase or disk is full')).History will not be written to the database.
```

Octal

- octa mean 8 : base=8

- base=8 , 8 digits required
- the 8 digits are 0,1,2,3,4,5,6,7
- The representation is 0o
- example: 0o4567 or 0O1234

In [7]:
```
0o123
```
Out[7]:
```
83
```

hexa

- hexa mean 16 : base=16
- base=16 , 16 digits required
- the 16 digits are 0,1,2,3,4,5,6,7,8,9,A(10),B(11),C(12),D(13),E(14),F(15)
- The representation is 0x
- example: 0xABC or 0XabcF

esc+m

In [6]:
```
0XabcF
```
Out[6]:
```
43983
```

float

In [8]:
```
num=1.234
type(num)
```
Out[8]:
```
float
```

In [9]:
```
float
```
Out[9]:
```
float
```

In [11]:
```
2e4   #2*10000
```
```
...
```

In [12]:
```
2e6  # 2*1000000
```
Out[12]:
```
2000000.0
```

In [13]:
```
2e-3 # 2/1000
```
Out[13]:
```
0.002
```

In [ ]:
```
1.00000000000000e-14

#1/1000000000000000= 0.0000000000000001
# the senario: ans=0
# 1.00000000000000e-14
```

```
In [15]: 1.000000
Out[15]: 1.0
```

```
In [ ]: e+3 or e3 are same?
```

```
In [16]: 2e+3
Out[16]: 2000.0
```

```
In [17]: 2e3
Out[17]: 2000.0
```

boolean

boolean

```
In [21]: a=True      #  number = 1
         type(a)
Out[21]: bool
```

```
In [19]: b=False   # number = 0
         type(b)
Out[19]: bool
```

```
In [23]: true=100  # we alreday defined here
         name=true
         name
Out[23]: 100
```

String

- String means charcters
- English letters

```
In [2]: python=100
        # value 100 we are saving a variable python
        # so now python is intialised
```

```
In [3]: name=python
        # Name error
        # we are saving python in a variable name
        # It will try to understand where is
        # python intialized
```

In [4]:
```
name
```
Out[4]:
```
100
```
In [5]:
```
name1='python'
name1
```
Out[5]:
```
'python'
```
In [6]:
```
type(name1)
```
Out[6]:
```
str
```
In [7]:
```
name2="python"
name2
```
Out[7]:
```
'python'
```
In [8]:
```
name3="python'
name3
```
```
  Cell In[8], line 1
    name3="python'
          ^
SyntaxError: unterminated string literal (detected at line 1)
```
In [ ]:
```
hai                 # first
how are you         # second

with out /t  and  /n
```
In [9]:
```
print("Hai")
print("How are you")

# 10 lines
```
```
Hai
How are you
```
In [12]:
```
name4="""hai
         how are you
         """

print(name4)

name4

# DOC STRING
```
```
hai
         how are you
```
Out[12]:
```
'hai \n         how are you\n         '
```

Doc string

In programming, a docstring is a string literal specified in source code that is used, like a comment, to document a specific segment of code

use triple quotes

$Complex - Conjugate$:

- It represents a+jb
- complex
- where a= real number
- b= imaginary number
- i= square root(-1)

In [13]:
```
complex()

# curoser should be inside bracket
# shift+tab at a time

# if you are not provide anything isside the bracket
# the default arguments will use
# real=0  and img=0
# 0+0j
# when you enter 0+0j , the python output is 0j


A) 0+0j  ( why cant answer A)
B) 0j
C) 0
D)None of the above
```

Out[13]:
```
0j
```

In [14]:
```
0+0j
```

Out[14]:
```
0j
```

In [15]:
```
complex(3,5)
# 3+5j
```

Out[15]:
```
(3+5j)
```

- integer : int
- float : float
- string : str
- complex : complex

In [16]:
```
num=3+10j
num
```

Out[16]:
```
(3+10j)
```

```
In [17]: type(num)

Out[17]: complex

In [18]: num.real

Out[18]: 3.0

In [19]: num.imag

Out[19]: 10.0

In [ ]:

In [ ]:
```