complex number

“complex” is a data type or class which is used to represent complex number or complex object.

Complex number is having two values

1. real
2. imag

In python complex number is represented with following syntax

        real+imagj

```
>>> c1=1+2i
SyntaxError: invalid decimal literal
>>> c1=1+2j
>>> c1
(1+2j)
>>> type(c1)
<class 'complex'>
>>> c1.real
1.0
>>> c1.imag
2.0
>>> c1.real=1.2
Traceback (most recent call last):
  File "<pyshell#6>", line 1, in <module>
    c1.real=1.2
AttributeError: readonly attribute
>>> c1=1.5+2j
>>> c1.real
1.5
>>> c1.imag
2.0
>>> c2=1j
>>> c2
1j
>>> type(c2)
<class 'complex'>
>>> c2.real
0.0
>>> c2.imag
```

1.0

bool type
this data type is used to represent Boolean values
In python Boolean values represented using two keywords

1. True
2. False

These are constants literals whose value never changed

Relational operations

10>20 → False
20>10 → True
10==10 → True

Rollno=101 # integer type
Fee=4000.0 # float type
Fee_paid=True # Boolean

Example:

```
>>> b1=True
>>> b2=False
>>> type(b1)
<class 'bool'>
>>> type(b2)
<class 'bool'>
>>> b3=0
>>> type(b3)
<class 'int'>
>>> True+False
1
>>> True+True
2
>>> False+False
0
```

NoneType

NonType is used to represent **None** value

If the variable does not have any value, it is represented as None

```
>>> x=None
>>> type(x)
<class 'NoneType'>
>>> y=None
>>> type(y)
<class 'NoneType'>


>>> y=8
>>> p=1.5
>>> c=1+2j
>>> d-=True
Traceback (most recent call last):
  File "<pyshell#47>", line 1, in <module>
    d-=True
NameError: name 'd' is not defined. Did you mean: 'id'?
>>> d=True
>>> e=None
r
Traceback (most recent call last):
  File "<pyshell#50>", line 1, in <module>
    r
NameError: name 'r' is not defined
d
True
>>> e
>>> c
(1+2j)
p
1.5
>>> y
8
>>> u
Traceback (most recent call last):
  File "<pyshell#56>", line 1, in <module>
 >>>   u
NameError: name 'u' is not defined
>>> u=0
```

```
u
0
>>> u=5
>>> u
5
>>> k
Traceback (most recent call last):
  File "<pyshell#61>", line 1, in <module>
    k
NameError: name 'k' is not defined
>>> number1=100
>>> number2=200
>>> number1
100
>>> number2
200
>>> number1+number2
300
>>> number3=number1+number2
>>> number3
300
```

Scalar types

1. Int → decimal,octal,hexa,binary
2. Float → fixed , exponent
3. Complex → real+imagj
4. Bool → True,False
5. Nonetype → None

String data type

"str" class or data type is used to represent string object
String is a collection of character or group of characters
String is immutable, after creating string we cannot modify
String sequence data type

In python string is represented in 3 ways

1. Within single quotes

2. Within double quotes
3. Within triple quotes

Within single quotes we can represent single line string

Example:

```
>>> a='naresh'
>>> a
'naresh'
>>> student_name='suresh'
>>> student_name
'suresh'
>>> player_name='rohit'
>>> player_name
'rohit'
>>> str1='python is a
SyntaxError: unterminated string literal (detected at line 1)
>>> s1='45'
>>> s2='1.5'
>>> type(s1)
<class 'str'>
>>> type(s2)
<class 'str'>
>>> s3='python is "easy" language'
>>> print(s3)
python is "easy" language
>>> s4='python is 'HL' langauge'
SyntaxError: invalid syntax
```

Within single quotes we can insert double quotes

Within double quotes we can represent single line string
Within double quotes we can insert single quotes

```
>>> s5="Python is a programming language"
>>> print(s5)
Python is a programming language
>>> s6="python is a 'scripting' langauge"
>>> print(s6)
python is a 'scripting' language
```

```
>>> s1="abcd"
>>> print(s1)
abcd
>>> user="nit123"
>>> pwd="nit123$%*"
```

within triple quotes
within triple single quotes or double quotes, we represent multiline string

```
name=”naresh”
course=”python”
remarks=’’’
                        ‘’’
```