Complex
This function is used to construct complex number
String→ complex, complex→ complex, float → complex
Syntax-1: complex(value)
Syntax-2: complex(real=0.0,imag=0.0)

```
>>> c1=complex(s1)
>>> print(c1,s1)
(1+2j) 1+2j
>>> print(type(c1),type(s1))
<class 'complex'> <class 'str'>
>>> c2=1+2j
>>> c3=complex(c2)
>>> print(c2,c3)
(1+2j) (1+2j)
>>> c3=complex(1.5)
>>> print(c3)
(1.5+0j)
>>> c4=complex(1.5,1.0)
>>> print(c4)
(1.5+1j)
>>>
```

Example:
```
# write a program to add two complex numbers

c1=complex(input("Input Complex Number1:"))
c2=complex(input("Input Complex Number2:"))
c3=c1+c2
print(c1,c2,c3)
```

Output:
```
Input Complex Number1:1+2j
Input Complex Number2:1+0j
(1+2j) (1+0j) (2+2j)
>>>
```

Example
```
# write a program to add two complex numbers
```

```
print("Enter complex1")
real=float(input("Enter real"))
imag=float(input("Enter imag"))
c1=complex(real,imag)
print("Enter Complex2")
real=float(input("Enter real"))
imag=float(input("Enter imag"))
c2=complex(real,imag)
c3=c1+c2
print(c1,c2,c3)
```

Output:

```
Enter complex1
Enter real1.0
Enter imag1.0
Enter Complex2
Enter real2.0
Enter imag1.0
(1+1j) (2+1j) (3+2j)
>>>
```

bool()

this function or type is used for constructing Boolean object.
Bool→ bool, int—bool, float – bool

```
>>> b1=bool("True")
>>> print(b1)
True
>>> print(type(b1))
<class 'bool'>
>>> b2=bool("False")
>>> print(b2)
True
>>> b3=bool("Python")
>>> print(b3)
True
>>> ord('A')
65
>>> ord('B')
66
>>> ord('Z')
```

```
90
>>> ord('a')
97
>>> chr(65)
'A'
>>> chr(66)
'B'
>>> b4=bool(0)
>>> print(b4)
False
>>> b5=bool(1)
>>> print(b5)
True
>>> b6=bool(100)
>>> print(b6)
True
>>>
```

str()
it is used to represent string object.
String-string, int—string,float—string,complex—string,

```
>>> s1=str(65)
>>> print(s1)
65
>>> s2=str(1.5)
>>> print(s2)
1.5
>>> s1+s2
'651.5'
>>> s3=str(1+2j)
>>> print(s3)
(1+2j)
>>> s4=str(1+1j)
>>> s5=s3+s4
>>> print(s5)
(1+2j)(1+1j)
>>> s5=str(True)
>>> print(s5)
True
```

```
>>>
```

Python Operators

What is operator?

Operator is a special symbol, which is used to perform some operations.
Based on operands on which it perform operation the operators are classified into 3 categories

1. Unary operators : uses one operand to perform operation
2. Binary operators : uses two operands to perform operation
3. Ternary operator : uses three operands to perform operation

Types of operators

1. Arithmetic operators
2. Relational operators
3. Logical operators
4. Assignment operators
5. Bitwise operators
6. Conditional operator
7. Identity operator
8. Membership operator
9. Walrus operator (Python 3.8)

Arithmetic operators

Arithmetic operators are binary operators.

Operator	**Description**

+	+ operator is used to perform two operations
	1. Adding numbers
	2. Concatenation of strings/sequence
	If two operands are numbers it performs addition
	If two operands are string or sequence types it performs concatenation.
	
	`>>> n1=65`
	`>>> n2=10`
	`>>> n3=n1+n2`
	`>>> print(n1,n2,n3)`
	`65 10 75`
	`>>> s1="65"`
	`>>> s2=10`
	`>>> s3=s1+s2`
	`Traceback (most recent call last):`

	File "<pyshell#54>", line 1, in <module>
	    s3=s1+s2
	TypeError: can only concatenate str (not "int") to str
	>>> s2="70"
	>>> s3=s1+s2
	>>> print(s1,s2,s3)
	65 70 6570
	>>> s4="Python"
	>>> s5="Language"
	>>> s6=s4+s5
	>>> print(s4,s5,s6)
	Python Language PythonLanguage
	>>>
	>>> x=45
	>>> y=65
	>>> x.__add__(y)
	110
	>>>

-	This operator is used to subtract numbers
	>>> n1=65
	>>> n2=10
	>>> n3=n1-n2
	>>> print(n1,n2,n3)
	65 10 55
	>>>

*	This operator is used to perform two operations
	1. Multiplying numbers
	2. Repeating a sequence number of times
	>>> n1=5
	>>> n2=6
	>>> n3=n1*n2
	>>> print(n1,n2,n3)
	5 6 30
	>>> print("-"*80)
	--- 	
	-
	>>> print("Python"*5)
	PythonPythonPythonPythonPython
	>>> print(5*"Python")
	PythonPythonPythonPythonPython

	>>>
/	
//	
%	
**	

Implicit conversion
Arithmetic operations done on different data types, python return result in broader type.

1. Int → III
2. Float → II
3. Complex → I