

03/11/23  
Fri

### 3. > Complex() function:

This fn is used to create complex no (OR) perform the following conversions.

- i) complex to complex.
- ii) complex to int.
- iii) str to complex.
- iv) bool to complex.
- v) float to complex.

Ex:- comp1 = complex(1+2j)

```
>>> print(comp1)  
(1+2j)
```

Ex!:- print(type(comp1))

```
<'class 'complex'>
```

Ex:- >>> comp2 = complex("1+2j")

```
>>> print(comp2)  
(1+2j)
```

Ex:- print(type(comp2))

```
<'class 'complex'>
```

~~Ex:-~~ >>> comp3 = complex("2j")

```
>>> print(comp3)  
2j
```

```
>>> print(type(comp3))  
<'class 'complex'>
```

```
>>> comp4 = complex("4")  
print(comp4)
```

```
(4+0j)
```

```
>>> comp5 = complex(1)
```

```
>>> point(comp5)
```

~~'(1+0j)'~~

```
>>> comp6 = complex(2j)
```

```
>>> point(comp6)
```

$2j$

```
>>> point(comp6.real, comp6.imag)
```

$0.0 \cdot 2.0$

```
>>> comp7 = complex(1+0j)
```

```
>>> point(comp7, type(comp7))
```

$(1+0j) < \text{class 'Complex'}$

# N.A.P. to adding two complex numbers:-

```
Comp1 = complex(input("input complex number 1"))
```

```
Comp2 = input("input complex number 2")
```

```
Comp3 = Comp1 + Comp2
```

point(f"sum of {comp1} and {comp2} is {comp3}")

O/P:-  
input complex number 1+2j

input complex number 2+1+3j

sum of  $(1+2j)$  and  $(1+3j)$  is  $(2+5j)$

\* Any value except zero(0), All value will be true.

#### 4) bool function:-

This fn returns Boolean value (OR) this function performs the following conversions -

- 1) Bool to bool
- 2) int to bool
- 3) string to bool
- 4) complex to bool.
- 5) float to bool.

Eg:- Bool to Bool

(1)  $b_1 = \text{bool}(\text{True})$

>>> print( $b_1$ )

True.

(2) >>>  $b_2 = \text{bool}(\text{False})$

>>> print( $b_2$ )

False

(3) >>>  $b_3 = \text{bool}(0)$

>>> print( $b_3$ )

False

(4)  $b_4 = \text{bool}(1)$

>>> print( $b_4$ )

True

(5)  $b_5 = \text{bool}(100)$

>>> print( $b_5$ )

True.

(6)  $b_6 = \text{bool}(0.0)$

>>> print( $b_6$ )

False

(7)  $b_7 = \text{bool}(-5)$

>>> print( $b_7$ )

True.

(8)  $b_8 = \text{bool}(0+0j)$

>>> print( $b_8$ )

False

(9)  $b_{10} = \text{bool}(1+0j)$

>>> print( $b_{10}$ )

True

(10)  $b_{11} = \text{bool}("A")$

>>> print( $b_{11}$ )

True.

(11)  $b_{12} = \text{bool}("NARESH")$

>>> print( $b_{12}$ )

True.

(12)  $b_{13} = \text{bool}("False")$

>>> print( $b_{13}$ )

True

this is string  
convert ASCII  
value into  
bool.

(13)  $b_{14} = \text{bool}("True")$

>>> print( $b_{14}$ )

True.

same as  
above.

\* All ASCII value starts from capital 'A' → value will 65.  
and Z → 90.

# ord(), char().

ord(char) → this function returns ASCII value of input character.

char(ascii) → this function returns character value of input ~~ascci~~ ASCII value.

Example:-

1) >>> ord('A')

65

2) >>> ord('B')

66

3) >>> ord('C')

67

4) >>> ord('Z')

90

(1) 5) >>> ord('a')

97

6) >>> ord('b')

98

7) >>> ord('c')

99

8) >>> char(65)

'A'

9) >>> char(66)

'B'

10) > char(65)

'A'

11) > char(97)

'b'

12) > char(122)

'z'

Eg:- N.A.P. - to convert given character from uppercase to lowercase.

Soln:- Char='B'

point(char)

value = ord(char) # 66

Value = value + 32

char2 = chr(value)

point(char1)

value = ord(char1)

Value = value - 32

char2 = chr(value)

point(char2)

O/p  
B  
b  
B  
B

# str() function.

→ This function returns string object (OR) this function is used to convert other type into string type.

Synt: str(value)

>> s1 = str()

point(s1)

>> s2 = str(25)

>> point(s2)

25

>> s3 = str(1.5)

>> point(s3)

1.5

>> s2 + s3

'251.5'

>> point(type(s2), type(s3))

<class 'str'> <class 'str'>

>> s4 = str(1+2j)

>> point(s4) → (1+2j)



Complex (beej)  
complex is  
broadest/higher  
type.)

```

>>> print(type('sy'))
<class 'str'>
>>> ss = str(True)
>>> print(ss)
True.

>>> s6 = str(False)
>>> print(s6)
False.

>>> print(type(ss), type(s6))
<class 'str'> <class 'str'>.

```

## # Operators:-

Operator is a special symbol, which is used to perform operations. Based on the numbers of operand used to perform operation, the operators are classified into 3 categories.

- 1) Unary operator
- 2) Binary operator
- 3) Ternary operator

## # Types of operators:-

- 1) Arithmetic
- 2) Relational
- 3) Logical
- 4) Assignment
- 5) Membership
- 6) Identity
- 7) Conditions
- 8) Bitwise.

9) modulus operator (python 3.8)

(12) % 5 = 2.00

(12) % 6 = 0.00

(12) // 5 = 2.00

(12) // 6 = 2.00

2.00

## Arithmetic operators:

- These operators are used to perform arithmetic operators.
- All arithmetic operators are binary operators.

operators	Description.
+	Adding or concatenation
-	Subtraction
*	multiplication or repeat.
/	Float division
//	Floor division.
%	Modulo
**	Exponent or power.

①

+ operator :- is used to perform two operations.

1) Adding

2) Concatenating.

\* it performs addition, if two operands are numbers.

\* it performs concatenation, if two operands sequences  
(list, tuple, string, ---)

Example:-

>>> a = 10

>>> b = 20

>>> c = a + b

>>> print(a, b, c)

10 20 30

>>> x = 1.5

>>> y = 1.2

>>> z = x + y

>>> print(x, y, z)

1.5 1.2 2.7

```
>>> b1 = True  
>>> b2 = False  
>>> b3 = b1 + b2  
>>> print(b1, b2, b3)  
True False 1  
>>> b4 = True + 15  
>>> print(b4)  
16  
>>> c1 = 1+2j  
>>> c2 = 1+1j  
>>> c3 = c1 + c2  
>>> print(c1+c2+c3)  
(1+2j) (1+1j) (2+3j)  
>>> s1 = "Python"  
>>> s2 = "3.12"  
>>> s3 = s1 + s2  
>>> print(s1, s2, s3)  
Python 3.12 python 3.12  
>>> s4 = "10"  
>>> s5 = "15"  
>>> s6 = s4 + s5  
>>> print(s4, s5, s6)  
10 15 1015  
>>> s7 = "Python"  
>>> s8 = 3.12  
>>> print(s7+s8)
```

Traceback (most Recent call last):

File " <pyshell #108> ", line 1, in <module>  
print (s7 + s8)

Type Error: can only concatenate str (not 'float') to str

### eval() function

it is a Predefined function of python. this function evaluate String Representation of expression and return value.

### eval(expression)

```
>>> a = eval("1.5")
>>> print(a, type(a))
1.5 <class 'float'>
>>> b = eval("25")
>>> print(b, type(b))
25 <class 'int'>
>>> c = eval("1+2j")
>>> print(c, type(c))
(1+2j) <class 'complex'>
>>> d = eval("10+20+30")
>>> print(d)
60
```

### Example :-

```
# write a program to add two numbers
a = eval(input("enter first number"))
b = eval(input("enter second number"))
c = a+b
print(f sum of {a} and {b} is {c})
```

Output! - <sub:ans> in , 1.0, " (30 + 10j) >" 30  
(10 + 10j) tri

Enter first number 10

Enter Second number 20

Sum of 10 and 20 is 30

Enter first number 1.5

Enter Second number 1.2

Sum of 1.5 and 1.2 is 2.7

Enter first number 1+2j

Enter Second number 1+1j

Sum of (1+2j) and (1+3j) is (2+3j)

### - Operator (arithmetic subtract operator)

This operator is used for subtraction of two number.

```
>>> a = 10  
>>> b = 1.5  
>>> c = a - b  
>>> print(a, b, c)
```

10 1.5 8.5

```
>>> c1 = 1+2j  
>>> c2 = 1j  
>>> c3 = c1 - c2
```

1+2j 1j (1+1j)

```
>>> c3 = c1 - 2  
>>> print(c3)  
(-1+2j)
```