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:- compl = complex(1+2j)
>>> print(compl)
(1+2j)

Ex:- print(type(compl))
<class 'complex'>

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

Ex:- print(type(comp2))
<class 'complex'>

>>> comp3 = complex("2j")
>>> print(comp3)
2j

>>> print(type(comp3))
<'class' complex'>
>>> com4 = complex("4")
print(comp4)
(4+0j)

```
>>> comp5 = complex(1)
>>> print(comp)
(1+0j)

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

>>> print(comp6.real, comp6.img)
0.0 2.0

>>> comp7 = complex(true)
>>> print(comp7, type(comp7))
(1+0j) <class 'complex'>

>>> complex8 = complex(false)
>>> print(comp8, type(complex8))
0j <class 'complex'>

>>> print(complex8.real, complex
8.img)
0.0 0.0
```

W.A.P. to adding two complex numbers:-

```
Comp1 = complex(input("input complex number 1"))
Comp2 =  "  (  "  ("  "  "  "  2"))
Comp3 = comp1 + comp2
print(f'Sum of {comp1} and {comp2} is {comp3}')
```

O/P:-

input complex numbers 1+2j

" " number2 + 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) b1 = bool(True)
>>> print(b1)
True.

(2) >>> b2 = bool(False)
>>> print(b2)
False

(3) >>> b3 = bool(0)
>>> print(b3)
False

(4) b4 = bool(1)
>>> print(b4)
True

(5) b5 = bool(100)
>>> print(b6)
True.

(6) b7 = bool(0.0)
>>> Print(b7)
False

b8 = bool(-5)
(7) >>> Print(b8)
True.

(8) b9 = bool(0+0j)
>>> print(b9)
False

(9) b10 = bool(1+0j)
>>> print(b10)
True

(10) b11 = bool("A")
>>> print(b11)
True.

(11) b12 = bool("NARESH")
>>> print(b12)
True.

(12) b13 = bool("false")
>>> print(b13)
True

this is string convert ASCII value into bool.

(13) b14 = bool("True")
>>> print(b14)
True.

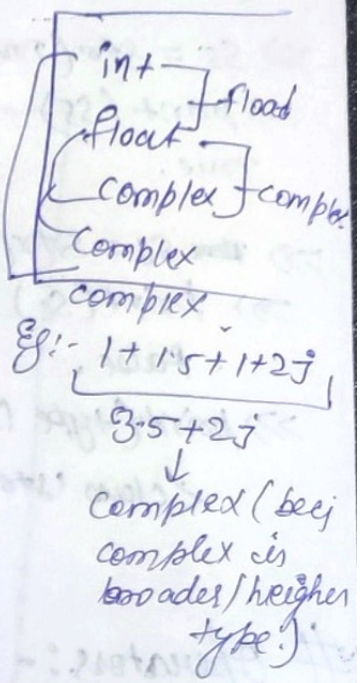
same as above.

Eg:-

W.AP to convert given character from uppercase to lowercase.

Sol^n:-
```
Char='B'
print(char)
value = ord(char) #66
Value = value +32
Char1 = chr(value)
print(Char1)
value = ord(Char1)
Value = value -32
Char2 = chr(value)
print(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.

Syntax: str(value)

```
>>> s1 = str()
    print(s1)

>>> s2 = str(25)
>>> print(s2)
    25

>>> s3 = str(1.5)
>>> print(s3)
    1.5

>>> s2 + s3
    '251.5'

>>> print(type(s2), type(s3))
    <class 'str'> <class 'str'>

>>> s4 = str(1+2j)
>>> print(s4) → (1+2j)
```

```
>>> print(type(S4))
    <class 'str'>
>>> S5 = str(True)
>>> print(S5)
    True.

>>> S6 = str(False)
>>> print(S6)
    False.
>>> print(type(S5), 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) Conditional
8) Bitwise.
9) Walrus operator (python 3.8)

1) Arithmetic operators:

→ These operators are used to perform arithmetic operations.

→ All arithmetic operators are binary operators.

Operators	Description.
+	Adding or concatenation
-	subtraction
*	multiplication or Repeat.
/	Float Division
//	Floor Division.
%	Modulo
**	Exponent or power.

(i)

+ operator :- is used to perform two operations.

1> Adding

2> Concatinating.

* it performs addition, if two operands are number.
* it performs concatenation, if two operands sequences (list, tuple, string, ----)

Examples:-

```
>>> 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 :-

```
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
>>> Print (c1, c2, c3)
(1+2j) 1j (1+1j)
>>> c3 = c1-2
>>> Print (c3)
(-1+2j)
```