str()
It is a predefined function; this function performs the following conversions.

1. str to str
2. int to str
3. float to str
4. complex to str
5. bool to str

Syntax: str(value)

```
>>> s1=str("python")
>>> s2=str(25)
>>> s3=str(1.5)
>>> s4=str(1+2j)
>>> s5=str(True)
>>> print(s1,s2,s3,s4,s5,sep="\n")
python
25
1.5
(1+2j)
True
>>> print(type(s1),type(s2),type(s3),type(s4),type(s5))
<class 'str'> <class 'str'> <class 'str'> <class 'str'> <class 'str'>
```

Operators
What is operator?
Operator is a special symbol which is used to perform operation.
Based on the number of operands on which it performs operations, operators are classified 3 categories

1. Unary Operators : An operator which uses one operand to evaluate expression.
2. Binary Operators : An operator which uses two operands to evaluate expression.
3. Ternary Operators: An operator which uses three operands to evaluate expression.

Types of operators

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

Arithmetic Operators

These are binary operators, It required 2 operands.

Operator	Description
+	Addition and Concatenation
-	Subtraction
*	Multiplication and repetition
/	Division (Float Division)
//	Division (Floor Division)
%	Division (Modulo)
**	Power/Exponent

+ operator

This operator is used to perform two operations

1. Adding numbers
2. Concatenating sequences

PVM performs addition, if two operands are numeric type.
PVM performs concatenation, if two operands are sequences.

```
>>> 10+
SyntaxError: incomplete input
>>> 10+20
>>> 30
>>> res1=10+20
>>> res2=1.5+1.1
>>> res3=1+2j+1+1j
>>> print(res1,res2,res3,sep="\n")
30
2.6
```

```
(2+3j)
>>> res4=True+True
>>> print(res4)
2
>>> res5="Python"+"Language"
>>> print(res5)
PythonLanguage
>>> res6="Python"+3.12
Traceback (most recent call last):
  File "<pyshell#17>", line 1, in <module>
    res6="Python"+3.12
TypeError: can only concatenate str (not "float") to str
>>> res7=[1,2]+[3,4]
>>> print(res7)
[1, 2, 3, 4]
>>> res8="PYTHON"+[3,4]
Traceback (most recent call last):
  File "<pyshell#20>", line 1, in <module>
    res8="PYTHON"+[3,4]
TypeError: can only concatenate str (not "list") to str
```

Example:

```
# write a program to input two numbers and perform
# addition

num1=int(input("Enter first number"))
num2=int(input("Enter second number"))
num3=num1+num2
print("Sum of ",num1,"and",num2,"is",num3)
```

Output:

```
Enter first number10
Enter second number20
Sum of  10 and 20 is 30
```

- **Operator**

This is used to perform subtraction

```
>>> n1=10
>>> n2=5
>>> n3=n1-n2
>>> print(n1,n2,n3)
10 5 5
>>> x=10
>>> y=1.5
>>> z=x-y
>>> print(x,y,z)
```

int+complex → complex
int+float →float

complex>float>int

Example:

```
>>> a=1
>>> b=1.5
>>> c=1
+
>>> c=1+2j
>>> d=a+b+c
>>> print(d)
(3.5+2j)
```

Example:

```
# write a program to swap two numbers

num1=int(input("Enter value of num1"))
num2=int(input("Enter value of num2"))

print("Before Swaping",num1,num2)

# Method-1 using 3 variable

num3=num1
num1=num2
num2=num3

print("After Swaping ",num1,num2)
```

```
# Method-2 without using 3 variable

num1=num1+num2
num2=num1-num2
num1=num1-num2

print("After Swaping ",num1,num2)
```

Output:

```
Enter value of num110
Enter value of num220
Before Swaping 10 20
After Swaping  20 10
After Swaping  10 20
```

***Operator**

This operator is used to perform two operations

1. Multiplying
2. Repeating