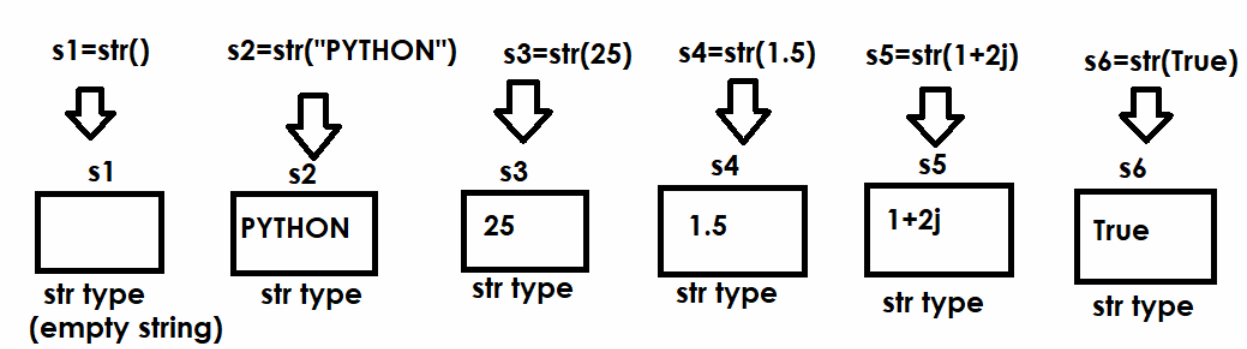
str() function

str() is a predefined function, this function is used to perform the following conversions.

1. String to string
2. Int to string
3. Float to string
4. Complex to string
5. Boolean to string

Note: str() function can be used to convert any type into string type



Example:	**Output**
a=10 print(a,type(a)) b=str(a) print(b,type(b)) c=1.5 print(c,type(c)) d=str(c) print(d,type(d)) e=1+2j print(e,type(e)) f=str(e) print(f,type(f)) g=True print(g,type(g)) h=str(g) print(h,type(h))	10 <class 'int'> 10 <class 'str'> 1.5 <class 'float'> 1.5 <class 'str'> (1+2j) <class 'complex'> (1+2j) <class 'str'> True <class 'bool'> True <class 'str'>

chr(),ord() function

chr() is a predefined function in python, this function returns character value of given ascii value

```
>>> chr(65)
'A'
>>> chr(66)
'B'
>>> chr(90)
'Z'
>>> chr(97)
'a'
>>> chr(98)
'b'
>>> chr(121)
'y'
>>> chr(122)
'z'
>>> chr(47)
'/'
>>> chr(49)
'1'
>>> chr(48)
'0'
>>> chr(52)
'4'
```

Ord() is predefined function in python. This function returns ASCII value of given character value

```
>>> ord('A')
65
>>> ord('B')
66
>>> ord('Z')
90
>>> ord('a')
97
>>> ord('b')
```

```
98
>>> ord('z')
122
>>> ord('0')
48
>>> ord('9')
57
```

Example:

```
# write a program to convert uppercase letter to lowercase

ch='A'
n=ord(ch)+32
ch1=chr(n)

print(ch,ch1)
```

Output

```
A a
```

Example:

```
# Write a program to convert lowercase letter to uppercase

ch='a'
n=ord(ch)-32
ch1=chr(n)
print(ch,ch1)
```

Output

```
a A
```

Operators

What is operator?

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

1. Unary Operators

2. Binary Operators
3. Ternary Operators

An operator required one operand to perform operation is called unary operator
An operator required two operands to perform operation is called binary operator
An operator required three operands to perform operation is called ternary operator

Python support 9 types of operators

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

Arithmetic Operators
Arithmetic operators are binary operators and required two operands to perform operation

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

+ Operator

In python + operator is used to perform two operations

1. Adding two numbers
2. Concatenation two sequences

If two operands are numeric type, + operator perform addition.
If two operands are sequences, + operator performs concatenation.

Example:

```
n1=10+20
n2=1.5+2.5
n3=1+2j+1+3j
n4=True+True
n5="PYTHON"+"PROGRAMMING"
n6=[10,20,30]+[40,50,60]
print(n1,n2,n3,n4,n5,n6,sep="\n")
n7="PYTHON"+"LANGUAGE"
print(n7)
```

Output

```
30
4.0
(2+5j)
2
PYTHONPROGRAMMING
[10, 20, 30, 40, 50, 60]
PYTHONLANGUAGE
```

Implicit Type Casting or Type conversion

The type conversion done by python while performing arithmetic operations is called implicit type casting or type conversion

1. Complex
2. Float
3. Int

```
>>> a=10+20
>>> print(a,type(a))
30 <class 'int'>
>>> b=10+1.5
>>> print(b,type(b))
11.5 <class 'float'>
```

```
>>> c=10+1.5+1+2j
>>> print(c,type(c))
(12.5+2j) <class 'complex'>
>>> d=1+1+2j
>>> print(d,type(d))
(2+2j) <class 'complex'>
>>> e=int(1+1.5)
>>> print(e,type(e))
2 <class 'int'>
```

Example:

```
# Write a program to name, 2 subject marks and calculate
# total marks

# Input
name=input("Enter Name :")
sub1=float(input("Enter Subject1 Marks "))
sub2=float(input("Enter Subject2 Marks "))

#Process
total=sub1+sub2

#Output
print("StudentName ",name)
print("Subject1 Marks ",sub1)
print("Subject2 Marks ",sub2)
print("Total Marks ",total)
```

Output

```
Enter Name :naresh
Enter Subject1 Marks 50
Enter Subject2 Marks 60.40
StudentName  naresh
Subject1 Marks  50.0
Subject2 Marks  60.4
Total Marks  110.4
```

-Operator (Arithmetic subtraction Operator)

This operator is used to perform only one operation

This operator is used to perform subtraction operation
It is a binary operator and required two operands

Example:

```
>>> a=10
>>> +a
10
>>> -a
-10
>>> ++a
10
>>> +-a
-10
>>> --a
10
>>> ++++++++------++++a
10
>>> a=5
>>> a++
SyntaxError: invalid syntax
>>> a--
SyntaxError: invalid syntax
>>> a++5
10
>>> a+-5
0
>>> a--5
10
>>> a---5
0
>>> r1=10-5
>>> r2=1.5-0.5
>>> print(r1,r2)
5 1.0
>>> r3=True-False
>>> print(r3)
1
```

***Operator**

*is an binary operator and required 2 operands
* is operator is used to perform 2 operations

1. Multiplication
2. Repeating

If two operands are numeric type, it performs multiplication
If one operand is a sequence type and other integer type, it performs repetition

```
>>> r1=5*2
>>> r2=1.5*2
>>> print(r1,r2)
10 3.0
>>> r3=1+2j*1+1j
>>> print(r3)
(1+3j)
>>> r4=(1+2j)*(1+1j)
>>> print(r4)
(-1+3j)

>>> r5="A"*10
>>> print(r5)
AAAAAAAAAA
>>> r6=[10]*5
>>> print(r6)
[10, 10, 10, 10, 10]
>>> r7=[10,20]*5
>>> print(r7)
[10, 20, 10, 20, 10, 20, 10, 20, 10, 20]
>>> r8="PYTHON"*3
>>> print(r8)
PYTHONPYTHONPYTHON
>>> r9=100*[0]
print(r9)
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
>>> r10=[5]*50
```

```
>>> print(r10)
[5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5]
>>> r11=[5]*1.5
Traceback (most recent call last):
  File "<pyshell#74>", line 1, in <module>
    r11=[5]*1.5
TypeError: can't multiply sequence by non-int of type 'float'
```