**Lecture 6**

**Operator Precedence** in **Descending** order is shown below. If operators from same precedence occur in an expression follow Left to Right associativity rule.

1. **( )**
2. ****** (exponent)
3. **~, +, -** (unary operators)
4. ***, /, %**(mod)**,** //(floor division)
5. **+, -**
6. **<<, >>** (shift)
7. **&** (Bitwise and)
8. **^, |**
9. **<, <=, >, >=** (comparison)
10. **==, !=**
11. **=, +=, *=, -=** (assignment)
12. **is, is not** (identity)
13. **in, not in** (membership)
14. **and, or, not** (logical)

Use **eval()** function to evaluate an expression.

**Eg1:**

```
print(eval('8+2*7/2-2'))  #13.0
print(eval('8+2*7//2-2'))  #13 (floor division only take integer part)
print(eval("2 + 10 // 3 + (5 % 2) ** 2")) #6
```

**Eg2:**

```
x=10 , y=10
print(eval('x!=y')) #False
print(eval("x < 200 and y > 100")) #False
print(eval("x in {50, 10, 150, 200}")) #True
print(eval("x is y")) #True

x=[10,20], y=[10,20]  # x and y are objects
print(eval("x is y"))  #False
```

**Problems**

```
x=input("Enter first no")
y=input("Enter second no")
print(x+y)
```

```
#The above code snippet maynot get proper result (need to specify the type as shown below)

x=int(input("Enter first no"))    #But we cannot give a float input here, will show error
y=float(input("Enter second no"))
print(x+y)
```

**eval() solve this (Automatically accepts any datatype)**

```
x=eval(input("Enter first no"))
y=eval(input("Enter second no"))
print(x+y)
```

```
Enter first no10
Enter second no20.8
30.8
```

**input() function also accepts input expressions and can be evaluated using eval()**

**Eg:**

```
evaluate = input("Enter what operation x has to perform: ")
print(evaluate)
print(type(evaluate))
-----------------------------------------------------------------------
Enter what operation x has to perform: x + x + 100 - 35 + 5 * 80
x + x + 100 - 35 + 5 * 80
<class 'str'>


x = 10
print(type(x))
-----------------------------------------------------------------------
<class 'int'>


expression = eval(evaluate)
print(expression)
print(type(expression))
-----------------------------------------------------------------------
485
<class 'int'>
```