****

**TRIBHUVAN UNIVERSITY**

**INSTITUTE OF ENGINEERING**

PULCHOWK CAMPUS

AN ASSIGNMENT ON

Operator Overloading in Python

SUBMITTED BY:

SUSHANT THAKUR (081BEL092)

SUBMITTED TO:

PRAKASH CHANDRA PRASAD SIR,

DEPARTMENT OF ELECTRONICS & COMPUTER ENGINEERING

**OPERATOR OVERLOADING IN PYTHON**

**Objective:**

The objective of this assignment is to implement operator overloading for all arithmetic and relational operators in Python.

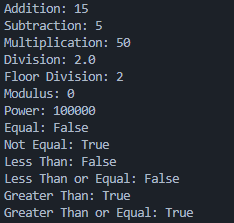
**Theory:**

Operator overloading in Python allows developers to redefine the behavior of built-in operators for user-defined classes. This enables intuitive interaction with objects of custom classes using standard operators like +, -, \*, /, ==, <, etc.

**Code:**

class Number:  
 def \_\_init\_\_(self, value):  
 self.value = value  
  
 # Arithmetic operators  
 def \_\_add\_\_(self, other):  
 return Number(self.value + other.value)  
  
 def \_\_sub\_\_(self, other):  
 return Number(self.value - other.value)  
  
 def \_\_mul\_\_(self, other):  
 return Number(self.value \* other.value)  
  
 def \_\_truediv\_\_(self, other):  
 return Number(self.value / other.value)  
  
 def \_\_floordiv\_\_(self, other):  
 return Number(self.value // other.value)  
  
 def \_\_mod\_\_(self, other):  
 return Number(self.value % other.value)  
  
 def \_\_pow\_\_(self, other):  
 return Number(self.value \*\* other.value)  
  
 # Relational operators  
 def \_\_eq\_\_(self, other):  
 return self.value == other.value  
  
 def \_\_ne\_\_(self, other):  
 return self.value != other.value  
  
 def \_\_lt\_\_(self, other):  
 return self.value < other.value  
  
 def \_\_le\_\_(self, other):  
 return self.value <= other.value  
  
 def \_\_gt\_\_(self, other):  
 return self.value > other.value  
  
 def \_\_ge\_\_(self, other):  
 return self.value >= other.value  
  
 def \_\_str\_\_(self):  
 return str(self.value)  
  
  
# Example Usage  
a = Number(10)  
b = Number(5)  
  
print("Addition:", a + b)  
print("Subtraction:", a - b)  
print("Multiplication:", a \* b)  
print("Division:", a / b)  
print("Floor Division:", a // b)  
print("Modulus:", a % b)  
print("Power:", a \*\* b)  
  
print("Equal:", a == b)  
print("Not Equal:", a != b)  
print("Less Than:", a < b)  
print("Less Than or Equal:", a <= b)  
print("Greater Than:", a > b)  
print("Greater Than or Equal:", a >= b)

**Output:**

****

**Conclusion:**

In this assignment, operator overloading was demonstrated for all arithmetic and relational operators in Python. This allows user-defined classes to interact with built-in operators seamlessly, enhancing code readability and flexibility.

==================================================================**GitHub:**

https://github.com/SushanThakur/2nd-sem-assignment/blob/master/assignments/assignment-4/code.py