Relational or Comparison Operators

4

Introduction to Python – 3.4

- Comparison operators
- https://docs.python.org/3/library/stdtypes.html#truth-value-testing

Operator	Meaning
<	strictly less than
<=	less than or equal
>	strictly greater than
>=	greater than or equal
==	equal
!=	not equal
is	object identity
is not	negated object identity

< Operator

- >>> 3<3.0
- False
- >>> 3==3.0
- True
- >>> 3==3
- True
- >>> 3=3
- **File "<stdin>", line 1**
- **SyntaxError: can't assign to literal**

> ,< & ==

- >>> 'KMIT' > 'kmit'
- False
- >>> 'kmit' > 'KMIT'
- True
- >>> 0.5 > False
- True
- >>> False == 0
- True
- >>> False < -10
- False
- >>> -10 < False
- True

- >>> 10<False
- False
- >>> 10<True
- False
- >>> True > -10
- True
- >>> True >0
- True
- >>> -10 == False
- False

> & <

- >>> 'KMIT' > 'kmit'
- False
- >>> 'kmit' > 'KMIT'
- True
- >>> 0.5 > False
- True
- >>> False == 0
- True
- >>> False < -10
- False
- >>> -10 < False
- True

- >>> 10<False
- False
- >>> 10<True
- False
- >>> True > -10
- True
- >>> True >0
- True
- >>> -10 == False
- False

> ,< & ==

- >>> from math import pi
- >>> pi
- 3.141592653589793
- >>> 3.1415 > pi
- False
- >>> pi > 3.1415
- True
- >>> False = 0

File "<stdin>", line 1

SyntaxError: can't assign to keyword

- >>> FALSE =0
- >>> 0 == FALSE
- True

- >>> 0 == false
- Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  NameError: name 'false' is not defined
- >>> 0 == False
- True
- >>> 0.0 == False
- True

Python Not Equal Operator (!=) Operator

- **>>> 3 != 3.0**
- **False**
- **>>> 0 == 0.0**
- **True**

Logical Operators

- Logical operators are the and, or, not operators.

Operator	Meaning	Example
and	True if both the operands are true	x and y
or	True if either of the operands is true	x or y
not	True if operand is false (complements the operand)	not x

Logical Operators

- Write a program using relational and logical operators to display the following:
- Give your age between 1 and 100? 44
- You have given True age.
- You are young False
- You are middle aged True
- You are Senior Citizen False
- You are Super Senior Citizen False

Logical Operators

- Give your age between 1 and 100? 0
- You have given False age.
- You are young False
- You are middle aged False
- You are Senior Citizen False
- You are Super Senior Citizen False

Logical Operators

```
age = int(input("Give your age between 1 and 100? "))
print("You have given ", age >=1 and age <=100, " age.",)
print("You are young " , (age > 0 and age <30))
print("You are middle aged " , (age >= 30 and age <60))
print("You are Senior Citizen " , (age > 60 and age <=80))
print("You are Super Senior Citizen " , (age > 80 ))
```

Introduction to Python – 3.4

- Let us assume a=10 and b=20

Operator	Function	Example
** Exponent	Performs exponential (power) calculation on operators	a**b =10 to the power 20

//	Floor Division - The division of operands where the result is the quotient in which the digits after the decimal point are removed.	9//2 = 4 and
		9.0//2.0 = 4.0
		>>> 90/25
		3.6
		>>> 90//25
		3
		>>> 90//25.0
		3.0

Introduction to Python – 3.4

- >>> 9//2
- 4
- >>> 9/2.2
- 4.090909090909091
- >>> 9/2.5
- 3.6
- >>> 9//2.5
- 3.0

Introduction to Python – 3.4

- Python Assignment Operators
- Assume variable a holds 10 and variable b holds 20.

Operator	Description	Example
=	Assigns values from right side operands to left side operand	c = a + b assigns value of a + b into c
+= Add AND	It adds right operand to the left operand and assign the result to left operand	c += a is equivalent to c = c + a
-= Subtract AND	It subtracts right operand from the left operand and assign the result to left operand	c -= a is equivalent to c = c - a
*= Multiply AND	It multiplies right operand with the left operand and assign the result to left operand	c *= a is equivalent to c = c * a

Introduction to Python – 3.4

- Python Assignment Operators
- Assume variable a holds 10 and variable b holds 20.

Operator	Description	Example
/= Divide AND	It divides left operand with the right operand and assign the result to left operand	c /= a is equivalent to c = c / a
%= Modulus AND	It takes modulus using two operands and assign the result to left operand	c %= a is equivalent to c = c % a
**= Exponent AND	Performs exponential (power) calculation on operators and assign value to the left operand	c **= a is equivalent to c = c ** a
//= Floor Division	It performs floor division on operators and assign value to the left operand	c //= a is equivalent to c = c // a

Introduction to Python – 3.4

- **=
- >>> x=5
- >>> x**=2
- >>> print(x)
- //=
- >>> x=5
- >>> x//=2
- >>> print(x)
- 2

Introduction to Python – 3.4

- To clear screen on windows
- import os
- os.system('cls') # on windows
- Subsequently we need to execute
- os.system('cls') # on windows

- >>> os.getcwd()
- #getcwd() is get Current Working Directory function
- 'C:\\Python34'

Exercise 2

- Write an interactive program which takes values of a,b & c from user.
- Then it displays the result whether
- "Under root >=0 " in terms of True or False
- Then it displays whether
- "Is a != 0" in terms of True or False
- It finally gives error or two values of roots.
- Use pow and sqrt function
- Hint: import math for sqrt()

Solution Exercise 2

```
import math
a = int(input('Enter value of a '))
b = int(input('Enter value of b '))
c = int(input('Enter value of c '))
print("Under root >=0 " , (math.sqrt(pow(b,2) - 4*a*c)) >= 0)
print("Is a != 0", a!=0)
print(-b + (math.sqrt(pow(b,2) - 4*a*c)/2*a))
print(-b - (math.sqrt(pow(b,2) - 4*a*c)/2*a))
```

Operators

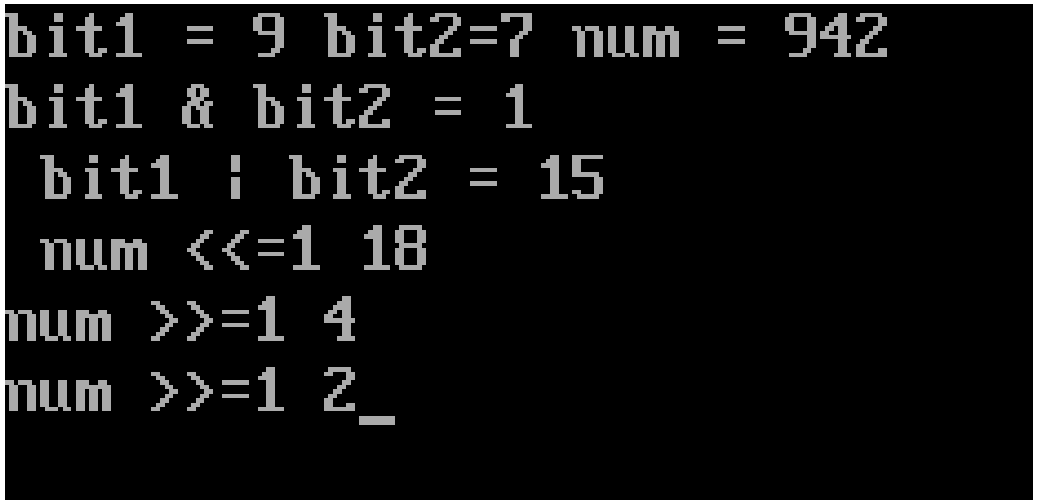
- **Bitwise Operators:**
- **<< Bitwise Left-shift**
- **Syntax operand op number**
- **Example if x = 0 0 0 1 1 1 0 1 then**
- **x<<1 produces 0 0 1 1 1 0 1 0**
- **>> Bitwise Right-shift operator**
- **Example if x = 0 0 0 1 0 0 1 0**
- **X >> 1 produces 0 0 0 0 1 0 0 1**
- **If x = 0001 1101 then what is x>>4?**
- **0000 0001**

Operators

- **Bitwise Operators:**
- **& Bitwise AND operator - | Bitwise OR operator**
- **Syntax operand op number**
- **Example if X = 9 (1001) Y = 7 (0111) then**
- **X 9 1001 1001**
- **Y 7 0111 0111**
- **--------- --------**
- **X&Y 0001 X|Y 1111**

Bitwise & | <<= >>=

```
bit1,bit2,num=9,7,942
print("bit1 = ",bit1," bit2= ",bit2," num = ",bit1,bit2,num)
num =bit1 & bit2;
print("\nbit1 & bit2 = ",num)
num = bit1 | bit2;
print("\n bit1 | bit2 = ",num);
num = bit1;
num <<=1;
print("\n num <<=1 ",num);
num = bit1;
num >>=1;
print("\nnum >>=1 ",num);
num >>=1;
print("\nnum >>=1 ",num)
```



Logical Operators