# Tutorials on Operators and Data Types

BENNETT UNIVERSITY
TIMES OF INDIA GROUP

## Comparison Operators (==,<,>,<=,>=)

| Operator | Example | Meaning | Result |
|---|---|---|---|
| == | a == b | Equal to | True if the value of a is equal to the value of b<br>False otherwise |
| != | a != b | Not equal to | True if a is not equal to b<br>False otherwise |
| < | a < b | Less than | True if a is less than b<br>False otherwise |
| <= | a <= b | Less than or equal to | True if a is less than or equal to b<br>False otherwise |
| > | a > b | Greater than | True if a is greater than b<br>False otherwise |
| >= | a >= b | Greater than or equal to | True if a is greater than or equal to b<br>False otherwise |

1. What will be the output of the following programs:

a.
```
a = 10
b = 20
print(a == b)
```

```
a. Solution       False
```

b.
```
a = 15
b = 25
print(a != b)
```

```
b. Solution       True
```

c.
```
a = 30
b = 15
```

print(a <= b)

print(a >= b)

c. Solution

```
False
True
```

d. a = 50

b = 35

print(a>b)

print(a<b)

d. Solution

```
True
False
```

**Arithmetic Operators (+,-,/,%,**)**

| Operator | Example | Meaning | Result |
|---|---|---|---|
| + (unary) | +a | Unary Positive | it doesn't really do anything. It mostly exists for the sake of completeness |
| + (binary) | a + b | Addition | Sum of a and b |
| - (unary) | -a | Unary Negation | Value equal to a but opposite in sign |
| - (binary) | a - b | Subtraction | b subtracted from a |
| * | a * b | Multiplication | Product of a and b |
| / | a / b | Division | Quotient when a is divided by b. The result always has type float. |
| % | a % b | Modulo | Remainder when a is divided by b |
| // | a // b | Floor Division (also called Integer Division) | Quotient when a is divided by b, rounded to the next smallest whole number |
| ** | a ** b | Exponentiation | a raised to the power of b |

1. Print the outputs:

a = 4

```
b = 3
print(+a)
print(-b)
print(a + b)
print(a - b)
print(a * b)
print(a / b)
```

1.Solution

```
4
-3
7
1
12
1.3333333333333333
```

2. Predict the outputs:

```
a=5
b=2
print(a % b)
print(a ** b)
print(10 / 4)
```

2. Solution

```
1
25
2.5
```

3. Predict the outputs:

```
a=9
b=3
print(a// b)
print(a// -b)
print(-a // b)
print(-a // -b)
```

3. Solution

```
3
-3
-3
3
```

## Logical Operators (not,or,and)

| Operator | Example | Meaning |
|---|---|---|
| not | not x | True if x is False<br>False if x is True<br>(Logically reverses the sense of x) |
| or | x or y | True if either x or y is True<br>False otherwise |
| and | x and y | True if both x and y are True<br>False otherwise |
| not in | x not in y | x not in y, here not in results in a 1 if x is not a member of sequence y |
| in | x in y | x in y, here in results in a 1 if x is a member of sequence y |

1.Predict the output

```
x = 15
y = 25
print(x > 10 or y < 8)
print(x > 10 or y > 8)
print(x < 10 or y > 8)
```

1.Solution

```
True
True
True
```

2.Predict the output

```
x = 8
y = 27
print(x > 10 and y < 8)
print(x > 10 and y > 8)
print(x < 10 and y > 8)
```

2. Solution

```
False
False
True
```

3.Predict the output

```
x = 5
y = 20
print(not x > 10 )
print(not x < 10)
print(not y > 8)
print(not y < 8)
```

3. Solution

```
True
False
False
True
```

4.Predict the output

```
x = 6
y = 21
print(not x > 10 and y > 8)
print(not x < 10 and y < 8)
```

4. Solution

```
True
False
```

5. Predict the output

```
x = 15

y = 25

z = 6

print(not x > 10 and y > 8 or z < 10)

print(x < 10 and not y > 6 or z < 5)
```

5. Solution

```
True
False
```

5. Predict the output

```
x = 24
y = 20
list = [10, 20, 30, 40, 50 ];

print(x not in list )
print(y not in list )
print(x in list )
print(y in list )
```

5. Solution

```
True
False
False
True
```

## Bitwise Operators (<<, >>, &, |, ~, and ^)

| Operator | Example | Meaning | Result |
|---|---|---|---|
| << | x << y | bits shifted to the left | Returns x with the bits shifted to the left by y places |
| >> | x >> y | bits shifted to the right | Returns x with the bits shifted to the right by y places |
| & | x & y | bitwise and | Each bit of the output is 1 if the corresponding bit of x AND of y is 1, otherwise it's 0 |
| \| | x \| y | bitwise or | Each bit of the output is 0 if the corresponding bit of x AND of y is 0, otherwise it's 1 |
| ~ | ~ x | complement of x | Returns the complement of x - the number you get by switching each 1 for a 0 and each 0 for a 1 |
| x ^ y | x ^ y | Bitwise XOR operator | Each bit of the output is the same as the corresponding bit in x if that bit in y is 0, and it's the complement of the bit in x if that bit in y is 1. |

1.
```
a = 10
b = 12
ans = a | b
print(ans)
```

1.Solution
```
14
```

2.
```
a = 5
b = 9
ans = a & b
print(ans)
```

2. Solution
```
1
```

3.
```
var = 2;
print("var = ", var<<0)
print("var = ", var<<1)
print("var = ", var<<2)
print("var = ", var<<3)
print("var = ", var<<4)
print("var = ", var<<5)
```

3. Solution

```
var =  2
var =  4
var =  8
var =  16
var =  32
var =  64
```

4.
```
var = 128;
print("var  = ", var>>0)
print("var  = ", var>>1)
print("var  = ", var>>2)
print("var  = ", var>>3)
print("var  = ", var>>4)
print("var  = ", var>>5)
```

4. Solution

```
var  =  128
var  =  64
var  =  32
var  =  16
var  =  8
var  =  4
```

5.
```
a = 5
b = 9
ans = a ^ b
print(ans)
```

5. Solution

```
12
```

6.

```
var = 3
print("value = ", ~var)
```

6. Solution

```
value =  -4
```

## Data Types (set,list,numbers,tuples)

| Data Type | Meaning |
|---|---|
| Booleans | Boolean in Python can have two values – True or False |
| Numbers | The numbers in Python are classified using the following keywords: **int, float, and complex**. |
| Strings | A sequence of one or more characters enclosed within either single quotes ‘ or double quotes ” is considered as String in Python. Any letter, a number or a symbol could be a part of the sting. |
| Lists | Lists in Python can be declared by placing elements inside **square brackets separated by commas**. |
| Tuples | A tuple is a heterogeneous collection of Python objects, using enclosing parentheses ( ) having its elements separated by commas inside. |
| Sets | A set is an unordered collection of unique and immutable objects. Its definition starts with enclosing braces { } having its items separated by commas inside. |
| Dictionaries | Python syntax for creating dictionaries use braces { } where each item appears as a pair of keys and values. |

1. Predict the outputs:

```
str = 'Learn Python'

print(type(str))

print(len(str))

print(len(str) == 12)

print(len(str) != 10)
```

1.Solution

```
<class 'str'>
12
True
True
```

2. Predict the outputs:

```
num = 2

print(type(num))

num = 3.0

print(type(num))

num = 3+5j

print(type(num))
```

2. Solution

```
<class 'int'>
<class 'float'>
<class 'complex'>
```

3. Predict the outputs:

```
my_string = 'Hello'
print(my_string)

my_string = "Hello"
print(my_string)

my_string = '''Hello'''
print(my_string)

my_string = """Hello, welcome to
        the world of Python"""
print(my_string)
print(my_string[5])
print(my_string[1:7])

print(my_string[2:5])
print(my_string[::-1])
```

3. Solution

```
Hello
Hello
Hello
```

```
Hello, welcome to
           the world of Python
,
ello,
llo
nohtyP fo dlrow eht
ot emoclew ,olleH
```

4. Predict the outputs:

```
assorted_list = [True, False, 1, 1.1, 1+2j, "Learn", "b","Python"]
first_element = assorted_list[0]
print(first_element)
first_element = assorted_list[3]
print(first_element)
first_element = assorted_list[3]
print(first_element)
print(assorted_list[5])
print(assorted_list)
```

4. Solution

```
True
1.1
1.1
Learn
[True, False, 1, 1.1, (1+2j), 'Learn', 'b', 'Python']
```

5. Predict the outputs:

```
first_tuple = (3, 5, 7, 9)
print(type(first_tuple))
print(first_tuple)
```

5. Solution

```
<class 'tuple'>
(3, 5, 7, 9)
```

6. Predict the outputs:

```
another_set = {"red", "green", "black"}
print(type(another_set))
print(another_set)
```

6. Solution

```
<class 'set'>
{'green', 'red', 'black'}
```

7. Predict the outputs:

```
sample_dict = {"key":"value", "jan":31, "feb":28, "mar":31}
print(type(sample_dict))
print(sample_dict)
```

7. Solution

```
<class 'dict'>
{'key': 'value', 'jan': 31, 'feb': 28, 'mar': 31}
```