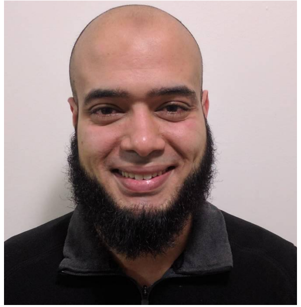
Python Programming

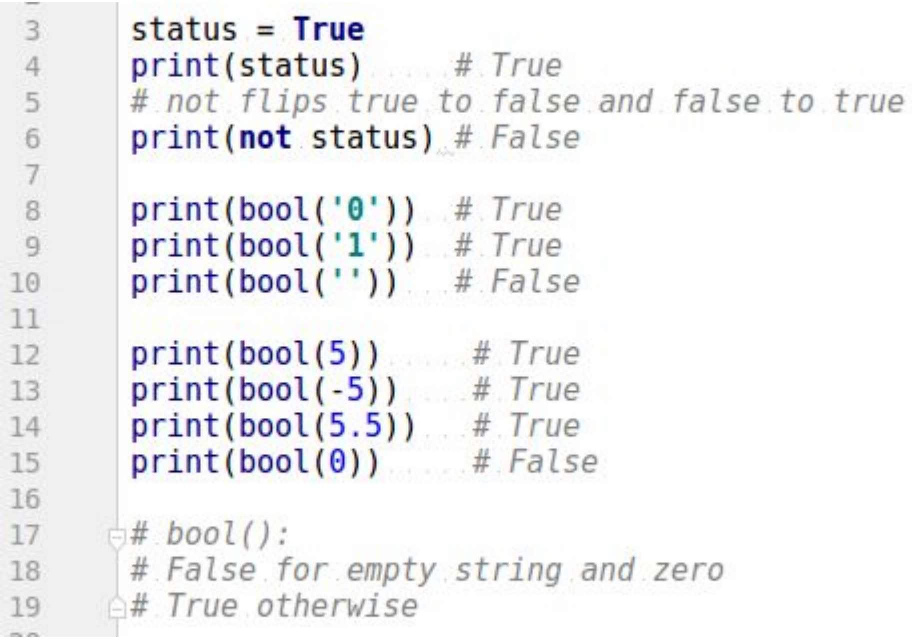
Relational Operators

Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)



Boolean data type



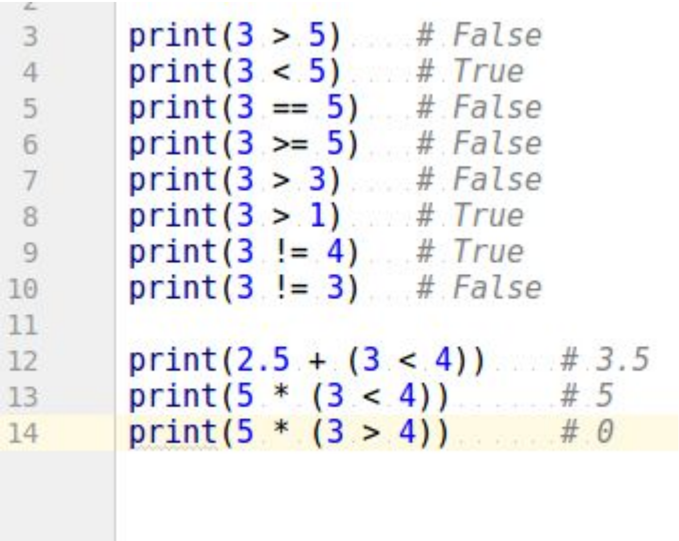
```
status = True
print(status)     # True
# not flips true to false and false to true
print(not status) # False

print(bool('0'))  # True
print(bool('1'))  # True
print(bool(''))   # False

print(bool(5))     # True
print(bool(-5))    # True
print(bool(5.5))   # True
print(bool(0))     # False

# bool():
# False for empty string and zero
# True otherwise
```

True or False?

Is 3 greater than 5? False	3 > 5
Is 3 less than 5? True	3 < 5
Is 3 equal to 5? False	3 == 5
Is 3 greater than or equal to 5? False	3 >= 5
Is 3 greater than or equal to 3? True	3 >= 3
Is 3 equal to 3? True	3 == 3
Is 3 greater than 1? True	3 > 1
Is 3 not equal to 4? True	3 != 4
Is 3 not equal to 3? False	3 != 3

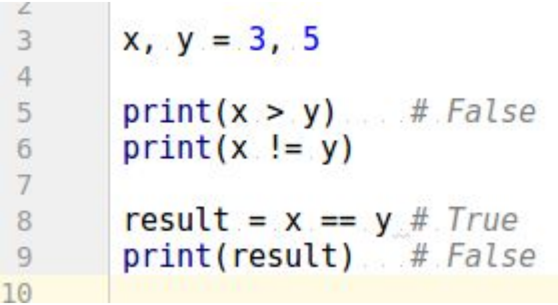
- Remember, we use bool for True and False conditions

Let’s code them



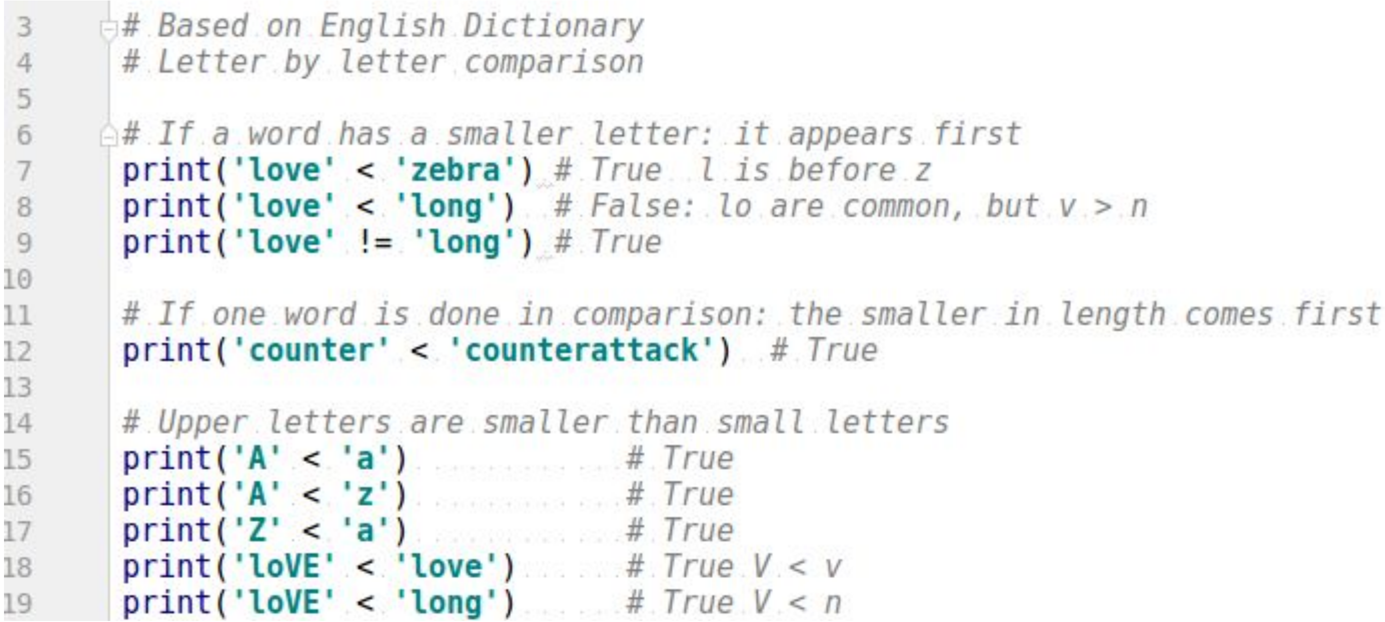
```
print(3 > 5)    # False
print(3 < 5)    # True
print(3 == 5)   # False
print(3 >= 5)   # False
print(3 > 3)    # False
print(3 > 1)    # True
print(3 != 4)   # True
print(3 != 3)   # False

print(2.5 + (3 < 4))    # 3.5
print(5 * (3 < 4))      # 5
print(5 * (3 > 4))      # 0
```

We can also use variables!



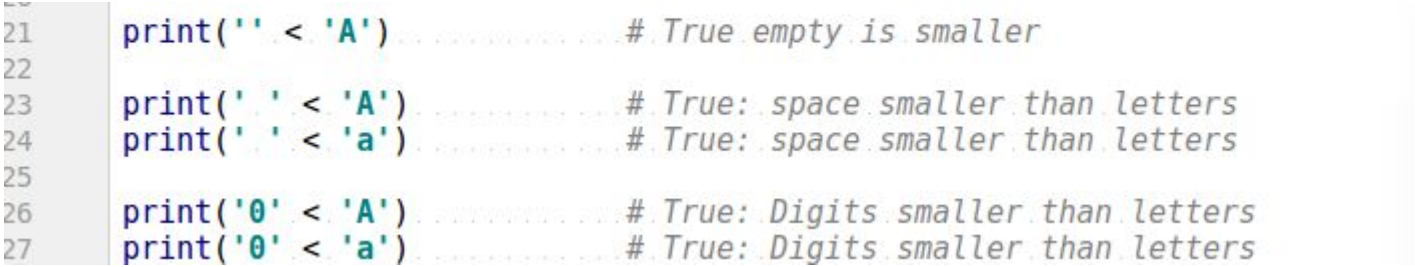
```python
x, y = 3, 5

print(x > y)    # False
print(x != y)

result = x == y # True
print(result)   # False

```

Comparing strings



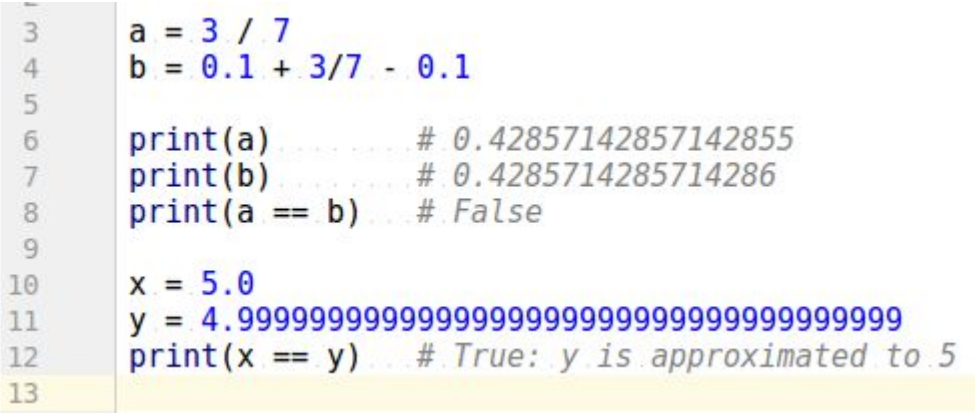
```
# Based on English Dictionary
# Letter by letter comparison

# If a word has a smaller letter: it appears first
print('love' < 'zebra') # True  l is before z
print('love' < 'long')  # False: lo are common, but v > n
print('love' != 'long') # True

# If one word is done in comparison: the smaller in length comes first
print('counter' < 'counterattack')  # True

# Upper letters are smaller than small letters
print('A' < 'a')            # True
print('A' < 'z')            # True
print('Z' < 'a')            # True
print('loVE' < 'love')      # True V < v
print('loVE' < 'long')      # True V < n
```

Comparing strings



```python
print('' < 'A')             # True empty is smaller

print(' ' < 'A')            # True: space smaller than letters
print(' ' < 'a')            # True: space smaller than letters

print('0' < 'A')            # True: Digits smaller than letters
print('0' < 'a')            # True: Digits smaller than letters
```

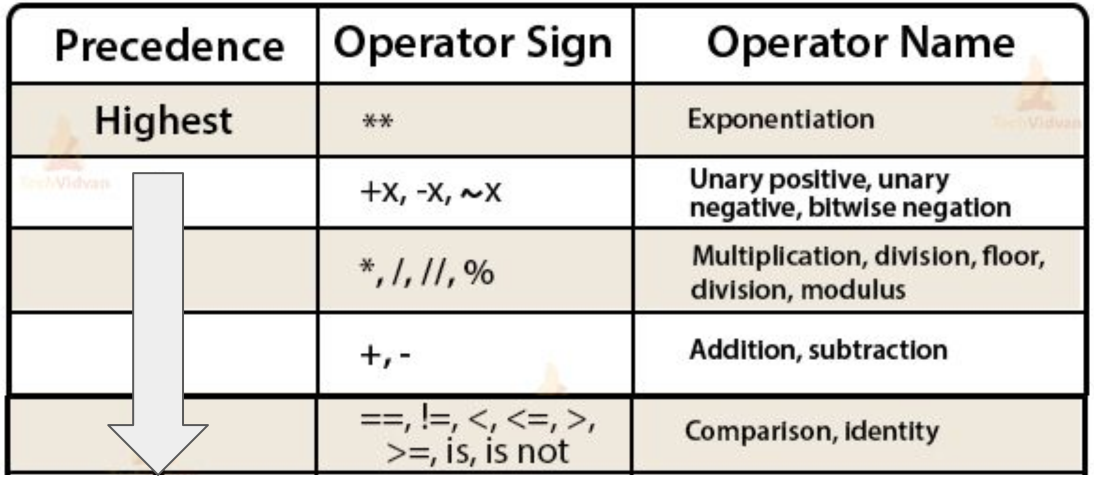
Comparing Floating point



```
a = 3 / 7
b = 0.1 + 3/7 - 0.1

print(a)        # 0.42857142857142855
print(b)        # 0.4285714285714286
print(a == b)   # False

x = 5.0
y = 4.9999999999999999999999999999999999999
print(x == y)   # True: y is approximated to 5
```

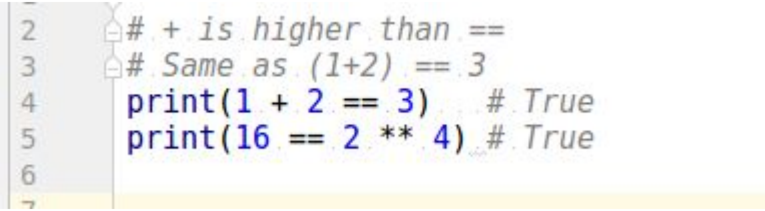
- Python has an approximate representation for floating point values
- Never compare directly
- Later we will learn how to do it

Operator Precedence



Precedence	Operator Sign	Operator Name
Highest	**	Exponentiation
	+X, -X, ~X	Unary positive, unary negative, bitwise negation
	*, /, //, %	Multiplication, division, floor, division, modulus
	+, -	Addition, subtraction
	==, !=, <, <=, >, >=, is, is not	Comparison, identity

Operator Precedence



```python
# + is higher than ==
# Same as (1+2) == 3
print(1 + 2 == 3)    # True
print(16 == 2 ** 4) # True
```

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."