

Comparison operators

INTERMEDIATE PYTHON FOR FINANCE



Kennedy Behrman

Data Engineer, Author, Founder

Python comparison operators

- Equality: `==` , `!=`
- Order: `<` , `>` , `<=` , `>=`

Equality operator vs assignment

Test equality: `==`

Assign value: `=`

Equality operator vs assignment

```
13 == 13
```

```
True
```

```
count = 13  
print(count)
```

```
13
```

Equality comparisons

- datetimes
- numbers (floats, ints)
- dictionaries
- strings
- almost anything else

Comparing datetimes

```
date_close_high = datetime(2019, 11, 27)
date_intra_high = datetime(2019, 11, 27)
print(date_close_high == date_intra_high)
```

True

Comparing dictionaries

```
d1 = {'high':56.88, 'low':33.22, 'closing':56.88}  
d2 = {'high':56.88, 'low':33.22, 'closing':56.88}  
print(d1 == d2)
```

True

```
d1 = {'high':56.88, 'low':33.22, 'closing':56.88}  
d2 = {'high':56.88, 'low':33.22, 'closing':12.89}  
print(d1 == d2)
```

False

Comparing different types

```
print(3 == 3.0)
```

True

```
print(3 == '3')
```

False

Not equal operator

```
print(3 != 4)
```

True

```
print(3 != 3)
```

False

Order operators

- Less than `<`
- Less than or equal `<=`
- Greater than `>`
- Greater than or equal `>=`

Less than operator

```
print(3 < 4)
```

```
True
```

```
print(3 < 3.6)
```

```
True
```

```
print('a' < 'b')
```

```
True
```

Less than operator

```
date_close_high = datetime(2019, 11, 27)
date_intra_high = datetime(2019, 11, 27)
print(date_close_high < date_intra_high)
```

False

Less than or equal operator

```
print(1 <= 4)
```

```
True
```

```
print(1.0 <= 1)
```

```
True
```

```
print('e' <= 'a')
```

```
False
```

Greater than operator

```
print(6 > 5)  
print(4 > 4)
```

True

False

Greater than or equal operator

```
print(6 >= 5)  
print(4 >= 4)
```

True

True

Order comparison across types

```
print(3.45454 < 90)
```

```
True
```

```
print('a' < 23)
```

```
<hr />-----
```

```
TypeError      Traceback (most recent call last)
```

```
...
```

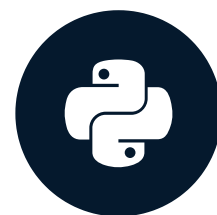
```
TypeError: '<' not supported between instances of 'str' and 'int'
```


Let's practice!

INTERMEDIATE PYTHON FOR FINANCE

Boolean operators

INTERMEDIATE PYTHON FOR FINANCE



Kennedy Behrman

Data Engineer, Author, Founder

Boolean logic



What are Boolean operations?

1. `and`
2. `or`
3. `not`

Object evaluation

Evaluates as False

- Constants:
 - `False`
 - `None`
- Numeric zero:
 - `0`
 - `0.0`
- Length of zero
 - `""`
 - `[]`
 - `{}`

Evaluates as True

- Almost everything else

The AND operator

True and True

True

True and False

False

The OR operator

False or True

True

True or True

True

False or False

False

Short circuit.

```
is_current() and is_investment()
```

False

```
is_current() or is_investment()
```

True

The NOT operator

```
not True
```

```
False
```

```
not False
```

```
True
```

Order of operations with NOT

```
True == False
```

```
False
```

```
not True == False
```

```
True
```

Object evaluation

"CUSIP" and True

True

Object evaluation

```
[] or False
```

```
False
```

Object evaluation

```
not {}
```

```
True
```

Returning objects

```
"Federal" and "State"
```

```
"State"
```

```
[] and "State"
```

```
[]
```

Returning objects.

```
13 or "account number"
```

```
13
```

```
0.0 or {"balance": 2200}
```

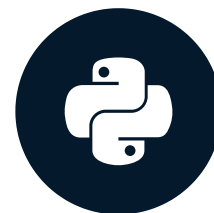
```
{"balance": 2200}
```

Let's practice!

INTERMEDIATE PYTHON FOR FINANCE

If statements

INTERMEDIATE PYTHON FOR FINANCE



Kennedy Behrman

Data Engineer, Author, Founder

Printing sales only

```
trns = { 'symbol': 'TSLA', 'type': 'BUY', 'amount': 300 }
```

```
print(trns['amount'])
```

```
300
```

Compound statements

```
control statement
```

```
    statement 1
```

```
    statement 2
```

```
    statement 3
```

Control Statement

```
if <expression> :
```

```
if x < y:
```

```
if x in y:
```

```
if x and y:
```

```
if x:
```

Code blocks

```
if <expression>:  
    statement  
    statement  
    statement
```

```
if <expression>: statement;statement;statement
```

Printing sales only

```
trns = { 'symbol': 'TSLA', 'type': 'BUY', 'amount': 300 }
```

```
if trns['type'] == 'SELL':  
    print(trns['amount'])
```

```
trns['type'] == 'SELL'
```

False

Printing sales only.

```
trns = { 'symbol': 'APPL', 'type': 'SELL', 'amount': 200 }
```

```
if trns['type'] == 'SELL':  
    print(trns['amount'])
```

```
200
```

Else

```
if x in y:  
    print("I found x in y")  
else:  
    print("No x in y")
```


Elif

```
if x == y:  
    print("equals")  
elif x < y:  
    print("less")
```

Elif

```
if x == y:  
    print("equals")  
elif x < y:  
    print("less")  
elif x > y:  
    print("more")  
elif x == 0:  
    print("zero")
```

Else with elif

```
if x == y:  
    print("equals")  
elif x < y:  
    print("less")  
elif x > y:  
    print("more")  
elif x == 0:  
    print("zero")  
else:  
    print("None of the above")
```

Let's practice!

INTERMEDIATE PYTHON FOR FINANCE

For and while loops

INTERMEDIATE PYTHON FOR FINANCE



Kennedy Behrman

Data Engineer, Author, Founder

Repeating a code block

CUSIP

037833100

17275R102

68389X105

SYMBOL

AAPL

CSCO

ORCL

Loops.

For loop

While loop

Statement components

```
<Control Statement>  
    <Code Block>
```

```
execution 1
```

```
execution 2
```

```
execution 3
```


For loops

```
for <variable> in <sequence>:
```

```
for x in [0, 1, 2]:
```

```
d = {'key': 'value1'}  
for x in d:
```

```
for x in "ORACLE":
```

List example

```
for x in [0, 1, 2]:  
    print(x)
```

0

1

2

Dictionary example

```
symbols = {'037833100': 'AAPL',  
           '17275R102': 'CSCO',  
           '68389X105': 'ORCL'}  
  
for k in symbols:  
    print(symbols[k])
```

```
AAPL  
CSCO  
ORCL
```

String example

```
for x in "ORACLE":  
    print(x)
```

```
O  
R  
A  
C  
L  
E
```

While control statements

```
while <expression>:
```

While example

```
x = 0

while x < 5:
    print(x)
    x = (x + 1)
```

```
0
1
2
3
4
```

Infinite loops

```
x = 0

while x <= 5:
    print(x)
```

Skipping with continue

```
for x in [0, 1, 2, 3]:  
    if x == 2:  
        continue  
    print(x)
```

0

1

3

Stopping with break.

```
while True:
    transaction = get_transaction()
    if transaction['symbol'] == 'ORCL':
        print('The current symbol is ORCL, break now')
        break
    print('Not ORCL')
```

```
Not ORCL
Not ORCL
Not ORCL
The current symbol is ORCL, break now
```

Let's practice 'for' and 'while' loops!

INTERMEDIATE PYTHON FOR FINANCE