INTERMEDIATE PYTHON FOR DATA SCIENCE

Comparison Operators

Numpy Recap

```
In [1]: import numpy as np
In [2]: np_height = np.array([1.73, 1.68, 1.71, 1.89, 1.79])
In [3]: np_weight = np.array([65.4, 59.2, 63.6, 88.4, 68.7])
In [4]: bmi = np_weight / np_height ** 2

In [5]: bmi
Out[5]: array([ 21.852,  20.975,  21.75 ,  24.747,  21.441])

In [6]: bmi > 23
Out[6]: array([False, False, False,  True, False], dtype=bool)

In [7]: bmi[bmi > 23]
Out[7]: array([ 24.747])
```

Code from Intro to Python for Data Science, Chapter 4

Comparison operators: how Python values relate

Numeric Comparisons

```
In [8]: 2 < 3
Out[8]: True

In [9]: 2 == 3
Out[9]: False

In [10]: 2 <= 3
Out[10]: True

In [11]: 3 <= 3
Out[11]: True

In [12]: x = 2

In [13]: y = 3

In [14]: x < y
Out[14]: True
```

Other Comparisons

```
In [15]: "carl" < "chris"
Out[15]: True

In [16]: 3 < "chris"
TypeError: unorderable types: int() < str()

In [17]: 3 < 4.1
Out[17]: True

In [18]: bmi
Out[18]: array([ 21.852,  20.975,  21.75 ,  24.747,  21.441])

In [19]: bmi > 23
Out[19]: array([False, False, False,  True, False], dtype=bool)
```

Comparators

<	strictly less than
<=	less than or equal
>	strictly greater than
>=	greater than or equal
==	equal
! =	not equal

INTERMEDIATE PYTHON FOR DATA SCIENCE

Let's practice!

INTERMEDIATE PYTHON FOR DATA SCIENCE

Boolean Operators

Boolean Operators

- `and`
- `or`
- `not`

and

```
In [1]: True and True
Out[1]: True

In [2]: False and True
Out[2]: False

In [3]: True and False
Out[3]: False

In [4]: False and False
Out[4]: False

In [5]: x = 12
          True        True
In [6]: x > 5 and x < 15
Out[6]: True
```

or

```
In [7]: True or True
Out[7]: True

In [8]: False or True
Out[8]: True

In [9]: True or False
Out[9]: True

In [10]: False or False
Out[10]: False

In [11]: y = 5
           True       False
In [12]: y < 7 or y > 13
Out[12]: True
```

not

```
In [13]: not True
Out[13]: False

In [14]: not False
Out[14]: True
```

Numpy

```
In [19]: bmi     # calculation of bmi left out
Out[19]: array([ 21.852,  20.975,  21.75 ,  24.747,  21.441])

In [20]: bmi > 21
Out[20]: array([ True, False,  True,  True,  True], dtype=bool)

In [21]: bmi < 22
Out[22]: array([ True,  True,  True, False,  True], dtype=bool)

In [23]: bmi > 21 and bmi < 22
ValueError: The truth value of an array with more than one element
is ambiguous. Use a.any() or a.all()
```

Numpy

```
In [19]: bmi     # calculation of bmi left out
Out[19]: array([ 21.852,  20.975,  21.75 ,  24.747,  21.441])

In [20]: bmi > 21
Out[20]: array([ True, False,  True,  True,  True], dtype=bool)

In [21]: bmi < 22
Out[22]: array([ True,  True,  True, False,  True], dtype=bool)

In [23]: bmi > 21 and bmi < 22
ValueError: The truth value of an array with more than one element
is ambiguous. Use a.any() or a.all()

In [24]: np.logical_and(bmi > 21, bmi < 22)
Out[24]: array([ True, False,  True, False,  True], dtype=bool)

In [25]: bmi[np.logical_and(bmi > 21, bmi < 22)]
Out[25]: array([ 21.852,  21.75,  21.441])
```

logical_and()
logical_or()
logical_not()

INTERMEDIATE PYTHON FOR DATA SCIENCE

Let's practice!

INTERMEDIATE PYTHON FOR DATA SCIENCE

if, elif, else

Overview

- Comparison Operators
  - `<, >, >=, <=, ==, !=`
- Boolean Operators
  - `and, or, not`
- Conditional Statements
  - `if, else, elif`

if

```
if condition :
    expression
```

control.py

```
z = 4
if z % 2 == 0 :
    print("z is even")
```

True

Output:

```
z is even
```

if

```
if condition :
    expression
expression # not part of if
```

control.py

```
z = 4
if z % 2 == 0 :
    print("z is even")
```

True

Output:

```
z is even
```

if

```
if condition :
    expression
```

control.py

```
z = 4
if z % 2 == 0 :
    print("checking "  + str(z))
    print("z is even")
```

Output:

```
checking 4
z is even
```

if

```
if condition :
    expression
```

control.py

```
z = 5
if z % 2 == 0 :
    print("checking "  + str(z))
    print("z is even")
```

False

Not executed

```
Output:
```

else

```
if condition :
    expression
else :
    expression
```

control.py

```
z = 5
if z % 2 == 0 :
    print("z is even")
else :
    print("z is odd")
```

False

Output:

```
z is odd
```

elif

```
if condition :
    expression
elif condition :
    expression
else :
    expression
```

control.py

```
z = 3
if z % 2 == 0 :
    print("z is divisible by 2")
elif z % 3 == 0 :
    print("z is divisible by 3")
else :
    print("z is neither divisible by 2 nor by 3")
```

False

True

Output:

```
z is divisible by 3
```

elif

```
if condition :
    expression
elif condition :
    expression
else :
    expression
```

control.py

```
z = 6
if z % 2 == 0 :  True
    print("z is divisible by 2")
elif z % 3 == 0 :   Never reached
    print("z is divisible by 3")
else :
    print("z is neither divisible by 2 nor by 3")
```

```
Output:
z is divisible by 2
```

INTERMEDIATE PYTHON FOR DATA SCIENCE

Let's practice!

INTERMEDIATE PYTHON FOR DATA SCIENCE

Filtering Pandas DataFrame

brics

```
In [1]: import pandas as pd

In [2]: brics = pd.read_csv("path/to/brics.csv", index_col = 0)

In [3]: brics
Out[3]:
         country    capital    area  population
BR        Brazil   Brasilia   8.516      200.40
RU        Russia     Moscow  17.100      143.50
IN         India  New Delhi   3.286     1252.00
CH         China    Beijing   9.597     1357.00
SA  South Africa   Pretoria   1.221       52.98
```

Goal

- Select countries with area over 8 million km²
- 3 steps
  - Select the area column
  - Do comparison on area column
  - Use result to select countries

```
        country    capital    area  population
BR       Brazil   Brasilia   8.516      200.40
RU       Russia     Moscow  17.100      143.50
IN        India  New Delhi   3.286     1252.00
CH        China    Beijing   9.597     1357.00
SA  South Africa   Pretoria   1.221       52.98
```

Step 1: Get column

	country	capital	area	population
BR	Brazil	Brasilia	8.516	200.40
RU	Russia	Moscow	17.100	143.50
IN	India	New Delhi	3.286	1252.00
CH	China	Beijing	9.597	1357.00
SA	South Africa	Pretoria	1.221	52.98

```
In [4]: brics["area"]
Out[4]:
BR     8.516
RU    17.100
IN     3.286
CH     9.597
SA     1.221
Name: area, dtype: float64
```

Alternatives:

```
brics.loc[:,"area"]

brics.iloc[:,2]
```

Need Pandas Series

Step 2: Compare

```
          country    capital    area  population
BR         Brazil   Brasilia   8.516      200.40
RU         Russia     Moscow  17.100      143.50
IN          India  New Delhi   3.286     1252.00
CH          China    Beijing   9.597     1357.00
SA   South Africa    Pretoria   1.221       52.98
```

```
In [4]: brics["area"]
Out[4]:
BR     8.516
RU    17.100
IN     3.286
CH     9.597
SA     1.221
Name: area, dtype: float64

In [5]: brics["area"] > 8
Out[5]:
BR     True
RU     True
IN    False
CH     True
SA    False
Name: area, dtype: bool

In [6]: is_huge = brics["area"] > 8
```

Step 3: Subset DF

	country	capital	area	population
BR	Brazil	Brasilia	8.516	200.40
RU	Russia	Moscow	17.100	143.50
IN	India	New Delhi	3.286	1252.00
CH	China	Beijing	9.597	1357.00
SA	South Africa	Pretoria	1.221	52.98

```
In [7]: is_huge
Out[7]:
BR     True
RU     True
IN    False
CH     True
SA    False
Name: area, dtype: bool

In [8]: brics[is_huge]
Out[8]:
   country   capital    area  population
BR  Brazil  Brasilia   8.516       200.4
RU  Russia    Moscow  17.100       143.5
CH   China   Beijing   9.597      1357.0
```

Summary

```
In [9]: is_huge = brics["area"] > 8

In [10]: brics[is_huge]
Out[10]:
   country   capital    area  population
BR  Brazil  Brasilia   8.516       200.4
RU  Russia    Moscow  17.100       143.5
CH   China   Beijing   9.597      1357.0

In [11]: brics[brics["area"] > 8]
Out[11]:
   country   capital    area  population
BR  Brazil  Brasilia   8.516       200.4
RU  Russia    Moscow  17.100       143.5
CH   China   Beijing   9.597      1357.0
```

	country	capital	area	population
BR	Brazil	Brasilia	8.516	200.40
RU	Russia	Moscow	17.100	143.50
IN	India	New Delhi	3.286	1252.00
CH	China	Beijing	9.597	1357.00
SA	South Africa	Pretoria	1.221	52.98

Boolean operators

	country	capital	area	population
BR	Brazil	Brasilia	8.516	200.40
RU	Russia	Moscow	17.100	143.50
IN	India	New Delhi	3.286	1252.00
CH	China	Beijing	9.597	1357.00
SA	South Africa	Pretoria	1.221	52.98

```
In [12]: import numpy as np

In [13]: np.logical_and(brics["area"] > 8, brics["area"] < 10)
Out[13]:
BR     True
RU    False
IN    False
CH     True
SA    False
Name: area, dtype: bool

In [14]: brics[np.logical_and(brics["area"] > 8, brics["area"] < 10)]
Out[14]:
   country   capital   area  population
BR  Brazil  Brasilia  8.516       200.4
CH   China   Beijing  9.597      1357.0
```

INTERMEDIATE PYTHON FOR DATA SCIENCE

Let's practice!