

# Numpy recap

```
# Code from Intro to Python for Data Science, Chapter 4
import numpy as np
np_height = np.array([1.73, 1.68, 1.71, 1.89, 1.79])
np_weight = np.array([65.4, 59.2, 63.6, 88.4, 68.7])
bmi = np_weight / np_height ** 2
bmi
```

```
array([ 21.852,  20.975,  21.75 ,  24.747,  21.441])
```

```
bmi > 23
```

```
array([False, False, False,  True, False], dtype=bool)
```

```
bmi[bmi > 23]
```

```
array([ 24.747])
```

- Comparison operators: how Python values relate

# Numeric comparisons

```
2 < 3
```

```
True
```

```
2 == 3
```

```
False
```

```
2 <= 3
```

```
True
```

```
3 <= 3
```

```
True
```

```
x = 2  
y = 3  
x < y
```

```
True
```

# Other comparisons

```
"carl" < "chris"
```

```
True
```

```
3 < "chris"
```

```
TypeError: unorderable types: int() < str()
```

```
3 < 4.1
```

```
True
```

# Other comparisons

```
bmi
```

```
array([21.852, 20.975, 21.75 , 24.747, 21.441])
```

```
bmi > 23
```

```
array([False, False, False, True, False], dtype=bool)
```

# Comparators

Comparator	Meaning
<	Strictly less than
<=	Less than or equal
>	Strictly greater than
>=	Greater than or equal
==	Equal
!=	Not equal

# Let's practice!

INTERMEDIATE PYTHON

# Boolean Operators

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

# and

True and True

True

```
x = 12
x > 5 and x < 15
# True     True
```

True

False and True

False

True and False

False

False and False

False



# or

True or True

True

False or True

True

True or False

True

False or False

False

```
y = 5  
y < 7 or y > 13
```

True

# not

```
not True
```

```
False
```

```
not False
```

```
True
```

# NumPy

```
bmi      # calculation of bmi left out
```

```
array([21.852, 20.975, 21.75 , 24.747, 21.441])
```

```
bmi > 21
```

```
array([True, False, True, True, True], dtype=bool)
```

```
bmi < 22
```

```
array([True, True, True, False, True], dtype=bool)
```

```
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

- `logical_and()`
- `logical_or()`
- `logical_not()`

```
np.logical_and(bmi > 21, bmi < 22)
```

```
array([True, False, True, False, True], dtype=bool)
```

```
bmi[np.logical_and(bmi > 21, bmi < 22)]
```

```
array([21.852, 21.75, 21.441])
```

# Let's practice!

INTERMEDIATE PYTHON

# 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 :    # True  
    print("z is even")
```

```
z is even
```

# if

```
if condition :  
    expression
```

- `expression` not part of `if`

`control.py`

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

```
z is even
```



# if

```
if condition :  
    expression
```

control.py

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

```
checking 4  
z is even
```

# if

```
if condition :  
    expression
```

control.py

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

# else

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

control.py

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

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

```
z is divisible by 3
```

# elif

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

control.py

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

```
z is divisible by 2
```

# Let's practice!

INTERMEDIATE PYTHON

# brics

```
import pandas as pd  
brics = pd.read_csv("path/to/brics.csv", index_col = 0)  
brics
```

	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

	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

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



# 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

```
brics["area"]
```

```
BR      8.516
RU     17.100
IN      3.286
CH      9.597
SA      1.221
Name: area, dtype: float64    # - Need Pandas Series
```

- Alternatives:

```
brics.loc[:, "area"]
brics.iloc[:, 2]
```

# Step 2: Compare

```
brics["area"]
```

```
BR      8.516  
RU     17.100  
IN      3.286  
CH      9.597  
SA      1.221  
Name: area, dtype: float64
```

```
brics["area"] > 8
```

```
BR      True  
RU      True  
IN     False  
CH      True  
SA     False  
Name: area, dtype: bool
```

```
is_huge = brics["area"] > 8
```

# Step 3: Subset DF

```
is_huge
```

```
BR    True
RU    True
IN    False
CH    True
SA    False
Name: area, dtype: bool
```

```
brics[is_huge]
```

	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

	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	52.988

```
is_huge = brics["area"] > 8
brics[is_huge]
```

	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

```
brics[brics["area"] > 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

# 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
```

```
import numpy as np
np.logical_and(brics["area"] > 8, brics["area"] < 10)
```

```
BR      True
RU      False
IN      False
CH      True
SA      False
Name: area, dtype: bool
```

```
brics[np.logical_and(brics["area"] > 8, brics["area"] < 10)]
```

```
country  capital  area  population
BR  Brazil  Brasilia  8.516      200.4
CH   China   Beijing  9.597     1357.0
```

# Let's practice!

INTERMEDIATE PYTHON