UCLA College | Social Sciences Economics

# Day 4: Classes, File Operations, Modules & Methods

Paul Schumacher, MSc Quantitative Economics

# Announcements

~~Attendance~~

**Midterm**

- Take Home + Open notes
- NOT Proctored
- 4:00 – 6:00 pm on Monday
- Upload your code

# Midterm Review

## I. Coding Problems

- Everything we learned so far *excluding* File Operations, Modules & Methods
- Questions examples:
  - “Initialize a list…”
  - “Construct a dictionary”
  - “Using order of operations…”
  - “Perform the following operations…” → application of functions
  - “Create an if statement…”
  - “Create a for loop…”
  - “Create a while loop…”
  - ”Write a function…”
  - “Write a class…”

# Midterm Review

## II. Multiple Choice Questions

- Everything we learned so far *excluding* File Operations, Modules & Methods
- Questions examples:
  - "What happens if we apply this code?" → *put code in console*
  - "Which Python statement can be used to…"
  - Google is your friend (but time is limited)

# Del vs. pop vs. remove

```
fruits = ['apple', 'banana', 'cherry']
fruits.remove('apple')
print(fruits)
```

```
['banana', 'cherry']
```

```
my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9]

# deleting the third item
del my_list[2]

# Output: [1, 2, 4, 5, 6, 7, 8, 9]
print(my_list)

# deleting items from 2nd to 4th
del my_list[1:4]

# Output: [1, 6, 7, 8, 9]
print(my_list)

# deleting all elements
del my_list[:]

# Output: []
print(my_list)
```

removes item/ slices at a **given index**

For both (del, pop):
The index of item is needed!!!
Not direct removal of value

Never used it in my life!!

```
fruits = ['apple', 'banana', 'cherry']
x = fruits.pop(2)
print(fruits)
```

```
['apple', 'banana']
```

pop() method removes the element at the **specified position**

same

= del fruits[2]

```
# create a list
prime_numbers = [2, 3, 5, 7, 9, 11]

# remove 9 from the list
prime_numbers.remove(9)

# Updated prime_numbers List
print('Updated List: ', prime_numbers)

# Output: Updated List:  [2, 3, 5, 7, 11]
```

For remove: no index of item is needed!!! → direct removal of value

Removes first "matching" item that is given in the brackets

# Del vs. pop vs. remove

This is about Lists...

Never used it in my life!!

```
my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9]

# deleting the third item
del my_list[2]

# Output: [1, 2, 4, 5, 6, 7, 8, 9]
print(my_list)

# deleting items from 2nd to 4th
del my_list[1:4]

# Output: [1, 6, 7, 8, 9]
print(my_list)

# deleting all elements
del my_list[:]

# Output: []
print(my_list)
```

removes item/ slices at a **given index**

For both (del, pop):
The index of item is needed!!!
Not direct removal of value

```
fruits = ['apple', 'banana', 'cherry']

x = fruits.pop(2)

print(fruits)
```

```
['apple', 'banana']
```

pop() method removes the element at the **specified position**

same

= del fruits[2]

```
fruits = ['apple', 'banana', 'cherry']

fruits.remove('apple')

print(fruits)
```

```
['banana', 'cherry']
```

```
# create a list
prime_numbers = [2, 3, 5, 7, 9, 11]

# remove 9 from the list
prime_numbers.remove(9)

# Updated prime_numbers List
print('Updated List: ', prime_numbers)

# Output: Updated List:  [2, 3, 5, 7, 11]
```

For remove: no index of item is needed!!! → direct removal of value

Removes first "matching" item that is given in the brackets

# Ways to remove a key from dictionary

- **Method 1: del → direct removal of value**

```
test_dict = {"Arushi": 22, "Anuradha": 21, "Mani": 21, "Haritha": 21}
del test_dict['Mani']
test_dict
```

```
{'Arushi': 22, 'Anuradha': 21, 'Haritha': 21}
```

- **Method 2: pop() → direct removal of value**

```
test_dict = {"Arushi": 22, "Anuradha": 21, "Mani": 21, "Haritha": 21}
test_dict.pop('Mani')
test_dict
```

```
{'Arushi': 22, 'Anuradha': 21, 'Haritha': 21}
```

# Quick Intro – 3 Students

1. Name, School, Hobbies, …
2. What are your future plans (university, work, hobbies, travel)
3. Experience in Python? Why do you want to learn it? What do you wish to get out of this course?
4. Which apps do you use most on your phone?
5. What was the best/ worst thing that happened to you last month?

# Class exercise: Functions

- “PLF Day 4 Worksheet A.ipynb”
- Breakout rooms
- Time: 15 min

# Class Recap: I. Classes

- Logically group data (“attributes”) and functions (“methods”) → save space & time
- Create *blueprint* for data input (”instance”)

```python
class Person:
    def __init__(self, age, weight, height, first_name, last_name):
        self.age = age
        self.weight = weight
        self.height = height
        self.first_name = first_name
        self.last_name = last_name

    def fullname(self):
        return '{} {}'.format(self.first_name, self.last_name)
```

arguments

methods

instance

```python
user = Person(20, 180, 6.0, "Alex", "Song")
```

```python
print(user.height)
```

```
6.0
```

attributes

```python
user.fullname()
```

```
'Alex Song'
```

# Class Recap: I. Classes

__init__: ”initialize”
Sees *self* as instance, rest as arguments

self: “placeholder” for instance (*user*)

```
class Person:
    def __init__(self, age, weight, height, first_name, last_name):
        self.age = age
        self.weight = weight
        self.height = height
        self.first_name = first_name
        self.last_name = last_name

    def fullname(self):
        return '{} {}'.format(self.first_name, self.last_name)
```

```
user = Person(20, 180, 6.0, "Alex", "Song")
```

# Class Recap: I. Classes

https://www.youtube.com/watch?v=ZDa-Z5JzLYM&ab_channel=CoreySchafer

__init__: ”initialize”
Sees *self* as instance, rest as arguments

self: “placeholder” for instance (*user*)

```
class Person:
    def __init__(self, age, weight, height, first_name, last_name):
        self.age = age
        self.weight = weight
        self.height = height
        self.first_name = first_name
        self.last_name = last_name

    def fullname(self):
        return '{} {}'.format(self.first_name, self.last_name)
```

```
user = Person(20, 180, 6.0, "Alex", "Song")
```

# Class exercise: Classes

- “PLF Day 4 Worksheet B.ipynb”
- Breakout rooms
- Time: 15 min

# Class Recap: IV. File Ops., Modules & Methods

```
pip install wooldridge
```

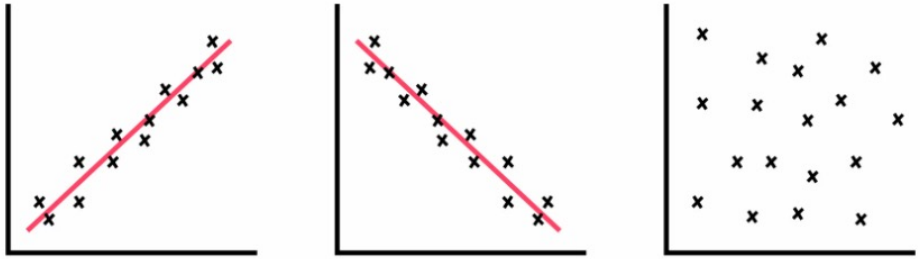
```
Collecting wooldridge
  Using cached wooldridge-0.4.4-py3-none-any.whl (5.1 MB)
Collecting pandas
  Downloading pandas-2.0.3-cp310-cp310-macosx_10_9_x86_64.whl (11.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 11.8/11.8 MB 11.6 MB/s eta
0:00:0000:0100:01
Requirement already satisfied: numpy>=1.21.0 in /Users/paulschumacher/ana
conda3/lib/python3.10/site-packages (from pandas->wooldridge) (1.25.0)
```

```
pip install pandas_datareader
```

```
Collecting pandas_datareader
  Using cached pandas_datareader-0.10.0-py3-none-any.whl (1
Requirement already satisfied: lxml in /Users/paulschumache
b/python3.10/site-packages (from pandas_datareader) (4.9.2)
```

.mean()
.median()
.std()
.describe()

# Class Recap: IV. File Ops., Modules & Methods

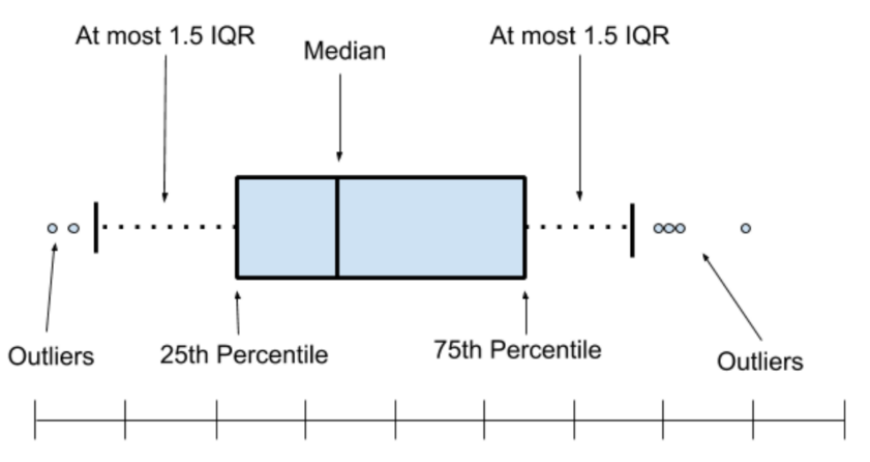


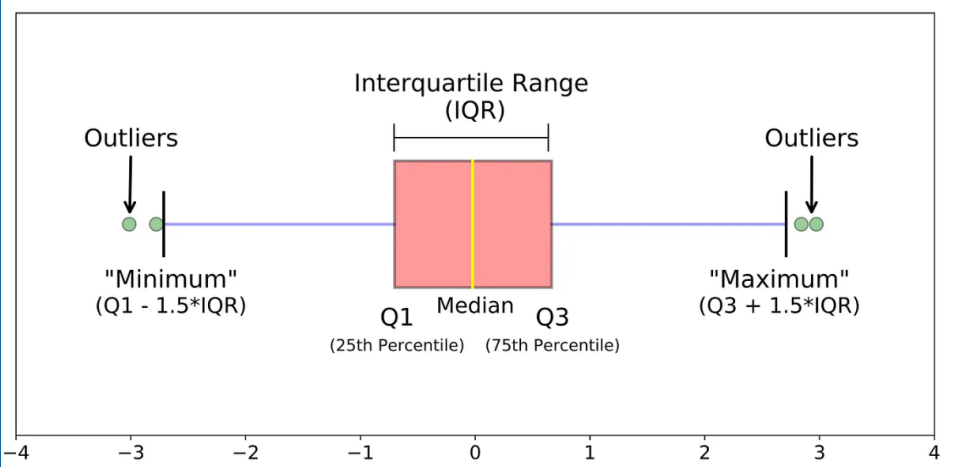

```python
# Find the correlation between the two stock prices.
np.corrcoef(price['AAPL'], price['AMZN'])
```

```
array([[1.        , 0.8944954],
       [0.8944954, 1.        ]])
```

# Class Recap: IV. File Ops., Modules & Methods

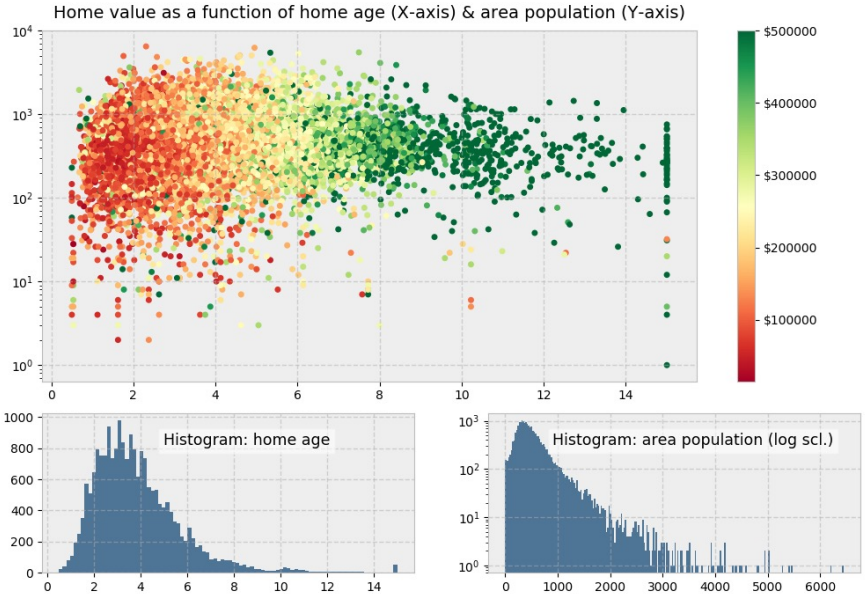
.boxplot()



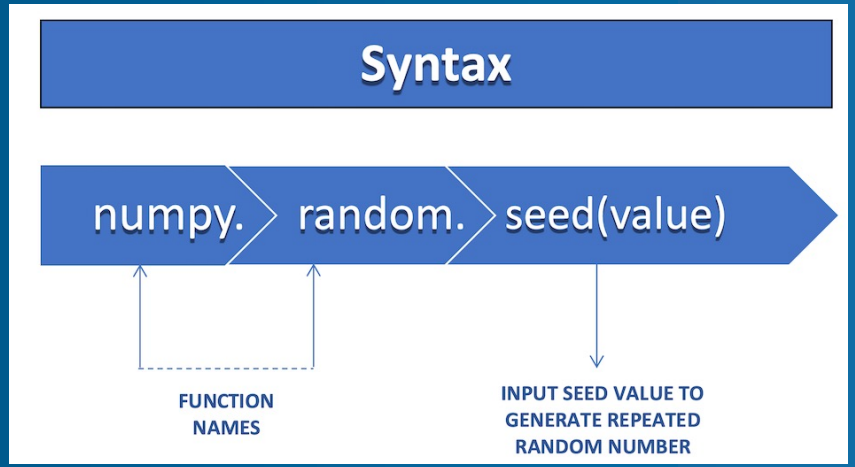



# Class Recap: IV. File Ops., Modules & Methods

matplotlib



# Class Recap: IV. File Ops., Modules & Methods




- Choose 5 number from 1 to 100
  - Seed 1 (3, 45, 12, 8, 99)
  - Seed 2 (4, 34, 71, 92, 2)
  - Seed 3 (82, 41, 65, 44, 3)
  - Seed 4 (…)

Choose same random numbers

# Any Questions?