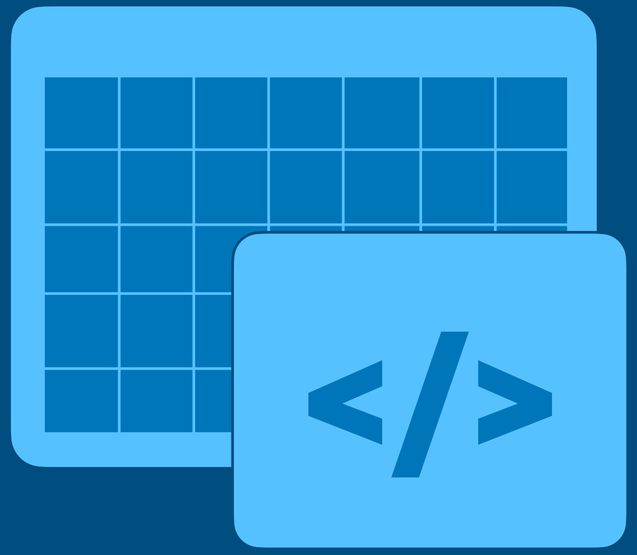


Analyzing Structured Data in Python

WEEK 1: PANDAS

14 March 2022

CDCS Python Course Series

Instructor: Lucy Havens

COURSE STRUCTURE

Anticipate about ~7 hours/week

- 2 course meetings per week
  - 10:00 - 11:00 AM BST Mondays
  - 10:00 - 11:00 AM BST Fridays
- 1 assignment per week ~2 hours
- Office hours on Wednesdays for 30 minutes per participant
- Independent learning ~2 hours

Teams for introductions, meetings, office hours, questions, files

COURSE TOPICS

Week 1: Pandas for CSV data

Week 2: ElementTree for XML data

Please let me know if there are specific topics you'd like to cover!

MORE PYTHON COURSES WITH THE CDCS

Machine Learning with Python
23 March - 13 April (2 days a week)

Text Analysis with Python's NLTK Library
11 April - 22 April (2 days a week)

The CDCS has many resources on it's website for courses and self-guided learning!

INTRODUCTIONS

Name

Have you programmed before?
If yes, in what language?

Why would you like to learn Python?

FOR PARTICIPANTS

- Introduce material for you to review in greater depth on your own
- I'll direct you to further resources if you'd like to go beyond material covered in each week's assignment
- Course meetings won't be recorded
  - Three strike policy
  - Please let me know in advance if you cannot attend!
- Office hours: questions about assignments, your own projects
  - *Chat with me on Teams to schedule!*

FOR PARTICIPANTS

Python version 3

Jupyter Notebooks - options:

A. Use Noteable: https://www.ed.ac.uk/information-services/learning-technology/noteable/accessing-noteable
B. Use GoogleColab: https://colab.research.google.com
C. Install to your computer: https://jupyter.org/install

LET'S CODE!

FURTHER RESOURCES

- Noteable User Guide: https://noteable.edina.ac.uk/user_guide/#hide_ge_7
- Jupyter Notebooks, Noteable: https://github.com/edina/Exemplars2020/blob/master/TeachingDocs/Tutorials/UsingNoteableBeginner.ipynb
- Jupyter Notebooks: https://glam-workbench.github.io/getting-started/
- Python: https://programminghistorian.org/en/lessons/introduction-and-installation

PANDAS

Pandas is built on NumPy

Pandas-specific data structures:

1. Series
2. DataFrame

Reference: https://pandas.pydata.org/pandas-docs/stable/user_guide/10min.html

PANDAS

DataFrames are like tables, storing data in rows and columns

- Text
- Numbers
- Lists, arrays, dictionaries

Each column can have a unique data type, or **dtype**

PANDAS

DataFrames can be created from data in numerous formats, including lists and arrays...

```
In [5]: dates = pd.date_range('20130101', periods=6)

In [6]: dates
Out[6]:
DatetimeIndex(['2013-01-01', '2013-01-02', '2013-01-03', '2013-01-04',
               '2013-01-05', '2013-01-06'],
              dtype='datetime64[ns]', freq='D')

In [7]: df = pd.DataFrame(np.random.randn(6, 4), index=dates, columns=list('ABCD'))

In [8]: df
Out[8]:
                   A         B         C         D
2013-01-01  0.469112 -0.282863 -1.509059 -1.135632
2013-01-02  1.212112 -0.173215  0.119209 -1.044236
2013-01-03 -0.861849 -2.104569 -0.494929  1.071804
2013-01-04  0.721555 -0.706771 -1.039575  0.271860
2013-01-05 -0.424972  0.567020  0.276232 -1.087401
2013-01-06 -0.673690  0.113648 -1.478427  0.524988
```

PANDAS

...dictionaries...

```
In [9]: df2 = pd.DataFrame({'A': 1.,
   ...:                     'B': pd.Timestamp('20130102'),
   ...:                     'C': pd.Series(1, index=list(range(4)), dtype='float32'),
   ...:                     'D': np.array([3] * 4, dtype='int32'),
   ...:                     'E': pd.Categorical(["test", "train", "test", "train"]),
   ...:                     'F': 'foo'})
   ...:

In [10]: df2
Out[10]:
     A           B    C  D      E    F
0  1.0 2013-01-02  1.0  3   test  foo
1  1.0 2013-01-02  1.0  3  train  foo
2  1.0 2013-01-02  1.0  3   test  foo
3  1.0 2013-01-02  1.0  3  train  foo
```

PANDAS

...and CSV files

```
In [142]: pd.read_csv('foo.csv')
Out[142]:
     Unnamed: 0          A          B          C          D
0    2000-01-01   0.350262   0.843315   1.798556   0.782234
1    2000-01-02  -0.586873   0.034907   1.923792  -0.562651
2    2000-01-03  -1.245477  -0.963406   2.269575  -1.612566
3    2000-01-04  -0.252830  -0.498066   3.176886  -1.275581
4    2000-01-05  -1.044057   0.118042   2.768571   0.386039
..          ...        ...        ...        ...        ...
995  2002-09-22 -48.017654  31.474551  69.146374 -47.541670
996  2002-09-23 -47.207912  32.627390  68.505254 -48.828331
997  2002-09-24 -48.907133  31.990402  67.310924 -49.391051
998  2002-09-25 -50.146062  33.716770  67.717434 -49.037577
999  2002-09-26 -49.724318  33.479952  68.108014 -48.822030

[1000 rows x 5 columns]
```

PANDAS

If you have a lot of data, you can select a subset of rows to view

```
In [13]: df.head()
Out[13]:
                   A         B         C         D
2013-01-01  0.469112 -0.282863 -1.509059 -1.135632
2013-01-02  1.212112 -0.173215  0.119209 -1.044236
2013-01-03 -0.861849 -2.104569 -0.494929  1.071804
2013-01-04  0.721555 -0.706771 -1.039575  0.271860
2013-01-05 -0.424972  0.567020  0.276232 -1.087401

In [14]: df.tail(3)
Out[14]:
                   A         B         C         D
2013-01-04  0.721555 -0.706771 -1.039575  0.271860
2013-01-05 -0.424972  0.567020  0.276232 -1.087401
2013-01-06 -0.673690  0.113648 -1.478427  0.524988
```

LET'S CODE!

PANDAS

There are lots of built in methods to reorganize your data...

```
In [20]: df.T
Out[20]:
   2013-01-01  2013-01-02  2013-01-03  2013-01-04  2013-01-05  2013-01-06
A    0.469112    1.212112   -0.861849    0.721555   -0.424972   -0.673690
B   -0.282863   -0.173215   -2.104569   -0.706771    0.567020
C   -1.509059    0.119209   -0.494929   -1.039575    0.276232
D   -1.135632   -1.044236    1.071804    0.271860   -1.087401
```

```
In [21]: df.sort_index(axis=1, ascending=False)
Out[21]:
                   D         C         B         A
2013-01-01 -1.135632 -1.509059 -0.282863  0.469112
2013-01-02 -1.044236  0.119209 -0.173215  1.212112
2013-01-03  1.071804 -0.494929 -2.104569 -0.861849
2013-01-04  0.271860 -1.039575 -0.706771  0.721555
2013-01-05 -1.087401  0.276232  0.567020 -0.424972
2013-01-06  0.524988 -1.478427  0.113648 -0.673690
```

```
In [22]: df.sort_values(by='B')
Out[22]:
                   A         B         C         D
2013-01-03 -0.861849 -2.104569 -0.494929  1.071804
2013-01-04  0.721555 -0.706771 -1.039575  0.271860
2013-01-01  0.469112 -0.282863 -1.509059 -1.135632
2013-01-02  1.212112 -0.173215  0.119209 -1.044236
2013-01-06 -0.673690  0.113648 -1.478427  0.524988
2013-01-05 -0.424972  0.567020  0.276232 -1.087401
```

PANDAS

...calculate statistics...

```
In [61]: df.mean()
Out[61]:
A   -0.004474
B   -0.383981
C   -0.687758
D    5.000000
F    3.000000
dtype: float64
```

```
In [62]: df.mean(1)
Out[62]:
2013-01-01    0.872735
2013-01-02    1.431621
2013-01-03    0.707731
2013-01-04    1.395042
2013-01-05    1.883656
2013-01-06    1.592306
Freq: D, dtype: float64
```

```
In [19]: df.describe()
Out[19]:
              A         B         C         D
count  6.000000  6.000000  6.000000  6.000000
mean   0.073711 -0.431125 -0.687758 -0.233103
std    0.843157  0.922818  0.779887  0.973118
min   -0.861849 -2.104569 -1.509059 -1.135632
25%   -0.611510 -0.600794 -1.368714 -1.076610
50%    0.022070 -0.228039 -0.767252 -0.386188
75%    0.658444  0.041933 -0.034326  0.461706
max    1.212112  0.567020  0.276232  1.071804
```

PANDAS

...and group and reshape your data!

```
In [87]: df = pd.DataFrame({'A': ['foo', 'bar', 'foo', 'bar',
   ....:                          'foo', 'bar', 'foo', 'foo'],
   ....:                    'B': ['one', 'one', 'two', 'three',
   ....:                          'two', 'two', 'one', 'three'],
   ....:                    'C': np.random.randn(8),
   ....:                    'D': np.random.randn(8)})
   ....:

In [88]: df
Out[88]:
     A      B         C         D
0  foo    one  1.346061 -1.577585
1  bar    one  1.511763  0.396823
2  foo    two  1.627081 -0.105381
3  bar  three -0.990582 -0.532532
4  foo    two -0.441652  1.453749
5  bar    two  1.211526  1.208843
6  foo    one  0.268520 -0.080952
7  foo  three  0.024580 -0.264610
```

```
In [89]: df.groupby('A').sum()
Out[89]:
            C         D
A
bar  1.732707  1.073134
foo  2.824590 -0.574779
```

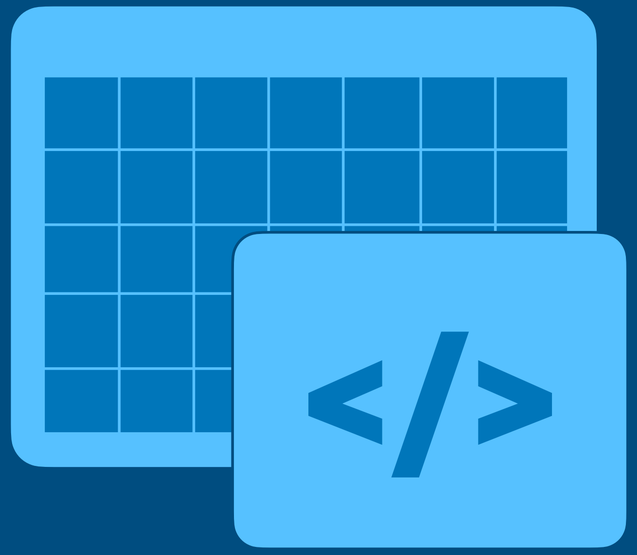
```
In [90]: df.groupby(['A', 'B']).sum()
Out[90]:
                  C         D
A   B
bar one    1.511763  0.396823
    three -0.990582 -0.532532
    two    1.211526  1.208843
foo one    1.614581 -1.658537
    three  0.024580 -0.264610
    two    1.185429  1.348368
```

LET'S CODE!

ASSIGNMENT

Watch the videos in sections *6. Introduction to Pandas* and *7. Baby Names with Pandas* in the course *Python: Data Analysis (2015)*
https://www.linkedin.com/learning/python-data-analysis-2015/dataframes-in-pandas?u=50251009
Follow along in your own Jupyter Notebook!

Find your own CSV file to load and turn into a DataFrame (or create your own). What questions can you ask about it using the methods and functions in Pandas?



THANKS EVERYONE!

Next course meeting: Friday, 10:00-11:00 AM BST

Office hours available on Wednesday (30 minutes)

To schedule, please message me on Teams!