In [1]:
```
from IPython.display import IFrame
import ipynb_style
from epstata import Stpy
import pandas as pd
from itertools import combinations
from importlib import reload
```

In [2]:
```
reload(ipynb_style)
ipynb_style.clean()
#ipynb_style.presentation()
#ipynb_style.pres2()
```

Out[2]:

Data Workflows in Stata and Python

(http://www.stata.com) (https://www.python.c

Data Workflows in Stata and Python

Dejan Pavlic, Education Policy Research Initiative, University of Ottawa

Stephen Childs (presenter), Office of Institutional Analysis, University of Calgary

UNIVERSITY OF CALGARY

uOttawa

(http://ucalgary.ca) (http://uottaw
(http://socialsciences.uottawa.ca/
epri/eng/index.asp)

EPRI Education Policy Research Initiative

Initiative de recherche sur les politiques de l'éducation

Introduction

About this talk

Objectives

- know what Python is and what advantages it has
- know how Python can work with Stata

Please save questions for the end. Or feel free to ask me today or after the conference.

Outline

- Introduction
  - Overall
  - Motivation
  - About Python

- Building Blocks
  - Running Stata from Python
  - Pandas
  - Python language features

- Workflows
  - ETL/Data Cleaning
  - Stata code generation
  - Processing Stata output

About Me

- Started using Stata in grad school (2006).
- Using Python for about 3 years.
- Post-Secondary Education sector

- University of Calgary - Institutional Analysis (https://oia.ucalgary.ca/Contact)
- Education Policy Research Initiative (http://socialsciences.uottawa.ca/irpe-epri/eng/index.asp - University of Ottawa (a Stata shop)

Motivation

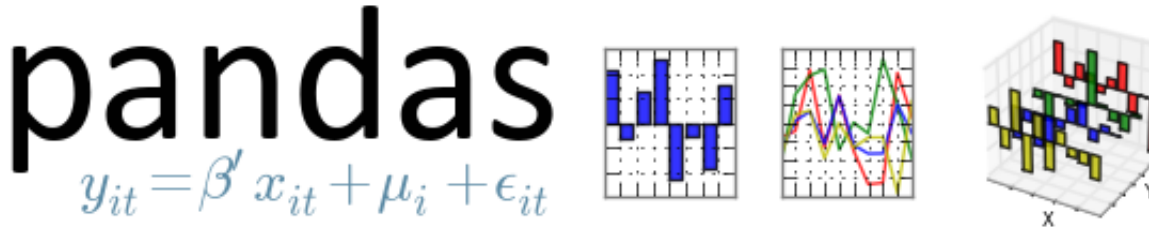
- Python is becoming very popular in the data world.
- Python skills are widely applicable.
- Python is powerful and flexible and will help you get more done, faster.

About Python

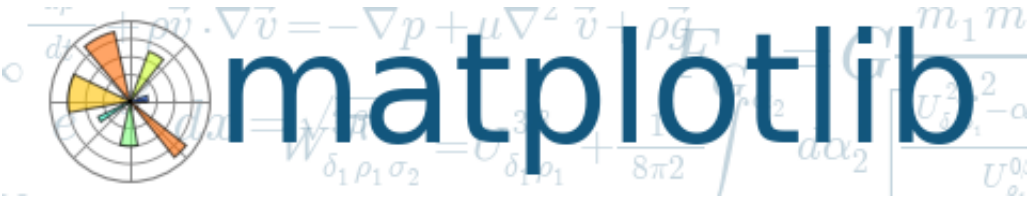
The Python Language

- General purpose programming language
- Name comes from Monty Python
- Python 2 vs. 3 - use Python 3
- "batteries included"

Scientific Python



(http://pandas.pydata.org)



(http://matplotlib.org/)

(http://www.numpy.org)

(http://scipy.org)SciPy

(https://jupyter.org/)

(http://continuum.io/downloads)

Building Blocks

Stata Commands from Python

- Use the Stata command line
- Python's `subprocess` module runs each instance of Stata
- Each instance is a Python object
- Can send it commands with the `write()` method

In [3]:
```
stata = Stpy()
```

```
  ___  ____  ____  ____  ____ (R)
 /__    /   ____/   /   ____/
___/   /   /___/   /   /___/   13.1   Copyright 1985-2013 StataCo
rp LP
  Statistics/Data Analysis            StataCorp
                                      4905 Lakeway Drive
     MP - Parallel Edition            College Station, Texas 7784
5 USA
                                      800-STATA-PC        htt
p://www.stata.com (http://www.stata.com)
                                      979-696-4600        stata@s
tata.com
                                      979-696-4601 (fax)

2-user 2-core Stata network perpetual license:
       Serial number:  501306211345
         Licensed to:  Stephen Childs
                       Education Policy Research Initiative

Notes:
      1.  (-v# option or -set maxvar-) 5000 maximum variables
      2.  Command line editing disabled
      3.  Stata running in batch mode

.
```

In [4]:
```
stata.write('sysuse auto')
```

```
sysuse auto
(1978 Automobile Data)

.
```

In [5]:
```
stata.write('describe')
```

```
describe

Contains data from /Applications/Stata/ado/base/a/auto.dta
  obs:            74                          1978 Automobile Dat
a
 vars:            12                          13 Apr 2013 17:45
 size:         3,182                          (_dta has notes)
------------------------------------------------------------------
--------------
              storage   display    value
variable name   type    format     label      variable label
------------------------------------------------------------------
--------------
make            str18   %-18s                 Make and Model
price           int     %8.0gc                Price
mpg             int     %8.0g                 Mileage (mpg)
rep78           int     %8.0g                 Repair Record 1978
headroom        float   %6.1f                 Headroom (in.)
trunk           int     %8.0g                 Trunk space (cu. f
t.)
weight          int     %8.0gc                Weight (lbs.)
length          int     %8.0g                 Length (in.)
turn            int     %8.0g                 Turn Circle (ft.)
displacement    int     %8.0g                 Displacement (cu. i
n.)
gear_ratio      float   %6.2f                 Gear Ratio
foreign         byte    %8.0g      origin     Car type
------------------------------------------------------------------
--------------
Sorted by:  foreign

.
```

Python strings have a `format()` method that allows you to substitute the contents of Python variables

In [6]:
```
depvar = 'mpg'
indepvars = ['weight', 'wtsq', 'foreign']
```

In [7]:
```
'regress {depvar} {indepvars}'.format(depvar=depvar,
                                      indepvars=' '.join(indepvars))
```

Out[7]:
```
'regress mpg weight wtsq foreign'
```

Pandas

- General introduction
  - Origins - NumPy
  - Current popularity

- Key concepts
  - DataFrame
  - Series
  - index

Example

I will use the `sysuse auto` dataset to demonstrate some basic functions with Pandas. This is taken fr
Stata tutorial and reflects basic commands for exploring and manipulating your data.

Import Pandas and Load Data

In [8]:
```
import pandas as pd
```

In [2]:
```
auto = pd.read_stata('auto.dta')
```

Overall Dataset Description

In [10]:
```
auto.shape
```

Out[10]:
```
(74, 12)
```

In [11]:
```
auto.dtypes
```

Out[11]:
```
make              object
price              int16
mpg                int16
rep78            float64
headroom         float32
trunk              int16
weight             int16
length             int16
turn               int16
displacement       int16
gear_ratio       float32
foreign         category
dtype: object
```

Overall Summary Statistics

In [12]:
```
stata.write('summarize')
```

```
summarize

    Variable |       Obs        Mean    Std. Dev.       Min
Max
-------------+--------------------------------------------------------
        make |         0
       price |        74    6165.257    2949.496       3291
15906
         mpg |        74     21.2973    5.785503         12
41
       rep78 |        69    3.405797    .9899323          1
5
    headroom |        74    2.993243    .8459948        1.5
5
-------------+--------------------------------------------------------
       trunk |        74    13.75676    4.277404          5
23
      weight |        74    3019.459    777.1936       1760
4840
      length |        74    187.9324    22.26634        142
233
        turn |        74    39.64865    4.399354         31
51
displacement |        74    197.2973    91.83722         79
425
-------------+--------------------------------------------------------
  gear_ratio |        74    3.014865    .4562871       2.19
3.89
     foreign |        74    .2972973    .4601885          0
1

.
```

In [13]:
```
auto.describe().T
```

Out[13]:

	count	mean	std	min	25%	50%	75%
price	74	6165.256757	2949.495885	3291.00	4220.25	5006.500	6332.2
mpg	74	21.297297	5.785503	12.00	18.00	20.000	24.750
rep78	69	3.405797	0.989932	1.00	3.00	3.000	4.000(
headroom	74	2.993243	0.845995	1.50	2.50	3.000	3.500(
trunk	74	13.756757	4.277404	5.00	10.25	14.000	16.750
weight	74	3019.459459	777.193567	1760.00	2250.00	3190.000	3600.(
length	74	187.932432	22.266340	142.00	170.00	192.500	203.75
turn	74	39.648649	4.399354	31.00	36.00	40.000	43.00(
displacement	74	197.297297	91.837219	79.00	119.00	196.000	245.25
gear_ratio	74	3.014865	0.456287	2.19	2.73	2.955	3.3525

View the first observations

In [14]:
```
stata.write('list make price mpg rep78 foreign in 1/5')
```

```
list make price mpg rep78 foreign in 1/5

     +-------------------------------------------------+
     | make            price   mpg   rep78    foreign |
     |-------------------------------------------------|
  1. | AMC Concord     4,099    22       3   Domestic |
  2. | AMC Pacer       4,749    17       3   Domestic |
  3. | AMC Spirit      3,799    22       .   Domestic |
  4. | Buick Century   4,816    20       3   Domestic |
  5. | Buick Electra   7,827    15       4   Domestic |
     +-------------------------------------------------+

.
```

In [15]:
```
auto[['make', 'price', 'mpg', 'rep78', 'foreign']].head(5)
```

Out[15]:

	make	price	mpg	rep78	foreign
0	AMC Concord	4099	22	3	Domestic
1	AMC Pacer	4749	17	3	Domestic
2	AMC Spirit	3799	22	NaN	Domestic
3	Buick Century	4816	20	3	Domestic
4	Buick Electra	7827	15	4	Domestic

What about listing the first 5 foreign cars...

In [16]:
```
auto[auto.foreign=='Foreign'][['make',
                               'price',
                               'mpg',
                               'rep78',
                               'foreign']].head(5)
```

Out[16]:

	make	price	mpg	rep78	foreign
52	Audi 5000	9690	17	5	Foreign
53	Audi Fox	6295	23	3	Foreign
54	BMW 320i	9735	25	4	Foreign
55	Datsun 200	6229	23	4	Foreign
56	Datsun 210	4589	35	5	Foreign

List Observations missing 1978 Repair Record

In [17]:
```
stata.write('list make price mpg rep78 foreign if missing(rep78)')
```

```
list make price mpg rep78 foreign if missing(rep78)

     +-------------------------------------------------+
     | make             price   mpg   rep78    foreign |
     |-------------------------------------------------|
  3. | AMC Spirit       3,799    22       .   Domestic |
  7. | Buick Opel       4,453    26       .   Domestic |
 45. | Plym. Sapporo    6,486    26       .   Domestic |
 51. | Pont. Phoenix    4,424    19       .   Domestic |
 64. | Peugeot 604     12,990    14       .    Foreign |
     +-------------------------------------------------+

.
```

In [18]:
```
auto[auto.rep78.isnull()][['make', 'price', 'mpg', 'rep78', 'foreign']]
```

Out[18]:

	make	price	mpg	rep78	foreign
2	AMC Spirit	3799	22	NaN	Domestic
6	Buick Opel	4453	26	NaN	Domestic
44	Plym. Sapporo	6486	26	NaN	Domestic
50	Pont. Phoenix	4424	19	NaN	Domestic
63	Peugeot 604	12990	14	NaN	Foreign

One-way Tabs

In [19]:
```
stata.write('tabulate foreign')
```

```
tabulate foreign

   Car type |      Freq.     Percent        Cum.
------------+-----------------------------------
   Domestic |         52       70.27       70.27
    Foreign |         22       29.73      100.00
------------+-----------------------------------
      Total |         74      100.00

.
```

In [20]:
```
auto.foreign.value_counts()
```

Out[20]:
```
Domestic    52
Foreign     22
dtype: int64
```

Crosstabs

In [21]:
```
stata.write('tabulate rep78 foreign')
```

```
tabulate rep78 foreign

    Repair |
    Record |       Car type
      1978 |  Domestic    Foreign |     Total
-----------+----------------------+----------
         1 |         2           0 |         2
         2 |         8           0 |         8
         3 |        27           3 |        30
         4 |         9           9 |        18
         5 |         2           9 |        11
-----------+----------------------+----------
     Total |        48          21 |        69


.
```

In [22]:
```
pd.crosstab(auto.rep78, auto.foreign)
```

Out[22]:

foreign	Domestic	Foreign
rep78		
1	2	0
2	8	0
3	27	3
4	9	9
5	2	9

By and Groupby

In [23]:
```
stata.write('by foreign: summarize')
```

```
by foreign: summarize

-------------------------------------------------------------------------------
-> foreign = Domestic

    Variable |       Obs        Mean    Std. Dev.       Min
Max
-------------+--------------------------------------------------------
-----
        make |         0
       price |        52    6072.423    3097.104       3291
15906
```

```
         mpg |        52    19.82692    4.743297         12
34
       rep78 |        48    3.020833     .837666          1
5
    headroom |        52    3.153846    .9157578        1.5
5
-------------+--------------------------------------------------------
       trunk |        52       14.75    4.306288          7
23
      weight |        52    3317.115    695.3637       1800
4840
      length |        52    196.1346    20.04605        147
233
        turn |        52    41.44231    3.967582         31
51
displacement |        52    233.7115    85.26299         86
425
-------------+--------------------------------------------------------
  gear_ratio |        52    2.806538    .3359556       2.19
3.58
     foreign |        52           0           0          0
0

-------------------------------------------------------------------------------
-> foreign = Foreign

    Variable |       Obs        Mean    Std. Dev.       Min
Max
-------------+--------------------------------------------------------
        make |         0
       price |        22    6384.682    2621.915       3748
12990
         mpg |        22    24.77273    6.611187         14
41
       rep78 |        21    4.285714    .7171372          3
5
    headroom |        22    2.613636    .4862837        1.5
3.5
-------------+--------------------------------------------------------
       trunk |        22    11.40909    3.216906          5
16
      weight |        22    2315.909    433.0035       1760
3420
      length |        22    168.5455    13.68255        142
193
        turn |        22    35.40909    1.501082         32
38
displacement |        22    111.2273    24.88054         79
163
```

```
-------------+--------------------------------------------------------
  gear_ratio |         22    3.507273    .2969076        2.98
3.89
     foreign |         22           1           0           1
1


.
```

In [24]:
```
auto.groupby(by=auto.foreign).describe(percentiles=[]).T
```

Out[24]:

foreign	Domestic						Foreig
	count	mean	std	min	50%	max	count
displacement	52	233.711538	85.262993	86.00	231.00	425.00	22
gear_ratio	52	2.806538	0.335960	2.19	2.75	3.58	22
headroom	52	3.153846	0.915758	1.50	3.50	5.00	22
length	52	196.134615	20.046054	147.00	200.00	233.00	22
mpg	52	19.826923	4.743297	12.00	19.00	34.00	22
price	52	6072.423077	3097.104279	3291.00	4782.50	15906.00	22
rep78	48	3.020833	0.837666	1.00	3.00	5.00	21
trunk	52	14.750000	4.306288	7.00	16.00	23.00	22
turn	52	41.442308	3.967582	31.00	42.00	51.00	22
weight	52	3317.115385	695.363740	1800.00	3360.00	4840.00	22

Mean By Category

In [25]:
```
stata.write('tabulate foreign, summarize(mpg)')
```

```
tabulate foreign, summarize(mpg)

           |      Summary of Mileage (mpg)
   Car type |        Mean   Std. Dev.       Freq.
------------+------------------------------------
   Domestic |   19.826923   4.7432972          52
    Foreign |   24.772727   6.6111869          22
------------+------------------------------------
      Total |   21.297297   5.7855032          74


.
```

In [26]:
```
auto.groupby(by=auto.foreign)['mpg'].mean()
```

Out[26]:
```
foreign
Domestic    19.826923
Foreign     24.772727
Name: mpg, dtype: float64
```

Correlation

In [27]:
```
stata.write('correlate mpg weight')
```

```
correlate mpg weight
(obs=74)

             |      mpg   weight
-------------+------------------
         mpg |   1.0000
      weight |  -0.8072   1.0000


.
```

In [28]:
```
auto.mpg.corr(auto.weight)
```

Out[28]:
```
-0.80717485894244212
```

In [29]:
```
auto[['mpg','weight']].corr()
```

Out[29]:

	mpg	weight
mpg	1.000000	-0.807175
weight	-0.807175	1.000000

Python Language Features

Data Types: Lists and Dictionaries

In [30]:
```
software = ['Stata', 'Python'] # list
person = {'name': 'Stephen Childs',
          'employer': 'University of Calgary',
          'city': 'Calgary',
          'province': 'Alberta'} #dictionary
software[0], person['name']
```

Out[30]:
```
('Stata', 'Stephen Childs')
```

Python has some very useful and powerful language features. A good starting point is to look at some o

structures built into the language. You can think of a Python list as a Stata macro list. In Python, lists ca
any type of object and can even contain different types in the same list. Lists are *ordered*.

Dictionaries are a very powerful data type. It lets you define a set of **keys** and related **values**. This is a v
powerful and flexible data structure. The values are *unordered*.

Functions

In [31]:
```
def say_hello():
    print('Hello world!')

say_hello()
```

```
Hello world!
```

In [32]:
```
def double(x):
    return x*2

double(2)
```

Out[32]:
```
4
```

`lambda` Functions

In [33]:
```
(lambda x: x**2)(8)
```

Out[33]:
```
64
```

In [34]:
```
a = pd.Series(range(8))
a
```

Out[34]:
```
0    0
1    1
2    2
3    3
4    4
5    5
6    6
7    7
dtype: int64
```

```
In [35]: a.apply(lambda x: x**2)

Out[35]: 0     0
         1     1
         2     4
         3     9
         4    16
         5    25
         6    36
         7    49
         dtype: int64
```

Glossary

Stata	**Python**
macro	variable
macro list	list
variable	`pd.Series` or a column of a `pd.DataFrame`
data	`pd.DataFrame`
`.dta` file	• Pickle (https://docs.python.org/3/library/pickle (though be careful! (https://www.youtube.com/watch?v=7KnfGDajDQw)) • `csv` • HDF5 (https://www.hdfgroup.org/HDF5/) (Pyt package (www.h5py.org)) • etc…
`ssc`	`pip`
Boston College Statistical Software Components (SSC) archive (https://ideas.repec.org/s/boc/bocode.html)	PyPI - the Python Package Index (https://pypi.pytho ("The Cheeseshop" (https://www.youtube.com/watc v=PPN3KTtrnZM))
`program`	function
`do` file	module (`.py` file)
`ado` file	package

Workflows

ETL/Data Cleaning

- **advantages:**
  - wide variety of tools already available
  - create new data cleaning functions
- use pandas to prepare the data
  - organize code in separate files
  - move data into Stata when analysis file is ready

In [ ]:
```
"""Recode the PSIS Gender variable.

Input
-----
sd.raw.Gender.pd

Output
------
sd.Gender.pd

"""

import epandas as pd
import numpy as np

from eppdoc import eppdoc

# Load series.
d = eppdoc(__doc__)
s = d.read()

# Change to full labels.
s = s.replace({'F': 'Female', 'M': 'Male', np.nan: 'N/A'})

# Categorize and Order
s = s.astype('category')
s = s.cat.reorder_categories(['Female', 'Male', 'N/A'])

# Pickle.
d.writes(s)
```

e.g. You have a data source - a set of files - it is easy to write a Python sccript to turn that set of files int analysis files.

Stata Code Generation

- a less "fiddly" replacement for macros (see the `format` method above)
- deal with repitition and patterns

In [36]:
```
vars = ['mpg', 'rep78', 'headroom', 'trunk', 'weight',
        'length', 'turn', 'displacement',
        'gear_ratio', 'i.foreign']
```

In [37]:
```
from itertools import combinations
```

In [38]:
```
for x in combinations(vars, 2):
    print('regress price {vars}'.format(vars=' '.join(x)))
```

```
regress price mpg rep78
regress price mpg headroom
regress price mpg trunk
regress price mpg weight
regress price mpg length
regress price mpg turn
regress price mpg displacement
regress price mpg gear_ratio
regress price mpg i.foreign
regress price rep78 headroom
regress price rep78 trunk
regress price rep78 weight
regress price rep78 length
regress price rep78 turn
regress price rep78 displacement
regress price rep78 gear_ratio
regress price rep78 i.foreign
regress price headroom trunk
regress price headroom weight
regress price headroom length
regress price headroom turn
regress price headroom displacement
regress price headroom gear_ratio
regress price headroom i.foreign
regress price trunk weight
regress price trunk length
regress price trunk turn
regress price trunk displacement
regress price trunk gear_ratio
regress price trunk i.foreign
regress price weight length
regress price weight turn
regress price weight displacement
regress price weight gear_ratio
regress price weight i.foreign
regress price length turn
regress price length displacement
regress price length gear_ratio
regress price length i.foreign
regress price turn displacement
regress price turn gear_ratio
regress price turn i.foreign
regress price displacement gear_ratio
regress price displacement i.foreign
regress price gear_ratio i.foreign
```

Processing Stata Output

In [ ]:
```
import pandas as pd
import numpy as np
from epstata import Stpy
import predict_models

def predict_test(a, filename, model, key, iteration, results):
    a.write('use "{filename}", clear'.format(filename=filename))
    a.write('generate phat = .')
    tmp1 = NamedTemporaryFile(suffix='.dta')
    a.write('save {name}, replace'.format(name=tmp1.name))
    a.write('capture drop phat')
    a.write('set seed {iteration}'.format(iteration=iteration))
    a.write('generate cut = runiform()')
    a.write('logit Leaver {model} if cut < .5, iterate(200)'.format(
            model=model))
    a.write('predict phat if cut >= .5')
    a.write('keep if cut >= .5')
    a.write('keep ID Leaver phat')
    a.write('save {name}, replace'.format(name=tmp1.name))
    df = pd.read_stata(tmp1.name)
    df.set_index(df['ID'], inplace=True)
                df['Log Loss'] = -1 * (df['Leaver']*np.log(df['phat'])
            + (1-df['Leaver'])*np.log(1-df['phat']))
    results.write('{key} {iteration}: Log Loss: {logl}\n'.format(
            key=key,
            iteration=iteration,
            logl=df['Log Loss'].mean()))
    return df['Log Loss']
```

Conclusion

- only an introduction to Python meant to whet your appitite
- show some possibilites
- Stata/Python integration is still a work in progress
- allow you to mix and match - replace part of your workflow with Python

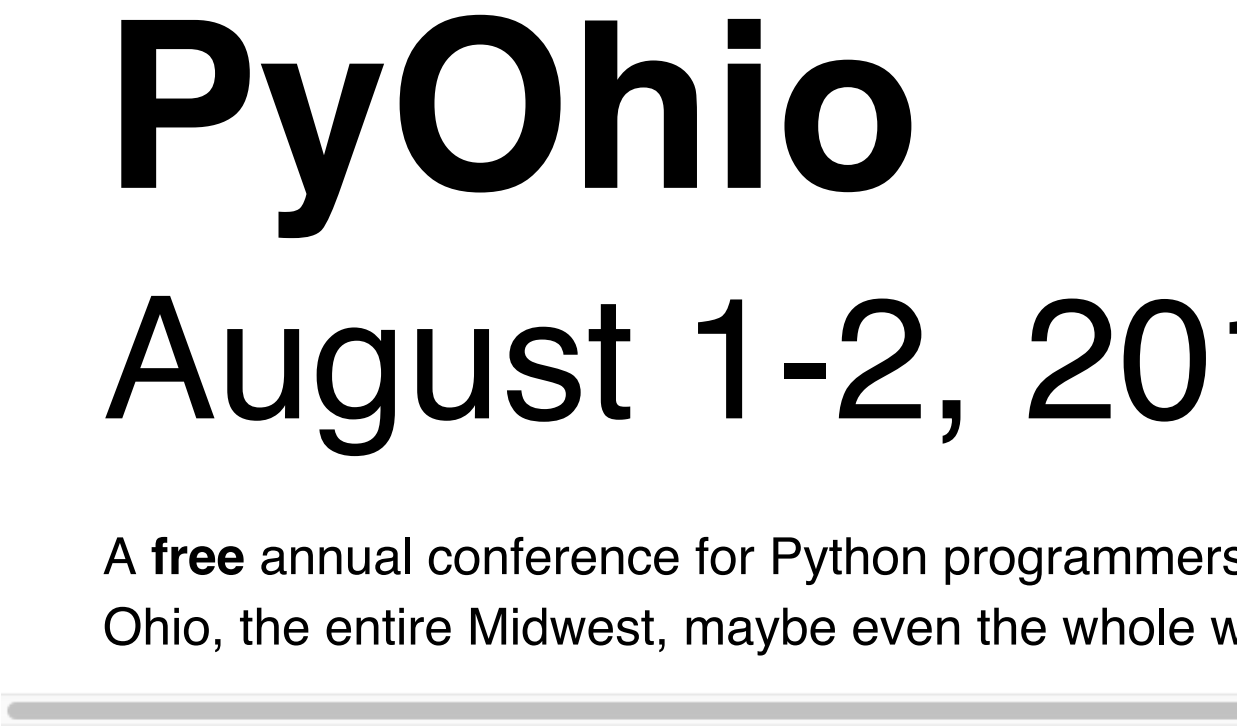
In [39]:
```
import this
```

```
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do
it.
Although that way may not be obvious at first unless you're Dutc
h.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

In [3]:
```
IFrame(src='http://www.pyohio.org/', width=1250, height=450)
```

Out[3]:



Resources

- Brandon Rhodes pandas tutorial (https://github.com/brandon-rhodes/pycon-pandas-tutorial) - PyC
  Montréal 2015

Contact

- The notebook will be up on github: https://github.com/sechilds/stataconf2015 (https://github.com/sechilds/stataconf2015)
  - You can see it rendered on nbviewer (http://nbviewer.ipython.org/github/sechilds/stataconf2015/blob/master/StataConf2015.ipynb
- Twitter: @sechilds (https://twitter.com/sechilds)
- E-mail: Stephen.Childs@ucalgary.ca (mailto:Stephen.Childs@uCalgary.ca)