Agenda

- Numpy
- Pandas
- Lab

Introduction

Create a new notebook for your code-along:

If necessary, from our lab03 submission directory type

```
ipython notebook
```

from the IPython Dashboard open a new notebook. Change the title to "Numpy and Pandas"

Introduction to Numpy

- Overview
- ndarray
- Indexing and Slicing

More info: http://wiki.scipy.org/Tentative_NumPy_Tutorial (http://wiki.scipy.org/Tentative_NumPy_Tutorial)

Numpy Overview

- Why Python for Data? Numpy brings decades of C math into Python
- Numpy provides wrapper for extensive C/C++/Fortran codebases for data analysis and analytical functionality
- NDAarray allows easy vectorized math and broadcasting (functions on vector elements of different shapes)
- The computational foundation fro

In []:
```
from numpy import * #Load all the numpy packages
```

Note: "import as"

import * loads all sub module and is wasteful of memory when incorporated into deployed code. We use it here by example -- and its fine to use for learning purposes, legibility, etc.

As we'll see later, the the convention is to use:

```
import numpy as np
```

And then to specifically call needed methods:

```
In [2]: import numpy as np

In [5]: zeros?
        Object `zeros` not found.

In [6]: np.zeros?
```

```
Type:       builtin_function_or_method
String Form:<built-in function zeros>
Docstring:
zeros(shape, dtype=float, order='C')

Return a new array of given shape and type, filled with zeros.

Parameters
----------
shape : int or sequence of ints
```

Creating ndarrays

An array object represents a multidimensional, homogeneous array of fixed-size items.

```
In [ ]: # Creating arrays
        a = zeros((3))
        b = ones((2,3))
        c = random.randint(1,10,(2,3,4))
        d = arange(0,11,1)
```

What are these functions?

```
arange?
```

```
In [ ]: # Note the way each array is printed:
        a,b,c,d
```

```
In [ ]: ## Arithmetic in arrays is element wise
```

```
In [ ]: >>> a = array( [20,30,40,50] )
        >>> b = arange( 4 )
        >>> b
```

```
In [ ]: >>> c = a-b
        >>> c
```

In []:
```
>>> b**2
```

Indexing, Slicing and Iterating

In []:
```
# one-dimensional arrays work like lists:
a = arange(10)**2
```

In []:
```
a
```

In []:
```
a[2:5]
```

In []:
```
# Multidimensional arrays use tuples with commas for indexing
# with (row,column) conventions beginning, as always in Python, from 0
```

In []:
```
b = random.randint(1,100,(4,4))
```

In []:
```
b
```

In []:
```
# Guess the output
print(b[2,3])
print(b[0,0])
```

In []:
```
b[0:3,1],b[:,1]
```

In []:
```
b[1:3,:]
```

Pandas

- Object Creation
- Viewing data
- Selection
- Missing data
- Grouping
- Reshaping
- Time series
- Plotting
- i/o

pandas.pydata.org

Pandas Overview

Source: pandas.pydata.org (http://pandas.pydata.org/pandas-docs/stable/10min.html)

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [ ]:
```
dates = pd.date_range('20140101',periods=6)
```

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

In [ ]:
```
# Index, columns, underlying numpy data
df
```

In [ ]:
```
df2 = pd.DataFrame({ 'A' : 1.,
                     'B' : pd.Timestamp('20130102'),
                     'C' : pd.Series(1,index=list(range(4)),dtype='float3
2'),
                     'D' : np.array([3] * 4,dtype='int32'),
                     'E' : 'foo' })



df2
```

In [ ]:
```
# With specific dtypes
df2.dtypes
```

Viewing Data

In [ ]:
```
df.head()
```

In [ ]:
```
df.tail()
```

In [ ]:
```
df.index
```

In [ ]:
```
df.describe()
```

In [ ]:
```
df.sort(columns='B')
```

Selection

In [ ]:
```
df['A']
```

In [ ]:
```
df[0:3]
```

In [ ]:
```
# By label
df.loc[dates[0]]
```

In [ ]:
```
# multi-axis by label
df.loc[:,['A','B']]
```

In [ ]:
```
# Date Range
df.loc['20140102':'20140104',['B']]
```

In [ ]:
```
# Fast access to scalar
df.at[dates[1],'B']
```

In [ ]:
```
# iloc provides integer locations similar to np style
```

Boolean Indexing

In [ ]:
```
df[df.A < 0] # Basically a 'where' operation
```

Setting

In [ ]:
```
df_posA = df.copy() # Without "copy" it would act on the dataset
df_posA[df_posA.A < 0] = -1*df_posA
```

In [ ]:
```
df_posA
```

In [ ]:
```
#Setting new column aligns data by index
s1 = pd.Series([1,2,3,4,5,6],index=pd.date_range('20140102',periods=6))
```

In [ ]:
```
s1
```

In [ ]:
```
df['F'] = s1
```

In [ ]:
```
df
```

Missing Data

In [ ]:
```
# Add a column with missing data
df1 = df.reindex(index=dates[0:4],columns=list(df.columns) + ['E'])
```

In [ ]:
```
df1.loc[dates[0]:dates[1],'E'] = 1
```

In [ ]:
```
df1
```

In [ ]:
```
# find where values are null
pd.isnull(df1)
```

Operations

In [ ]:
```
df.describe()
```

In [ ]:
```
df.mean(),df.mean(1) # Operation on two different axes
```

Applying functions

In [ ]:
```
df
```

In [ ]:
```
df.apply(np.cumsum)
```

In [ ]:
```
df.apply(lambda x: x.max() - x.min())
```

In [ ]:
```
# Built in string methods
s = pd.Series(['A', 'B', 'C', 'Aaba', 'Baca', np.nan, 'CABA', 'dog', 'cat'])
s.str.lower()
```

Merge

In [ ]:
```
random.randn(10,4)
```

In [ ]:
```
#Concatenating pandas objects together
```

```
df = pd.DataFrame(np.random.randn(10,4))
df
```

In [ ]:
```
# Break it into pieces
pieces = [df[:3], df[3:7],df[7:]]
```

In [ ]:
```
pd.concat(pieces)
```

In [ ]:
```
# Also can "Join" and "Append"
```

Grouping

In [ ]:
```
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 [ ]:
```
df
```

In [ ]:
```
df.groupby(['A','B']).sum()
```

Reshaping

In [ ]:
```
# You can also stack or unstack levels
```

In [ ]:
```
a = df.groupby(['A','B']).sum()
```

In [ ]:
```
# Pivot Tables
pd.pivot_table(df,values=['C','D'],rows=['A'],cols=['B'])
```

Time Series

In [ ]:
```
import pandas as pd
import numpy as np
```

In [ ]:
```
# 100 Seconds starting on January 1st
```

```
rng = pd.date_range('1/1/2014', periods=100, freq='S')
```

In []:
```
# Give each second a random value
ts = pd.Series(np.random.randint(0, 500, len(rng)), index=rng)
```

In []:
```
ts
```

In []:
```
# Built in resampling
ts.resample('1Min',how='mean') # Resample secondly to 1Minutely
```

In []:
```
# Many additional time series features
ts. #use tab
```

Plotting

In []:
```
ts.plot()
```

In []:
```
def randwalk(startdate,points):
    ts = pd.Series(np.random.randn(points), index=pd.date_range(startdate, pe
riods=points))
    ts=ts.cumsum()
    ts.plot()
    return(ts)
```

In []:
```
# Using pandas to make a simple random walker by repeatedly running:
a=randwalk('1/1/2012',1000)
```

In []:
```
# Pandas plot function will print with labels as default
```

In []:
```
df = pd.DataFrame(np.random.randn(100, 4), index=ts.index,columns=['A', 'B',
'C', 'D'])
df = df.cumsum()
plt.figure();df.plot();plt.legend(loc='best') #
```

I/O

I/O is straightforward with, for example, pd.read_csv or df.to_csv

The benefits of open source:

Let's look under x's in plt modules

Lab

Next Steps

Recommended Resources

Name	Description
Official Pandas Tutorials (http://pandas.pydata.org/pandas-docs/stable/tutorials.html)	Wes & Company's selection of tutorials and lectures
Julia Evans Pandas Cookbook (https://github.com/jvns/pandas-cookbook)	Great resource with examples from weather, bikes and 311 calls
Learn Pandas Tutorials (https://bitbucket.org/hrojas/learn-pandas)	A great series of Pandas tutorials from Dave Rojas
Research Computing Python Data PYNBs (https://github.com/ResearchComputing/Meetup-Fall-2013/tree/master/python)	A super awesome set of python notebooks from a meetup-based course exclusively devoted to pandas