Python Walkthrough

Python Setup (I)

- Necessary for the programming portions of the assignments
- More precisely, use Ipython (ipython.org)

IP[y]: IPython
Interactive Computing

Install · Documentation · Project · Jupyter · News · Cite · Donate · Books

IPython provides a rich architecture for interactive computing with:

- A powerful interactive shell.
- A kernel for Jupyter.
- Support for interactive data visualization and use of GUI toolkits.
- Flexible, embeddable interpreters to load into your own projects.
- Easy to use, high performance tools for parallel computing.

Python Setup (II)

- Install Python
  - Use Anaconda (https://www.continuum.io/downloads)
  - Python 2 vs Python 3 (your choice)

Python Setup (III)

- Install Ipython/Jupyter
  - If you installed the Anaconda distribution, you are all set
  - Else use the command on the command-line

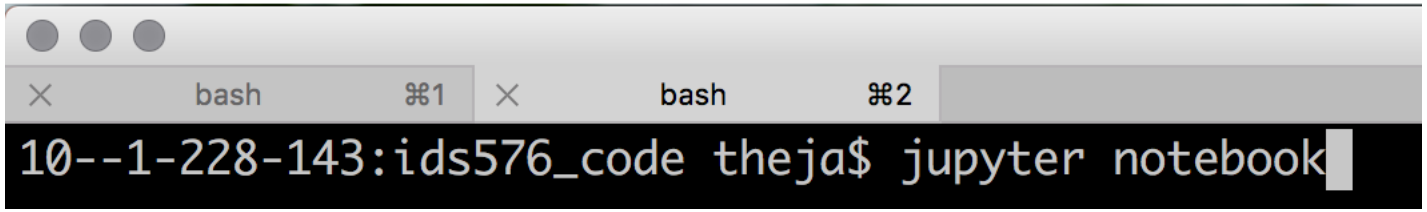
```
bash ⌘1    bash ⌘2
10--1-228-143:~ theja$ pip install jupyter
```

or

```
10--1-228-143:~ theja$ pip install ipython
```

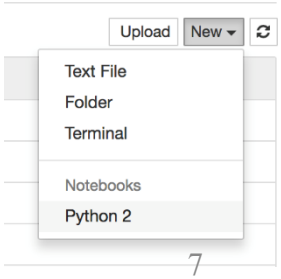
Python Setup (IV)

- Run Jupyter (or ipython)



- Your browser with open a page like this

- Start a new notebook (see button on the right)



Python Setup (V)

jupyter Untitled

File Edit View Insert Cell Kernel Help Python 2

Code CellToolbar

```
In [ ]: x = 1
        y = 2
        print(x+y)
```

(code)

cells

jupyter Untitled

File Edit View Insert Cell Kernel Help Python 2

Code CellToolbar

```
In [1]: x = 1
        y = 2
        print(x+y)

        3

In [ ]:
```

Press shift+enter, or ctrl+enter

Python Setup (VI)

- Global variables are shared between cells
- Cells are typically run from top to bottom

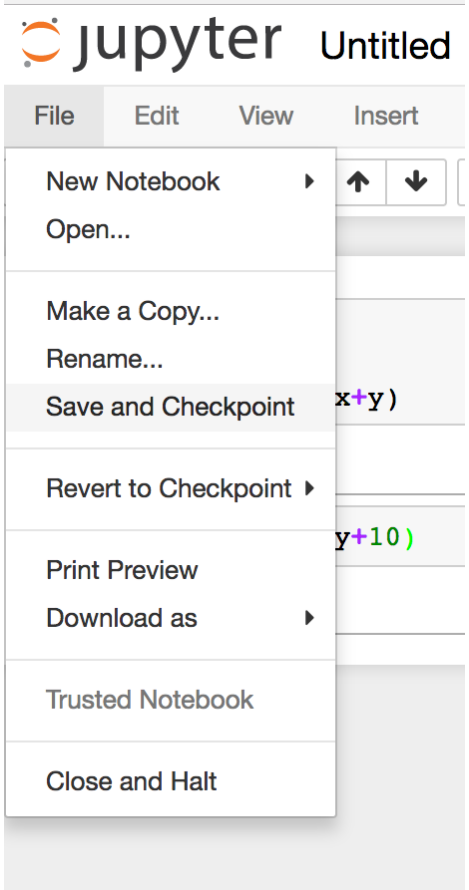
```
In [1]: x = 1
        y = 2
        print(x+y)

        3

In [2]: print(y+10)

        12
```

- Save changes using the save button



Python Review

- General purpose programming language
- 2 vs 3 (3 is backward incompatible)
- Very similar to Matlab (and better) for scientific computing
- It is dynamically typed

Python Review: Data Types

```
In [1]: x = 3
        y = 3.0
        z = 2
        print(x)
        print(y)
        print type(x)
        print type(y)
        print(x/z)
        print(y/z)

        3
        3.0
        <type 'int'>
        <type 'float'>
        1
        1.5
```

Python Review: Data Types

```
]: x +=1 #This is a comment. No unary operators (x++ will not work)
   print(x)
   y **=2
   print y
```

```
4
9.0
```

```
]: a,b = True,False
   mystring = 'ids676'
   print a,b,mystring,'. In upper case: ' + mystring.upper()
```

```
True False ids676 . In upper case: IDS676
```

Python Review: List and Tuple

Dictionary, List, Tuple, Set

```
mylist = ['i','d','s']
mytuple = (5,7,6)
print mylist, mytuple
```

```
['i', 'd', 's'] (5, 7, 6)
```

```
mylist[0] = 'c'
mylist[1] = 'b'
mylist[2] = 'a'
mylist.append(5)
mylist.extend([7,6])
print mylist
```

```
['c', 'b', 'a', 5, 7, 6]
```

Python Review: Dictionary & Set

```
mylist[:2] = 'a','a'
print mylist
print set(mylist) #a set object will have unique elements
```

```
['a', 'a', 'a', 5, 7, 6]
set(['a', 5, 6, 7])
```

```
course = {} #An empty dictionary/hash-map
course[mytuple] = 'Advanced Prediction Models'
course['572'] = 'Data Mining'
print course
```

```
{(5, 7, 6): 'Advanced Prediction Models', '572': 'Data Mining'}
```

Python Review: Naïve for-loop

```
for x in mylist: #A for loop
    print x
```

```
a
a
a
5
7
6
```

Python Review: Function

Functions

```
import math, numpy
def softmax(z):
    return (1.0/(1+math.e**(-z)))
print softmax(-20)
print softmax(numpy.asarray([-1,0,1]))
```

```
2.06115361819e-09
[ 0.26894142  0.5         0.73105858]
```

Python Review: Numpy

Numpy

```
a = numpy.array([-1,0,1])
print a,type(a),a.shape,a.dtype
b = numpy.array([[1.0,2,3],[1,2,3]])
print b, type(b), b.shape,b.dtype
```

```
[-1  0  1] <type 'numpy.ndarray'> (3,) int64
[[ 1.  2.  3.]
 [ 1.  2.  3.]] <type 'numpy.ndarray'> (2, 3) float64
```

```
c1 =  b[1:,0:2]#note the slice indexing
print c1,c1.shape
c2= b[1,0:2] #note the integer indexing
print c2,c2.shape
```

```
[[ 1.  2.]] (1, 2)
[ 1.  2.] (2,)
```

Python Review: Numpy

```
print b>2, b[b>2]
```

```
[[False False  True]
 [False False  True]] [ 3.  3.]
```

```
x = numpy.array([[1,2],[3,4]])
y = numpy.array([[1,1],[1,1]])
z = numpy.array([1,1])
print x*y #elementwise product
print x.dot(z) #matrix vector product
```

```
[[1 2]
 [3 4]]
[3 7]
```

```
print x.sum(), x.T
```

```
10 [[1 3]
 [2 4]]
```

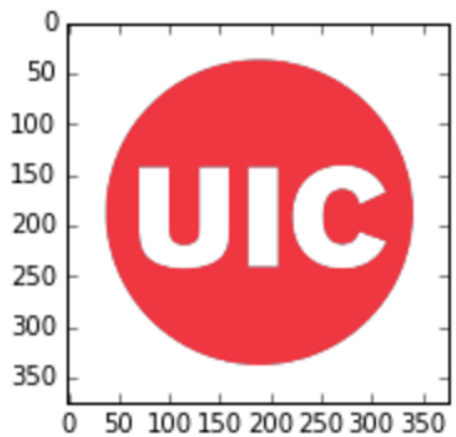
Python Review: Scipy Images

Scipy images

```
from scipy.misc import imread, imresize
%matplotlib inline
import matplotlib.pyplot as plt

img = imread('uic-logo-circle-red.jpg')

# Show the original image
plt.subplot(1, 2, 1)
plt.imshow(numpy.uint8(img))
plt.show()
```



Additional resources: 1. http://cs231n.github.io/python-numpy-tutorial/
2. http://docs.scipy.org/doc/scipy/reference/index.html

Some Relevant Packages in Python

- Keras
  - An open-source neural network library running on top of various deep learning frameworks.
- Tensorflow
  - A programming system to represent computations as graphs
  - Two steps:
    - Construct the graph
    - Execute (via session)

Questions?