ECE 210 Honors
Python Session 1

Python Programming

- Object Oriented Programming
  - <Class/Library name>.<Function/method name>
    - Ex: import numpy as np # import library and set class name
      np.cos(np.pi/4) # to compute cos(pi), pi is defined in numpy
    - Other Libraries that are important in this class
      - scipy, matplotlib.pyplot
- Indent sensitive
- Comment Symbol : #

OBJECTS

- programs manipulate **data objects**
- objects have a **type** that defines the kinds of things programs can do to them
- objects are
  - scalar (cannot be subdivided)
  - non-scalar (have internal structure that can be accessed)

Jupyter Interface

- Inserting New Cells
  - Insert tab on the taskbar
- Help function
  - help(<function name>)
- Run individual Cells
  - CTRL-ENTER to run cells

Documentation on Numpy and Scipy and Matplotlib

- http://docs.scipy.org/doc/numpy/
- http://matplotlib.org/
- If you need help, try the documentation first

Some commonly Used Function in Numpy

- Trigonometric Functions
  - np.cos, np.sin. np.tan, np.arcsin, np.arccos, np.arctan
- Exponents and Logarithms
  - np.exp, np.log, np.log10, np.sqrt
- Complex Numbers
  - np.angle, np.real, np.imag, np.conj
- Other operations
  - np.mod, np.floor, np.ceil, np.maximum, np.minimum, np.linspace
- Constants
  - np.pi

Datatypes in Numpy

- bool: Boolean (true/false)
- int : Integer (32/64 bit, depends on computer architecture)
- int<8/16/32/64>: n-bit signed integer
- uint<8/16/32/64>: n-bit unsigned integer
- float : 64-bit floating point
- complex: 128-bit complex floating point

Plotting In Python

1. Import the matplotlib.pyplot library
   - import matplotlib.pyplot as plt
2. Create an figure object from the pyplot class
   - plt.figure(<num>)
3. Plot the Graph
   - plt.plot(<x>,<y>,<line style>,label="<legend name>")
4. Add labels and titles
   - plt.xlabel<label>
   - plt.ylabel<label>
   - plt.title<title>
5. Display plot
   - plt.legend(loc = 'best')
   - plt.show

Axis Bounds and Scaling

- Set x and y axis bounds
  - plt.xlim([<x_min>,<x_max>])
  - plt.ylim([<y_min>,<y_max>])

- Make figures tightly packed
  - plt.tight_layout()

Subplots & Inserts

- Subplot
  - plt.subplot(nrows=<int>, ncols=<int>)

- Multiple figures on the same graph
  - Do not need to use hold on and hold off. Just type two plot commands for each plot. They should be under the same figure number.

Displaying Information in Python

- print(<string>) to display the string
- Convert Numbers to Strings
  - str(<variable>)
- Concatenate strings
  - s=‘<1st string>’ + ‘<2nd string>’ + str(<variable>)
- Split and join
  - Assume you have a list a = [‘1’, ‘2’, ‘3’]
  - Concatenate them using a user defined literal: “<someseperator>”.join(a)
  - To split a string separated by a specific separator into a list:

    “1+2+3”.split(‘+’) # would return [‘1’,’2’,’3’]

For loop in python

for <index> in <list>

<\t> <Line 1 to run>

<\t> <Line 2 to run>

<Line to not run in for loop>

- Indent for every for/while/if statement
  - Nest accordingly

```
for <index> in <list>
        for <index> in <list>
                <code to run>
```

If, elif, else

- If <condition>:
  <\t><expression>
  elif<condition>
  <\t><expression>
  else
  <\t><expression>

Display Formatting in Python

- Similar to that in C and related programming languages
- S2= ‘%<d/f/h/b>’ % <value>
  - %d: decimal
  - %f: float
    - %<>.<decimal places>f
  - %h: hexadecimal
  - %b: binary

Indexing & Lists

- Index in Python **start from 0, not from 1**
- Need to use commas to separate values & use “[ ]”
- Lists values do not need to be of same type
  - You can put strings and numbers in the same list
- Define lists (much like an array in Matlab)
  - l=[1,2,3,4,5,6,7,8,9]
- Index Syntax: <start>:<end>:<step size> or range(<start>,<stop>,<step>)
  - Unlike Matlab, the last element is not indexed
  - If Unspecified, step size=1

Tuple

- Use “( )”
- Use it like a list, but cannot be modified
- Define tuples
  - Loc=(10,20,30)

Arrays

- Array is not a native type, need to import numpy
- To define 1-D array
  - np.array(<list of numbers>)
  - np.arange(<start>,<stop>,<step>)
- To define 2-D array
  - np.array([[<row 1>],[<row 2>],…])
- Get size of array
  - np.shape(<array>)
- Get transpose of array
  - <array>.transpose()

Array Arithmetic

- Cell-wise Multiplication/Division
  - C=A*B
- Dot product
  - D=np.dot(A,B)
  - If A,B are matrices, then it is equivalent to a Matrix Multiplication
- Cross product
  - D=np.cross(A,B)