An Introduction to Programming With Python

September 22th, 2017, Cal State Fullerton

Center for Computational and Applied Mathematics

(Part-2)

Arjang Fahim, PhD

Department of Biomedical Engineering

University of California, Irvine

Libraries in Python
Introducing numpy, pandas, matplotlib

Enhancing Python with Libraries

- Out of the box there are many functions available to you in Python

- But you are not limited to the ‘out of the box’ version of Python and there are entire libraries of functions that are available to use within Python

Python's Standard Library

- Python's standard library is very extensive
- The standard library provide access to functionality such as I/O that would otherwise be inaccessible to Python programmers
- Additionally it provides standardized solutions to many problems that occur in programming
- Often in Python it's worth checking to see if someone has done it before you reinvent the wheel

Common Modules, Packages, Solutions

- **String Services**
  `string` - Common string operations
  `re` - Regular expression operations
- **Data Types**
  `datetime` - Basic date and time types
  `collections` - High-performance container datatypes
  `array` - Efficient arrays of numeric values
- **Numeric and Mathematical Modules**
  `math` - Mathematical functions
  `decimal` - Decimal fixed point and floating point arithmetic
  `random` - Generate pseudo-random numbers
- **File and Directory Access**
  `glob` - Unix style pathname pattern expansion
- **File Formats**
  `csv` - CSV File Reading and Writing
- **Generic Operating System Services**
  `os` - Miscellaneous operating system interfaces
- **Python Runtime Services**
  `sys` - System-specific parameters and functions
- **Many, many, many more**
  https://docs.python.org/2/library/

Using external modules and libraries

- It’s nice that all of these modules and libraries exist, but it isn’t apparent yet how to interface with them
- Python has a syntax for imports, for example

  import math

  import math.pi

- The above makes the module math available for the code. The second line prints the attribute pi from the math module

- To see all of the functions available to the math module see

  https://docs.python.org/2/library/math.html

Using external modules and libraries

- There is an alternative way of importing modules and packages where we shorten their names

- This is especially useful because using the ‘dot’ – notation of the python objects can require a lot of typing and programmers want to minimize their typing as much as possible

- This motivates the “**as”** statement

```
import math as m
print m.pi
```

Importing specific functions

- If we wanted to be really aggressive we could import all of the functions and attributes from a library with the following notation

  from math import *

- Generally this is unadvisable ! **(Don’t do it)**
- Consider

  pi = 5

  from math import pi

  print pi

- We see that pi has been replaced by the value from the math module. We’re getting into the types of programming that makes us dangerous if we’re not careful

Command-line Example

Libraries for Data Science

Package Name	Description
numpy	NumPy is the fundamental package for scientific computing with Python.
scipy	SciPy is an open source Python library used for scientific computing and technical computing.
pandas	The Python Data Analysis Library - pandas is an open source library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.
sklearn	Scikit-learn is a machine learning library for the Python programming language. It features various classification, regression and clustering algorithms and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.
statsmodels	Statsmodels is a Python module that allows users to explore data, estimate statistical models, and perform statistical tests.
matplotlib	matplotlib is a python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms.
seaborn	Seaborn is a Python visualization library based on matplotlib. It provides a high-level interface for drawing attractive statistical graphics.

Getting started with NumPy

- NumPy is imported into python as the identifier np

  Import numpy as np

- This is the first package to get familiar with for working with data

- Provides data structures for creating arrays and matrices and for manipulating them

NumPy arrays

- NumPy arrays are one of the reasons for the packages success

  my_array = np.array([1, 2, 3.])

- They can also be used to create multidimensional arrays

  matrix_array = np.array([[1, 2], [3, 4]])

- They look awfully familiar to lists …

NumPy arrays VS. Lists

- A simple check of the methods available to the two objects shows the major advantages of lists vs. numpy arrays

```
a = [ [ 1, 2] , [3, 4] ]
b = np.array( [ [ 1, 2], [3,4] ])
print dir(a)
print dir(b)
```

Some Methods and Attributes for Arrays

- Here are a sample of some useful methods and attributes of arrays

Syntax	Description
`array.shape`	Tuple of array dimensions.
`array.size`	Number of elements in the array.
`array.T`	Transpose of the array
`array.max(axis)`	Returns the maximum value of the array along a given axis
`array.round(digits)`	Return an array with each element rounded to the given number of decimals.
`array.sum(axis)`	Return the sum of the array elements over the given axis.
`array.cumsum(axis)`	Return the cumulative sum across a given axis
`array.mean(axis)`	Return the mean along a given axis
`array.var(axis)`	Return the variance along a given axis

But numpy has Matrix Object?

- NumPy provides, in addition to np.array( ), an additional np.matrix( ) type that you may see used in some existing code. Which one to use?
    - Short answer? Use np.array ( )
- NumPy provides the matrix class for matrix algebra, but you can almost consider it a “subclass” in numpy.
- Additionally, array are the standard vector/matrix/tensor type of numpy. Many numpy functions return arrays, not matrices.
- Any algebra you would want to use on matrix object is available to an array object

Matrix Operations in numpy

- Matrix multiplication can be done using the .dot() method

```
A = np.array([ [ 1, 2], [3, 4] ] )
b = np.array( [1, 2])

print  np.dot (A, b.T)     # Matrix Mult
print np.dot (A, b)        # numpy fixes dim
print A.dot (b)            # Chained method
print A * b                # Element-wise
```

Linear Algebra Operation – numpy.lialg

- There are other linear algebra functions within numpy.linalg

```
A = np.array([ [ 1, 2], [3, 4] ] )
b = np.array( [1, 2])

A_inv = np.linalg.inv(A)     # inverse
A_det = np.linalg.det(A)    # determinant
A_svd =np.linalg.svd(A)     #sing. Val.decomp
```

Examples numpy1.py, numpy2.py

Getting started with Pandas

- Pandas builds on the functionality of numpy, but introduces data frames

  import pandas as pd

- DataFrames are two-dimensional tabular data structure (very similar to data frames in R)

- Here the data is actually composed of numpy array! Therefore much of what we just learned is still useful

- Named columns for easy access

- We now have almost everything we had with numpy array, but with more functionality

A few ways to create a DataFrame

```
fake_data = np.array ( [ [ ‘AJ’,  36], [‘Brett’, 29], [‘Jake’, 26], [‘Bob’, 57 ] ] )
data_set = pd.DataFrame ( fake_data, columns = [‘Name’, ‘Age’])
print data_set.Name
print data_set.Age
```

Or

```
fake_data = {‘Names’ : [‘AJ’, ‘Brett’, ‘Jake’, ‘Bob’], ‘Age’ : [36, 29, 26, 57] }
print data_set.Names
print data_set.Age
```

Reading Files with Pandas is Easy

- Another reason for using pandas is the ease with which it brings data into python

```
pd.read_csv (some_file)
```

- This function can take in many arguments to customize its behavior, but it generally works well without any modifications