**Numerical and** Scientific Packages

Scipy and friends

By far, the most commonly used packages are those in the SciPy stack. *We* will focus on these in this class. These packages include:

• Numpy

• Scipy

• Matplotlib - plotting library.

• IPython – interactive computing.

• Pandas – data analysis library.

• Sympy – symbolic computation library*.*

Numpy

Let's start with NumPy. Among other things, Numpy contains:

• A powerful N-dimensional array object.

Sophisticated (broadcasting/universal) functions.

• Tools for integrating C*/*C++ and Fortran code.

• Useful linear algebra, Fourier transform, and random

number capabilities. Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data.

Numpy

The key to NumPy is the ndarray object, an *n* dimensional array of homogeneous data types, with many operations being performed in compiled code for performance. There are several important differences between NumPy arrays and the standard Python sequences:

• NumPy arrays have a fixed size. Modifying the size

means creating a new array.

• NumPy arrays must be of the same data type, but this

can include Python objects. More efficient mathematical operations than built-in sequence types.

Numpy Datatypes

To begin, Numpy supports a wider variety of data types than are built-in to the Python language by default. They are defined by the numpy.dtype class and include:

• intc (same as a Cinteger) and intp (used for indexing)

• int8, int16, int32, int64

• uint8, uint16, uint32, uint64

• float16, float32, float64

• complex64, complex128

• bool\_, int\_, float\_, complex\_ are shorthand for defaults.

These can be used as functions to cast literals or sequence types, as well as arguments to numpy functions that accept the dtype keyword argument.

(1870Țn, əd67p

Əd/7p•z ««< (87uỊN=ədáp '[2 'I 'O]) leite

z <<< (870Țndu=əd7p '€) əbuene du = z <<< ([B 'Z 'I]) AeJIE

I ««< ([8'Z'I]) quļ• du = À «<<

O'I

x <<< ( 01) 787e0TI•du = x <<< du se Adunu **710dut** <<<

:səjdwexə əwos •

səd jezeg Adunn

Numpy Arrays

There are a couple of mechanisms for creating arrays in NumPy:

• Conversion from other Python structures (e.g.,

lists, tuples).

• Built-in NumPy array creation (e.g., arange,

ones, zeros, etc.).

• Reading arrays from disk, either from standard

or custom formats (e.g. reading in from a CSV

file).

• and others ...

([[0+ • Ę '\*[+\*[ ] '[[°0+:0 '[°0+ 0 ] [.0+ • 2 '[0+ •[ ]])eledu = x << ([(.E' ÇI+I)'[oʻ0]4[0.Z']]])/eIIedu = x <<<

([o 'I 'E ']) Xente • du = x <<<

(boʻI'ęʻZ])*.*euve du = x <<<

sequence types like lists and tuples. function. The most obvious examples are an ndarray through use of the array()

in an array-like container can be converted to

• In general, any numerical data that is stored

*s*henje Adunn

Numpy Arrays There are a couple of built-in NumPy functions which will create arrays from scratch.

• zeros(shape) -- creates an array filled with O values

with the specified shape. The default dtype is float64.

>>> np.zeros((2, 3)) array([[ 0., 0., 0.], [ 0., 0., 0.]]).

• ones(shape) -- creates an array filled with 1 values.

arange() -- creates arrays with regularly incrementing values.

>>> np.arange (10) array([0, 1, 2, 3, 4, 5, 6, *7*, 8, 9]). >>> np.arange (2, 10, dtype=np.float) array([ 2., 3., 4., 5., 6.*, 7.,* 8., 9.]). >>> np.arange *(*2, 3, 0.1) array( [ 2. , 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.*7,* 2.8, 2.91).

\*(Tʻo) je Jəlu! ƏY7 jƏAO sieot wopues 47!M shenje səfləJ– (ədeys)wopues'wopues .

( [[89826*1*860 -68069890° 0 *‘L*8189I*LL:*0 ]

“[6IB *L*OOSE : 0 49166SLIB O*L*698899*4.*0 ]]) ARIJE

((ε'z)) ropues dopues du «

5 °£

8.2 2.2 '9I ' 'I ])enje (9 " B "I)əɔedsuȚtodu «<<

values. between the specified beginning and end number of elements, and spaced equally

• linspace() -- creates arrays with a specified Numpy Arrays

[[[*L* 9]

[9]]

[[Ɛ z]

[I oll] o quȚId <<<

np.arange (8).reshape (2,2,2)

[[8 */* 9]

[9 Ę ]

[2 I 0]] q qusid <<<

statement.

*pr*int

np.arange *(*9).reshape (3,3)

= q ««<

array([0, 1, 2])

a

>>>

done with the

array can be

• Printing an

[Z I 0]

**e quļıd** <<< (€)əbueje.du = e ««< du se Adunu **710du**ț <<<

Numpy Arrays

0 1 2 3 4 5 6 7 8 9

[I-ʻI] x <<<

8

[ɛʻ[] x <<< *Teu*o*ȚSUəwȚP-2 ST* X M*OU # (*9^2) = ədeys.x <<<

dimensional indexing.

• Multi-dimensional arrays support multi

68 *L* 9 SH E ZIO

[2-]x <<<

*z* [2]\* ««<

>>> X = np.arange (10)

jensn se pəys!jdwoɔɔe si Buļxəpup uopsuəw!p-əPUIS •

Buixopu|

Indexing

• Using fewer dimensions to index will result in

a subarray.

>>> x[0] array([0, 1, 2, 3, 4])

• This means that x[i, j] == x[i][j] but the

second method is less efficient.

([[LZ 'BZ ‘IZ] [ƐI *'*OI *‘L* ]]) XeIIe

[ɛ::-2:9:[]. ««< *(*L'S)ǝdeysəI (GE)əbuee.du = *À* «««

([s 'E '[]) AeIJE

[Z:*L*:[] x <<< ([z 'I '0]) Kelde

[-:]x ««< ( [= ''2] ) *A*RIJE

[9:2]x ««< (01) əbuene • du = x <<<

Python sequences.

• Slicing is possible just as it is for typical

Indexing

([91 '6 ' 'I ‘0 ]) Xente

q\*e <<<

([96820899*02* - '80002II8 I *L*ZE*L*6260o6 '9860LEI 5 8

10 l) *l*eite (e) uȚsoduxo I ««<

**array.**

True], dtype=bool)

False,

will modify the existing array([False, False, False, array([ 0, 1, 4, 9, 16]) Operations like \* = and +=

E<e ««<

<\*\* <<< ( [0 '0 '0 '0 '0l) *A*ende

q-e ««< ([8 9 'E *'*Z '0]) Xente

que ««< (9)əbuede du = q <<< *(*9) Əbuene du = Z <<<

resultant elements. is a new array with the

element-wise. The result

• Basic operations apply Array operations

multiplication. perform matrix

07 fɔnpojd

([[' *"*Z ] *'*[\* *-*0 ]])IBIJE

(qre) pop•du <<< ([[ ".0] '[:0 1:0 ]]) ARIJE

que «<< ([[ɛ '2] '[[ '0]]) BIJE

q ««< (2ʻz) ƏdeysəI (5)əbuejedu = q ««<

I = [['Ile ««<

I = [oʻole <<< ([[:0 10 ] '[10 ]]) Xente

e ««< *(*Zʻz) ƏdeysəI (5) Sovəz du = e <<<

perform a dot to specifically wise, you need

**A**

**A**

**-งนอนais euop** SI uoitellanu

**อวน!S •**

suoljеләdo Хел*у*

cos, etc...

‘uis 'ppe Pubs 'dxə əpnu! pərdde əq osje hew

([9€6LISO8° 0 [6899189.0 l)ecIE

(I=SȚXP) uȚwe ««<

([28919€69.0 860€8686° 0 '1808888*4*°0 l) leide

(0=SȚXe) xewe <<< E B 16 *L*EO1986LISOBO

() uȚWe <<< 59IZ*ZL*88EIZBLO8I'D

*(*) uns.e <<<

( [[986*L*IGOÐ° 0 “SZIL61290 '1808888*L*.0 ]

‘[28919€690 '860€8686°0 '[6899189.0 ]])eIIZ

e ««< ((€ʻZ) ) wopue I wopue du = z <<<

which Universal functions

objects. methods of ndarray

• There are also some

Array operations

[ 8., 9., 5*., 7., 2*., 5.]]).

'[*L '*806*L L '*06 ]])ende

*(*)ǝsodsueto e ««<

[[•ş :]

[.Z '8 ] [*L* •6] [§ *L* ] [.6 L ]

[.8 °6 ]]

e quȚad ««< (7ʻ9) = ədeyse ««< (['g *'L'oz*

new shape. the array with a return a copy of reshape(size) will

array([ 9., 8., *7.,* 9.*, 7.,* 5., 9., *7.,* 8.,

place. modify an array in resize(size) will

methods.

*(*) TƏNBI° e ««<

| (+ )

edeyse <<< [[Og *'L* 'Z '8] [*.* '6 'g '/] [°6 *7* 8.6 ]]

e quȚad <<< *(((*5'€) ) wopuen wopues dux01) JOOTI'du

= e ««<

jo jəqunu e aq pəzejndịuew

əq uey ədeys Келе uy •

suojejədo henny

([[GO- 'GoI]

*'*I 'I *'* .z-]]) Kedje *ƏSTƏQUỊ # (*e) AUỊ <<<

([[•] '.2 ] '[•"]])eine *(* ) əsodsuei7.««<

[[ P 8 ]

[•Z 'I ]] **e quȚId** <<<

( [COB '0€] [0.2 '0°1]l) Xedie = Z <<< **\* 410dut** 6IPUȚI•Adunu wont «««

**\* 730duğ** Adunu wolj <<<

"əјпрош елqə8је JEəU!| SI! SI ƏSeyed Idunn əyy Suisn joj suosees uowwoo jsou əyy jo əUO •

eJqəsje Jeau!

What is pandas ?

*• Panda*s is Python package for data analysis.

• It Provides built-in data structures which

simplify the manipulation and analysis of data

sets.

• Pandas is easy to use and powerful, but "with

great power comes great responsibility"

• We cannot teach you all things Pandas, we must focus on how it works, so you can figure

out the rest on your own.

• http*:/*/pandas.pydata.org/pandas

docs/stable*/*

Watch Me Code 1

Pandas Basics

• Series

• DataFrame

• Creating a DataFrame from a dict

• Select columns, Select rows with

Boolean indexing

Libraries - Pandas

• A popular library for importing and managing datasets in Python

for Machine Learning is 'pandas'.

Keyword to import a library

Keyword to refer to library by an alias (shortcut) name

import pandas as pd

**Used for:**

**• Data Analysis**

**• Data Manipulation**

**• Data Visualization**

PyData.org : high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

Pandas – Indexed Arrays

• Pandas are used to build indexed arrays (1D) and matrices (2D), where columns and rows are labeled (named) and can be accessed via the labels (names).

Columns (features)

index

Row (samples)

index

x1

x2

x3

x4

ra*w* data

one

1

4

8

2

5

9

3

6

10

two

**1**

4

8

***2***

5

9

**3**

6

10

***4***

7

11

11

three

Panda Indexed Matrix

Pandas - Series and Data Frames

Pandas Indexed Arrays are referred to as Series (1D) and Data Frames (2D).

Series is a 1D labeled (indexed) array and can hold any data type, and mix of data types.

Series

Raw data

Column Index Labels

s = pd.Series( data, index=[ 'x1', 'X2', 'x3', 'x4']).

Data Frame is a 2D labeled (indexed) matrix and can hold any data type, and mix of data types.

Data Frame

Row Index Labels

Column Index Labels

df = pd.DataFramel data, index=['one', 'two'], columns=['x1', '*X*2', 'X3', 'x4']).

Pandas: Essential Concepts

• A Series is a named Python list (dict with list as value). { 'grades' : [50,90,100,45] }

• A DataFrame is a dictionary of Series (dict of series): { { 'names : ['bob', 'ken', 'art', 'joe']}

{ 'grades': [50,90,100,45] }

Check Yourself Series or DataFrame?

*M*atch the code to the result. One result is a Series, the other a DataFrame

**Quarter Sold**

100

120

90

1.df [ ‘Quarter']

2.df [

[ ‘Quarter'] ]

3 Q4

150

Check Yourself Boolean Index

**Quarter Sold**

Which rows are included in this Boolean index?

100

120

90

&

df[ df['Sold']

< 110 ]

150

Pandas – Selecting

• Selecting One Column

Selects column labeled x1 for all rows

x1 = df[ 'x1']

• Selecting Multiple Columns

*Note: df['x1':'x3' ) this python syntax does not work!*

Selects columns labeled X1 and x3 for all rows

Selects columns labeled x1 through x3 for all rows

x1 = df[ [ 'x1', 'x3']]

1

*4*

8

3

6

10

x1 = df.ix[ :, 'x1':'x3']

1 2 3

4 5 6

8 9 10

rows (all)

columns

Slicing function

*And many more functions: merge, concat, stack, ...*

Libraries - Matplotlib

• A popular library for plotting and visualizing data in Python

Keyword to import a library

Keyword to refer to library by an alias (shortcut) name

import matplotlib.pyplot as plt

**Used for:**

**• Plots**

**• Histograms**

**• Bar Charts**

**• Scatter Plots**

**• etc**

matplotlib.org: Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms.

Matplotlib - Plot

• The function plot plots a 2D graph.

X values to plot

Y values to plot

Function to plot

plt.plot( x, y)

• Example:

plt.plot( [1,2,3], [4,6,8 ]) # Draws plot in the background plt.show()

# Displays the plot

Matplotlib - Plot Labels

• Add Labels for X and Y Axis and Plot Title (caption)

plt.plot( [1,2,3], [4,6,8]) plt.xlabel( “X Numbers”) plt.ylabel( "Y Numbers”*)* plt.title( “My Plot of X and Y”) plt.show()

# Label on the X-axis # Label on the Y-axis # Title for the Plot

**My Plot of X and Y**

**Y Numbers**

**1**

**2**

**3**

**X Numbers**

Matplotlib – Multiple Plots and Legend

• You can add multiple plots in a Graph

plt.plot( [ 1, 2, 3 ], [ 4, 6, 8 ], label=' 1st Line') # Plot for 1st Line plt.plot( ( 1, 2, 3 ], [2, 4, 6), label='2nd Line') # Plot for 2nd Line plt.xlabel( "X Numbers”) plt.ylabel( "Y Numbers”) plt.title( “My Plot of X and Y”) plt.legend()

# Show Legend for the plots

plt.show()

**My Plot of X and Y**

---- 1st Line

---- 2nd Line

**Y Numbers**

**1**

**2**

**3**

**X Numbers**

Matplotlib - Bar Chart

• The function bar plots a bar graph.

plt.plot( ( 1, 2, 3 ], [ 4, 6, 8]) # Plot for 1st Line plt.bar()

# Draw a bar chart

plt.show()

*And many more functions: hist, scatter, ...*