


## MODULE / WEEK 1

QMB4400

DATA ANALYSIS AND OPTIMIZATION



1

---

---

---

---

---

---


---

---

## COURSE SCHEDULE

TERM DATES: April 5, 2021 – May 11, 2021

WEEK #	DATES
Week 1	Monday, April 5, 2021 (12:00 a.m.) – Sunday, April 11, 2021 (11:59 p.m.)
Week 2	Monday, April 12, 2021 (12:00 a.m.) – Sunday, April 18, 2021 (11:59 p.m.)
Week 3	Monday, April 19, 2021 (12:00 a.m.) – Sunday, April 25, 2021 (11:59 p.m.)
Week 4	Monday, April 26, 2021 (12:00 a.m.) – Sunday, May 2, 2021 (11:59 p.m.)
Week 5	Monday, May 3, 2021 (12:00 a.m.) – Sunday, May 9, 2021 (11:59 p.m.)
Week 6	Monday, May 10, 2021 (12:00 a.m.) – Friday, May 11, 2021 (11:59 p.m.)



2

---

---

---

---

---

---

---

---


## COURSE BOOKS

The course book is available to you as an eText and it is located on the Course Menu. If you prefer a hardcopy of the course book, the optional purchase information is as follows:

McKinney, Wes (2013). *Python for Data Analysis*. Sebastopol, CA. O'Reilly Media, Inc. [ISBN: 978-1-449-32362-2]

Optional (but Recommended)

American Psychological Association. (2020). *Concise rules of APA style* (7th ed.). <https://doi.org/10.1037/0000165-000>.



3

---

---

---

---

---

---

---


---

## PYTHON LIBRARIES FOR DATA SCIENCE

- Many popular Python toolboxes/libraries:
  - NumPy
  - SciPy
  - Pandas
  - SciKit-Learn
- Visualization libraries
  - matplotlib
  - Seaborn

and many more ...

*All these libraries  
are installed on the  
SCC*



4

---

---

---

---

---

---


---

---

## Numeric and Scientific Applications

As you might expect, there are a number of third-party packages available for numerical and scientific computing that extend Python's basic math module. These include:

- NumPy/SciPy – numerical and scientific function libraries.
- Numba – Python compiler that support JIT compilation.
- ALGLIB – numerical analysis library.
- Pandas – high-performance data structures and data analysis tools.
- PyGSL – Python interface for GNU Scientific Library.
- ScientificPython – collection of scientific computing modules.



5

---

---

---

---

---

---


---

---

## PYTHON Libraries for Data Science

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.



6

---

---

---

---

---

---

---

---

## NumPy

### NumPy:

- introduces objects for multidimensional arrays and matrices, as well as functions that allow to easily perform advanced mathematical and statistical operations on those objects
- provides vectorization of mathematical operations on arrays and matrices which significantly improves the performance
- many other python libraries are built on NumPy

Link: <http://www.numpy.org/>



7

---

---

---

---

---

---

---

---

## 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.



8

---

---

---

---

---

---

---

---

## 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.



9

---

---

---

---

---

---

---

---

## 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 C integer) 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.



10

---

---

---

---

---

---

---

---

## NumPy Datatypes

- Some examples:

```
>>> import numpy as np
>>> x = np.float32(1.0)
>>> x
1.0
>>> y = np.int_([1,2,4])
>>> y
array([1, 2, 4])
>>> z = np.arange(3, dtype=np.uint8)
>>> z
array([0, 1, 2], dtype=uint8)
>>> z.dtype
dtype('uint8')
```



11

---

---

---

---

---

---

---

---

## 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 ...



12

---

---

---

---

---

---

---

---

## NumPy Arrays

- In general, any numerical data that is stored in an array-like container can be converted to an ndarray through use of the `array()` function. The most obvious examples are sequence types like lists and tuples.

```
>>> = np.array([2,3,1,0])
>>> = np.array([2, 3, 1, 0])
>>> = np.array([[1,2,0],[0,0],[1+1j,3.]])
>>> = np.array([[ 1.+0.j, 2.+0.j], [ 0.+0.j, 0.+0.j], [ 1.+1.j, 3.+0.j]])
```



13

---

---

---

---

---

---

---

---

## Pandas

- A fast, powerful, flexible and easy to use open source data analysis and manipulation tool
- it offers data structures and operations for manipulating numerical tables and time series.
- Built on top of the Python programming language
- Mainly used for data analysis. Pandas allows importing data from various file formats such as comma-separated values, JSON, SQL, Microsoft Excel
- Pandas allows various data manipulation operations such as merging, reshaping, selecting, as well as data cleaning, and data wrangling features.



14

---

---

---

---

---

---

---

---

## Pandas Dataframes

- A data structure in **Python** that provides the ability to work with tabular data
- Pandas** dataframes are composed of rows and columns that can have header names, and the columns in Pandas dataframes can be different types
  - the first column containing integers and the second column containing text strings
- Each value in **pandas** dataframe is referred to as a cell that has a specific row index and column index within the tabular structure.



15

---

---

---

---

---

---

---

---

## Pandas Dataframes

month	precip_in
Jan	0.70
Feb	0.75
Mar	1.85
Apr	2.93
May	3.05
June	2.02
July	1.93
Aug	1.62
Sept	1.84
Oct	1.31
Nov	1.39
Dec	0.84

The dataset to the left of average monthly precipitation (inches) for Boulder, CO provided by the [U.S. National Oceanic and Atmospheric Administration \(NOAA\)](#) is an example of the type of tabular dataset that can easily be imported into a **pandas** dataframe.



16

## Pandas Dataframes

- The characteristics (i.e. tabular format with rows and columns that can have headers) make **pandas** dataframes very versatile for not only storing different types, but for maintaining the relationships between cells across the same row and/or column.
- The structure of a pandas dataframe includes the column names and the rows that represent individual observations (i.e. records).
- In a typical pandas dataframe, the default row index is a range of values beginning at [0], and the column headers are also organized into an index of the column names.



17

## Pandas Dataframes

- The function DataFrame from pandas (e.g. `pd.DataFrame`) can be used to manually define a pandas dataframe.
- One way to use this function is to provide a list of column names (to the parameter `columns`) and a list of data values (to the parameter `data`), which is composed of individual lists of values for each row:
  - # Dataframe with 2 columns and 2 rows
  - `dataframe = pd.DataFrame(columns=["column_1", "column_2"],`
  - `data=[`
  - `[value_column_1, value_column_2],`
  - `[value_column_1, value_column_2]`
  - `])`



18

## Pandas Dataframes

- The pandas dataframe is created with a column called month containing abbreviated month names as text strings and another column called precip\_in for the precipitation (inches) as numeric values.
- For example, the first row is created using ["Jan", 0.70], with Jan as the value for month and 0.70 as the value for precip\_in.

```
import matplotlib.pyplot as plt
# Import pandas with alias pd
import pandas as pd
```



19

## Pandas Dataframes

```
# Average monthly precip for Boulder, CO
avg_monthly_precip = pd.DataFrame(columns=["month", "precip_in"],
    data=[
        ["Jan", 0.70], ["Feb", 0.75],
        ["Mar", 1.85], ["Apr", 2.93],
        ["May", 3.05], ["June", 2.02],
        ["July", 1.93], ["Aug", 1.62],
        ["Sept", 1.84], ["Oct", 1.31],
        ["Nov", 1.39], ["Dec", 0.84]
    ])

# Notice the nicely formatted output without use of print
avg_monthly_precip
```



20

## Pandas Dataframes

	month	precip_in
0	Jan	0.70
1	Feb	0.75
2	Mar	1.85
3	Apr	2.93
4	May	3.05
5	June	2.02
6	July	1.93
7	Aug	1.62
8	Sept	1.84
9	Oct	1.31
10	Nov	1.39
11	Dec	0.84

You can see from the pandas dataframe that each row has an index value, and that the default indexing still begins with [0], as it does for Python lists and numpy arrays.



21

## Pandas Dataframes

### A Quick Plot

You can plot pandas dataframe using matplotlib or using the pandas .plot() method which wraps around matplotlib.

```
f, ax = plt.subplots()
avg_monthly_precip.plot(x="month",
                        y="precip_in",
                        title="Plot of Pandas Data Frame using Pandas .plot",
                        ax=ax)
plt.show()
```



22

---

---

---

---

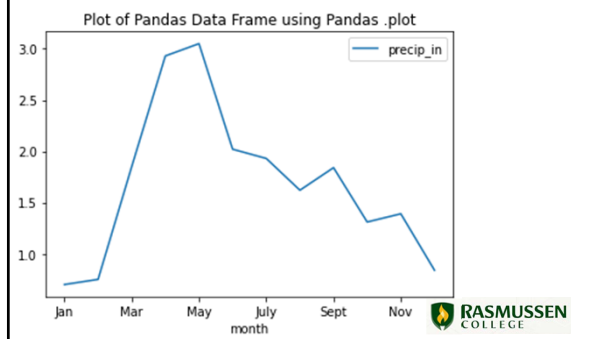
---

---

---

---

## Pandas Dataframes



23

---

---

---

---

---

---

---

---

## Pandas Dataframes

You can plot using the standard matplotlib approach. In this course we will encourage you to use the matplotlib approach which will be more flexible as you begin to create more complex plots.

```
f, ax = plt.subplots()
ax.plot(avg_monthly_precip.month,
        avg_monthly_precip.precip_in)

ax.set(title="Plot of Pandas Data Frame using Pandas .plot")
plt.show()
```



24

---

---

---

---

---

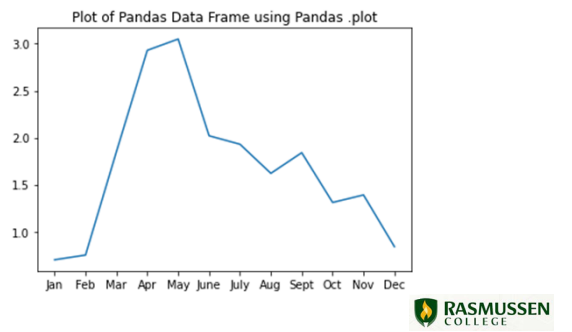
---

---

---



## Pandas Dataframes



25

---

---

---

---

---

---

---

---

## SciPy

### SciPy:

- collection of algorithms for linear algebra, differential equations, numerical integration, optimization, statistics and more
- part of SciPy Stack
- built on NumPy

Link: <https://www.scipy.org/scipylib/>



26

---

---

---

---

---

---

---

---

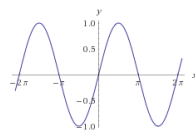
## SciPy

- We can't possibly tour all of the SciPy library and, even if we did, it might be a little boring. So let's just look at some example modules with SciPy to see how it can be used in a Python program.

Let's start with a simple little integration example.  
Say we wanted to compute the following:

$$\int_a^b \sin x \, dx$$

- Obviously, the first place we should look is `scipy.integrate`!



27

---

---

---

---

---

---

---

---

## SciPy.Integrate

Methods for Integrating Functions given a function object:

- `quad` -- General purpose integration.
- `dblquad` -- General purpose double integration.
- `tplquad` -- General purpose triple integration.
- `fixed_quad` -- Integrate `func(x)` using Gaussian quadrature of order `n`.
- `quadrature` -- Integrate with given tolerance using Gaussian quadrature.
- `romberg` -- Integrate `func` using Romberg integration.

Methods for Integrating Functions given a fixed set of samples:

- `trapez` -- Use trapezoidal rule to compute integral from samples.
- `cumtrapz` -- Use trapezoidal rule to cumulatively compute integral.
- `simps` -- Use Simpson's rule to compute integral from samples.
- `romb` -- Use Romberg Integration to compute integral from  $(2^{**k} + 1)$  evenly-spaced samples.



28

---

---

---

---

---

---

---

---

## IPython

Interesting features

- Command history
- Any Xterm command accessible via `!`
- Commands auto-completion
- Quick help through the use of `?`
- Inline and interactive graphics
- Timing and profiling tools
- Many many more ...

Best tool for exploring, debugging or work interactively. Have a look !

Link: <https://ipython.org/install.html>



29

---

---

---

---

---

---

---

---

## matplotlib

- python 2D plotting library which produces publication quality figures in a variety of hardcopy formats
- a set of functionalities similar to those of MATLAB
- line plots, scatter plots, barcharts, histograms, pie charts etc.
- relatively low-level; some effort needed to create advanced visualization



30

---

---

---

---

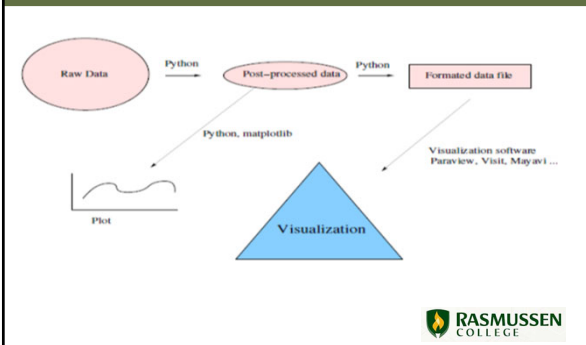
---

---

---

---

## Visualization workflow



31

---

---

---

---

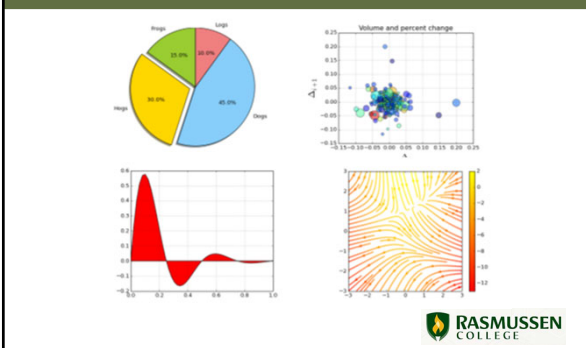
---

---

---

---

## Other features of matplotlib



32

---

---

---

---

---

---

---

---

## DISCUSSION FORUM

- Requirements
  - Thread Post must be posted by Tuesday at 11:59 PM.
  - Reply Post must be posted by Saturday at 11:59 PM.



33

---

---

---

---

---

---

---

---



MODULE 01 ASSIGNMENT:  
Food and Nutrition

You are working as a contractor on a Python script developed by a Food and Nutrition Department at a company. You need to determine to provide nutrition data on 80 cereal products. Data is stored in two different files, cereal and Cereal\_MFRCategorical, which are located in the **Course Files** folder.

- Cereal\_MFRCategorical.csv: It contains Manufacturing company category.
- Cereal.csv: It contains remaining data for each cereal.

You need to load CSV files into the Pandas and then append both the CSV files using name column.

For your submission, include the following:

- Attach a Python script.
- An output screenshot.

RASMUSSEN  
COLLEGE

37

---

---

---

---

---

---

---

---

MODULE 01 ASSIGNMENT:  
Food and Nutrition

Submit these files as a single zipped ".zip" file to the drop box below. Please check the **Course Calendar** for specific due dates.

**Note:** For help with zipping or compressing your files, visit the [How do you zip files?](#) Answers page.

The name of the file should be your first initial and last name, followed by an underscore and the name of the assignment, and an underscore and the date. (Mac users, please remember to append the ".zip" extension to the filename.) An example is shown below:

- Jstudent\_exampleproblem\_101504

RASMUSSEN  
COLLEGE

38

---

---

---

---

---

---

---

---

MODULE 01 ASSIGNMENT:  
Food and Nutrition

Criteria	Points
A correct Python script is attached.	50
Output screenshots are attached.	50
Total	100

RASMUSSEN  
COLLEGE

39

---

---

---

---

---

---

---

---