ENGINEERING PROBLEM SOLVING IN PYTHON

EMCH 561: Machine Learning for Mechanical Engineers

UNIVERSITY OF
South Carolina

U of SC. **South Carolina**

Python

Python is an **interpreted** (mostly), **high-level**, **general-purpose** programming language with an **object-oriented** approach.

- **Interpreted language:** is a type of programming language for which its instructions execute directly and freely, without previously compiling a program into machine-language instructions.

- **High-level:** a programming language with strong abstraction from the details of the computer.

- **General-purpose:** is a programming language designed to be used for writing software in the widest variety of application domains (a general-purpose language).

- **Object-oriented:** is a programming paradigm based on the concept of "objects", which can contain data, in the form of fields (often known as attributes or properties), and code, in the form of procedures (often known as methods).
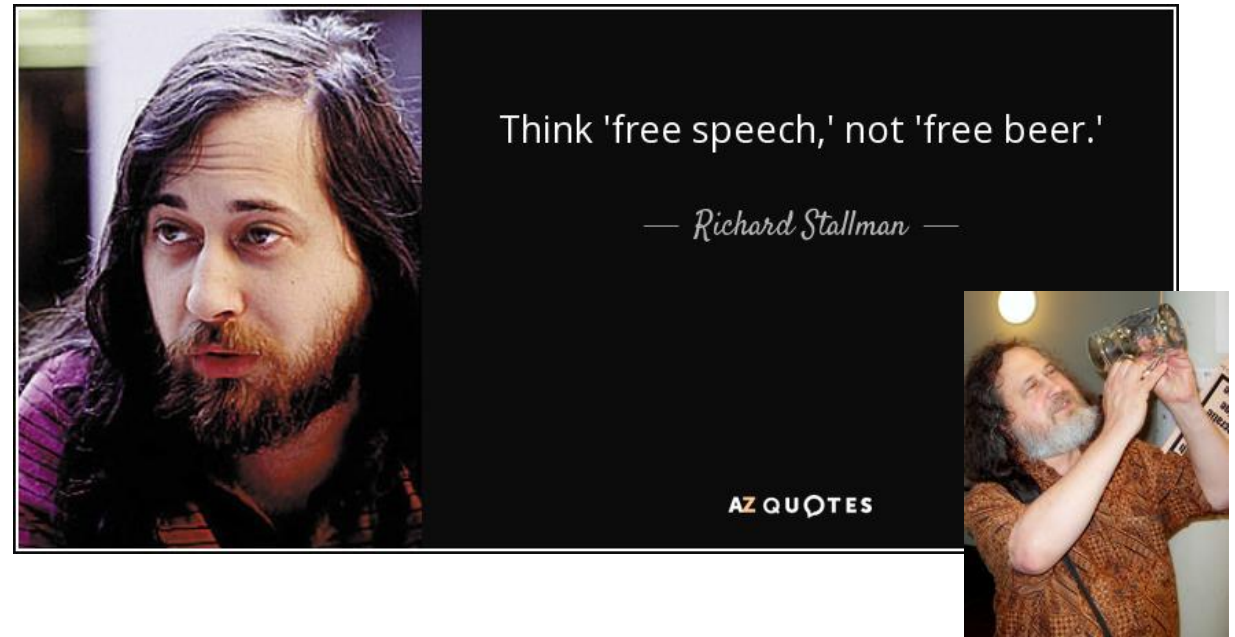
Python

Python's design philosophy emphasizes **code readability** with its notable use of significant **whitespace**. Python is **dynamically typed** and **garbage-collected**.

- **Code readability:** refers to the ease with which a human reader can comprehend the purpose, control flow, and operation of source code. It affects the aspects of quality, including portability, usability and most importantly maintainability.

- **Whitespace:** refers to the usage of whitespace in typing code. In Python, the information about blocks are expressed by their indentation.

- **Dynamically Typed:** refers to languages execute many common programming behaviors at runtime that static programming languages perform during compilation. For us, this means defining variable types.

- **Garbage-Collected:** is a form of automatic memory management. The garbage collector attempts to reclaim garbage, or memory occupied by objects that are no longer in use by the program.

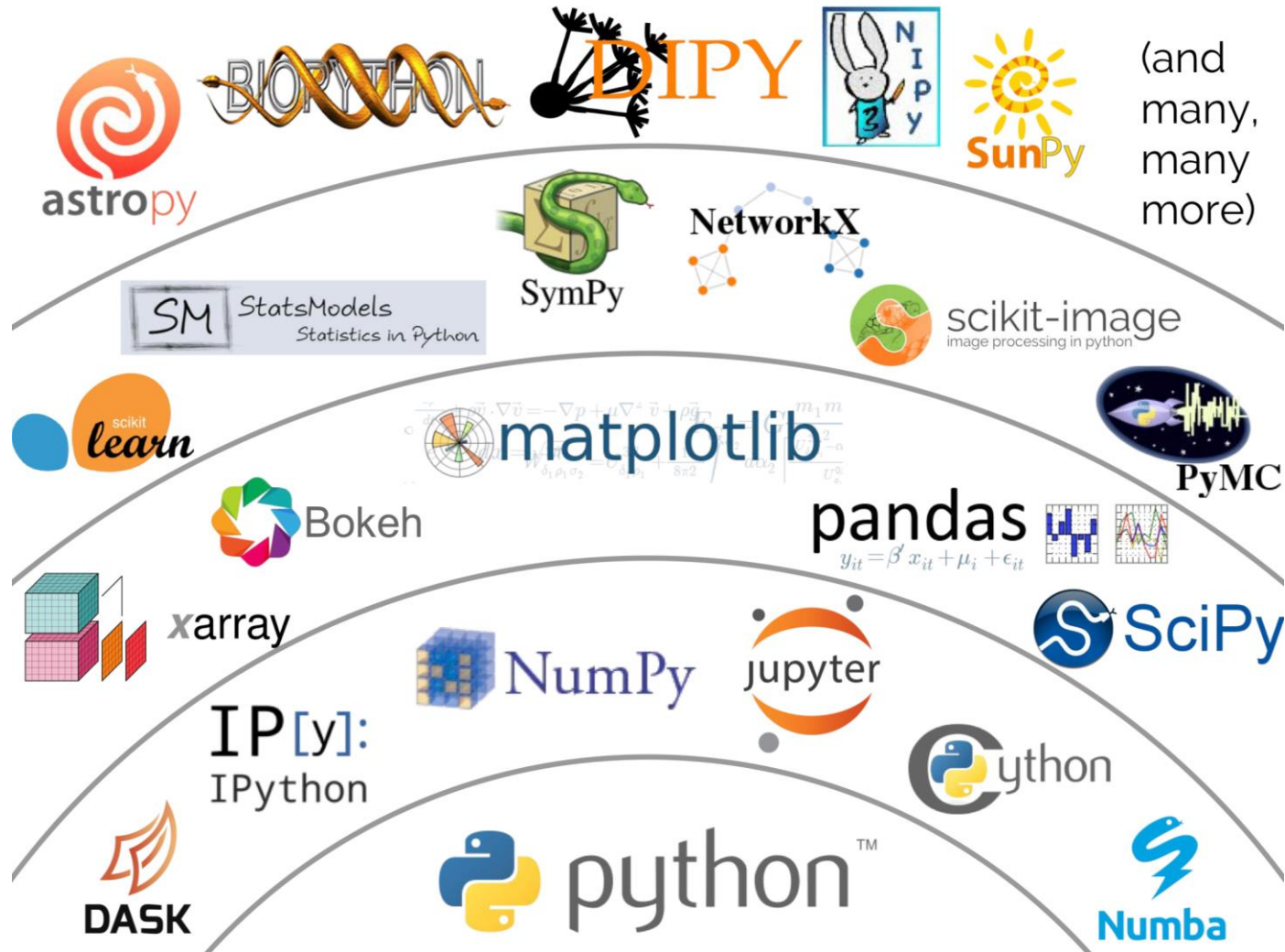# Python is Free as in beer and Free as in speech

Free beer (never)

Think 'free speech,' not 'free beer.'
— Richard Stallman —
AZ QUOTES

Free speech (First Amendment to the United States Constitution; December 15, 1791)
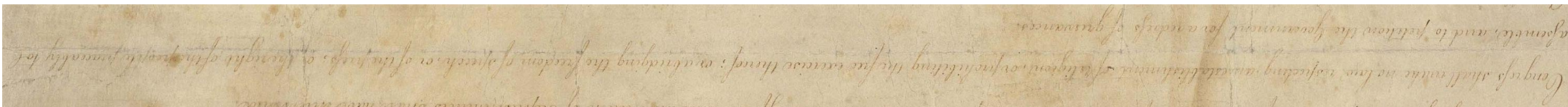
Python is a language, we use an ecosystem

In this class we will focus on a limited subset of this ecosystem:

- **IPython:** command shell for interactive computing in Python.
- **Spyder:** cross-platform integrated development environment (IDE) for scientific programming in Python.
- **Numby:** library adding support for multi-dimensional arrays.
- **SciPy:** library used for scientific computing and technical computing.
- **matplotlib:** API focused on plotting
- **scikit-learn:** library for machine learning.
- **TensorFlow:** symbolic math library, used for neural networks.

Python Distributions

A distribution is a group of packages that are assembled and more-or-less work well together. Some of these are free (beer/speech) while others there are required service packs for there use. In this class we will use Anaconda from Anaconda, Inc. Anaconda is a a free and open-source distribution for scientific computing.

# Why not just use MATLAB?

Pros:

- Works out of the box.

- Code is well managed and packages don't conflict.

- Widely used in engineering and computing.

- Well formulated for linear algebra with simple commands

- Fast implementation of code for complex problem solving

Cons:

- Not free as in speech, not free as in beer

- Each module must be paid for separately

- Not at the cutting edge of machine learning

- Difficult to keep license updated on multiple machines

- Lets you write poorly structured code.

- Code is slower than Python (personal experience)
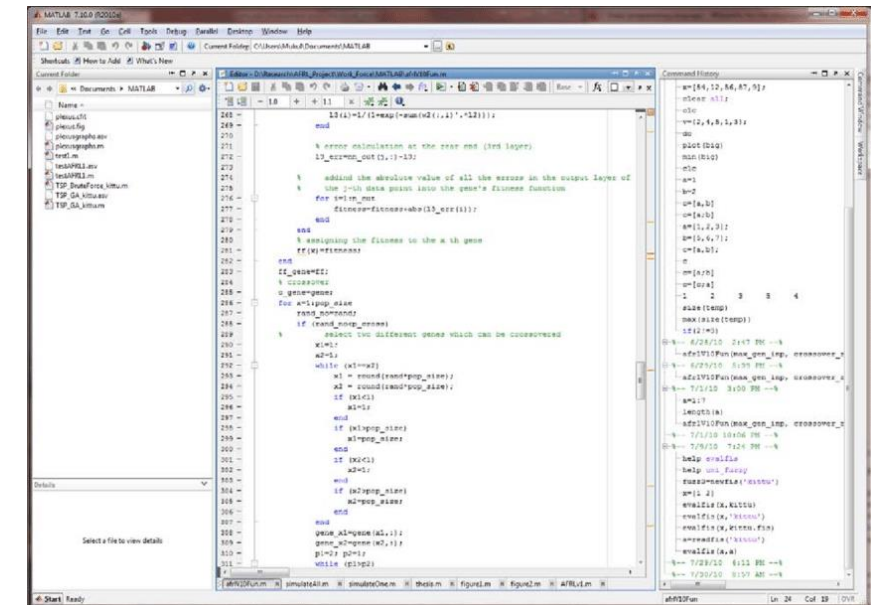
MATLAB does not use whitespace

Python

MATLAB

```python
""" whitespace.py """
from random import randrange

def numberizer():
    # Generate a random number from 1 to 10.
    return randrange(1, 11)

number = numberizer()
if number > 5:
    print("This number is big!")

class RandomNumberHolder(object):
    # Create and hold 20 random numbers using numberizer

    def __init__(self):
        self.numbers = [numberizer(x) for x in range(20)]

random_numbers = RandomNumberHolder()
```

```matlab
28      % counters for recording results
29 -    dd=1;
30 -    hh=1;
31
32 -    for history = history_lengths
33
34      % independent variable is the acceleration data
35 -    x = data.acceleration_data';
36
37      % dependent variable is the pin location
38 -    y = data.measured_pin_location';
39
40      % delete nan values from pin position
41 -    nanidx=find(isnan(y));
42 -    for i=nanidx
43 -    legitidx=find(~isnan(y(1:i)));
44 -    last_legitidx=legitidx(end);
45 -    y(i)=y(last_legitidx);
46 -    end
47 -    end
```
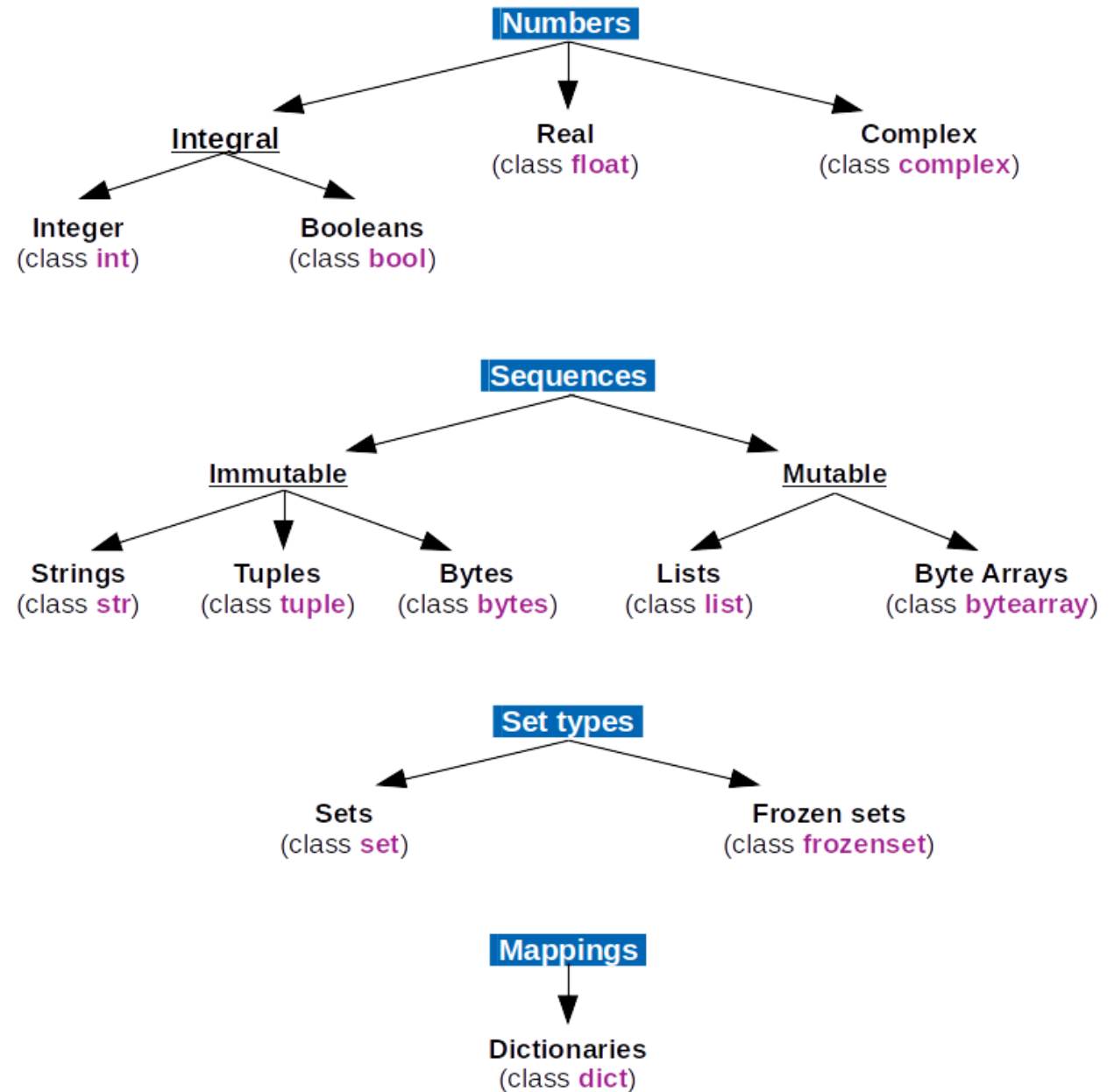
Example #1 – Plot beam vibration data in Python
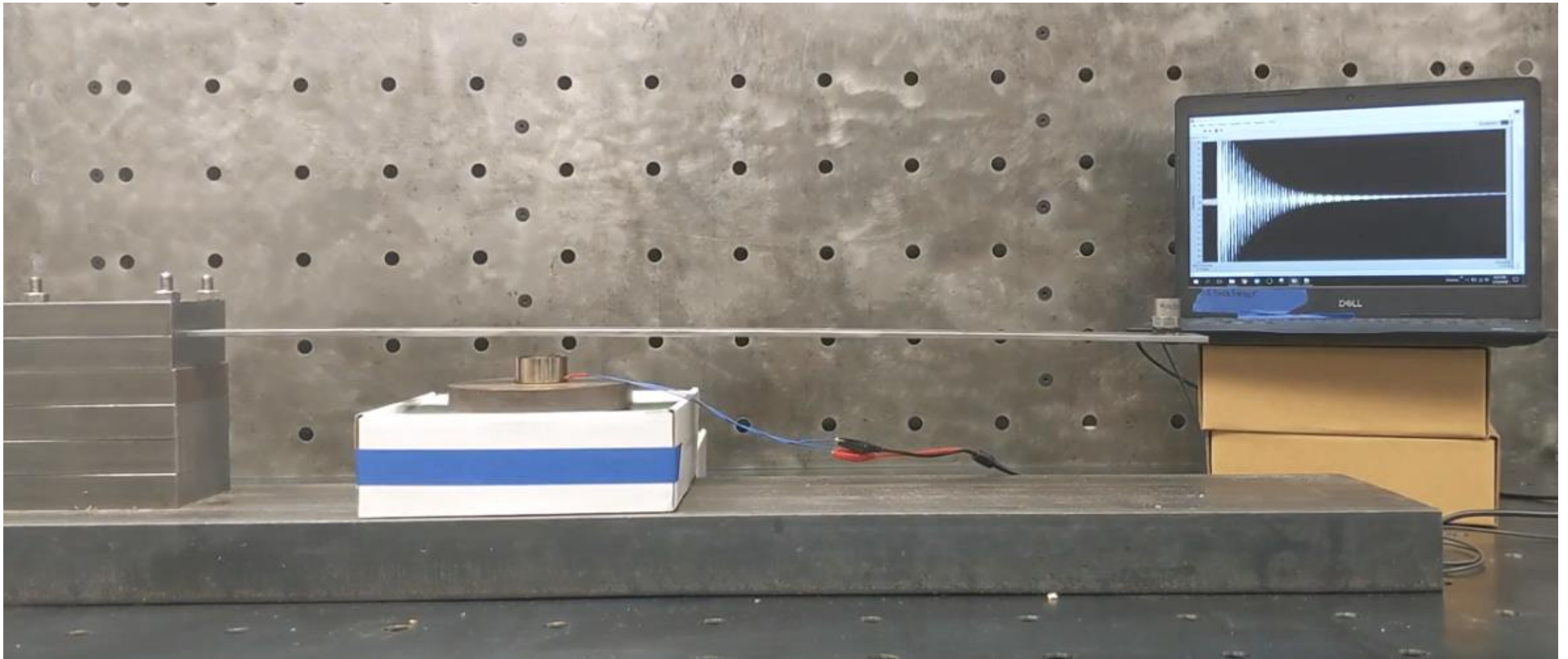
Python

Key types in Python are:

- Numbers:
  - Integer
  - Booleans
  - Real
- Sequences:
  - Strings '' or ""
  - Lists [ ]
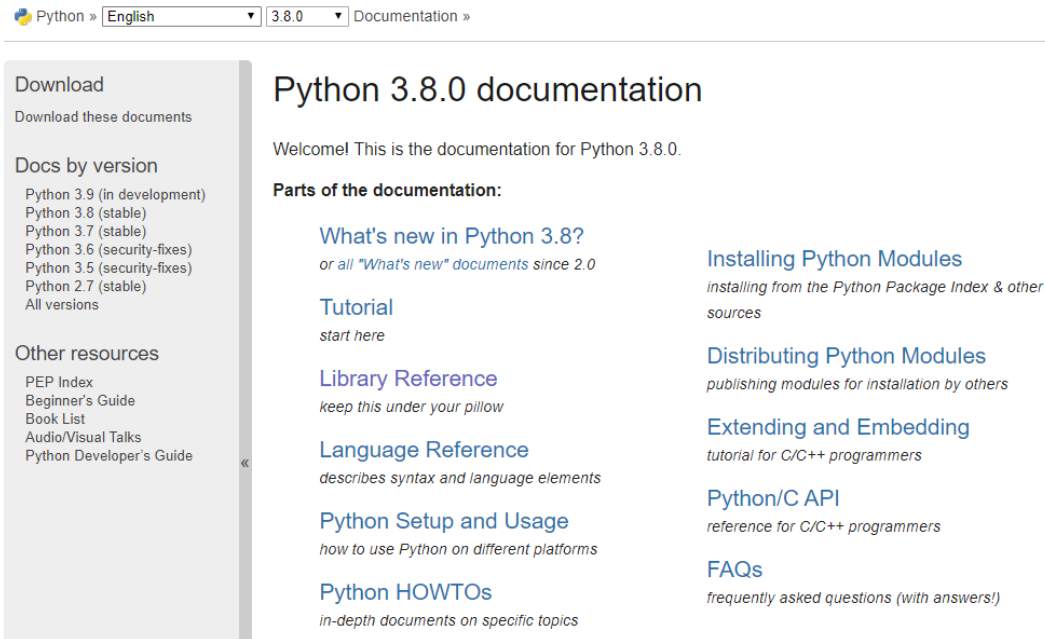 - Tuples ( )
- Mappings
  - Dictionaries { }

Numbers

Integral

Integer (class **int**)   Booleans (class **bool**)

Real (class **float**)   Complex (class **complex**)

Sequences

Immutable   Mutable

Strings (class **str**)   Tuples (class **tuple**)   Bytes (class **bytes**)   Lists (class **list**)   Byte Arrays (class **bytearray**)

Set types

Sets (class **set**)   Frozen sets (class **frozenset**)

Mappings

Dictionaries (class **dict**)

Example #2 – Pure Python

Additional Python Resources

Python documentation

https://docs.python.org/3.8/

YouTube : "Python Programming" by Derek Banas

https://www.youtube.com/watch?v=N4mEzFDjqtA

And many many more…………………….

Our Ministack
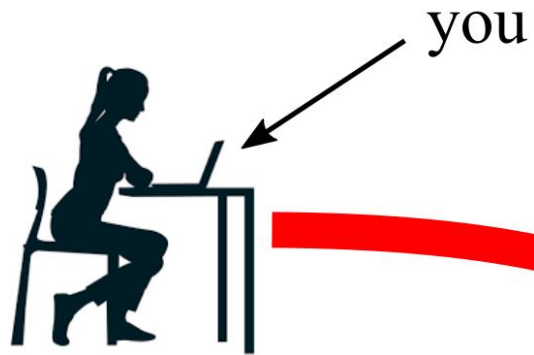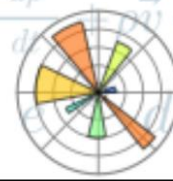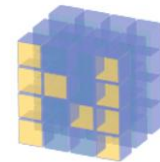
NumPy (Numerical Python)

- Python does not support arrays or matrices and its mathematical support is limited and slow.

- NumPy was created to add support for arrays and matrices to Python.

- Numpy adds support for high-level mathematical functions to operate on these arrays and matrices.

- Python documentation can be found at https://numpy.org/doc/1.17/reference/index.html

- Numpy is limited in scope and is only designed to add numerical computing to phyton.

- NumPy underpins nearly every scientific computing platform created for Python.

```
39 # import numpy as np
40 import numpy as np
41
42 # build an array
43 array = np.ones((5,5))
```

Mapplotlib

- Matplotlib is a plotting library for Python.

- At one point it was formed by combining multiple liberies (pyplot and pylab). We will use pyplot as pylabs use is discouraged.

- Pyplot is a Matplotlib module which provides a MATLAB-like interface.

- matplotlib documentation can be found at https://matplotlib.org/3.1.1/api/index.html

- Pyplot documentation can be found at https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.html#module-matplotlib.pyplot
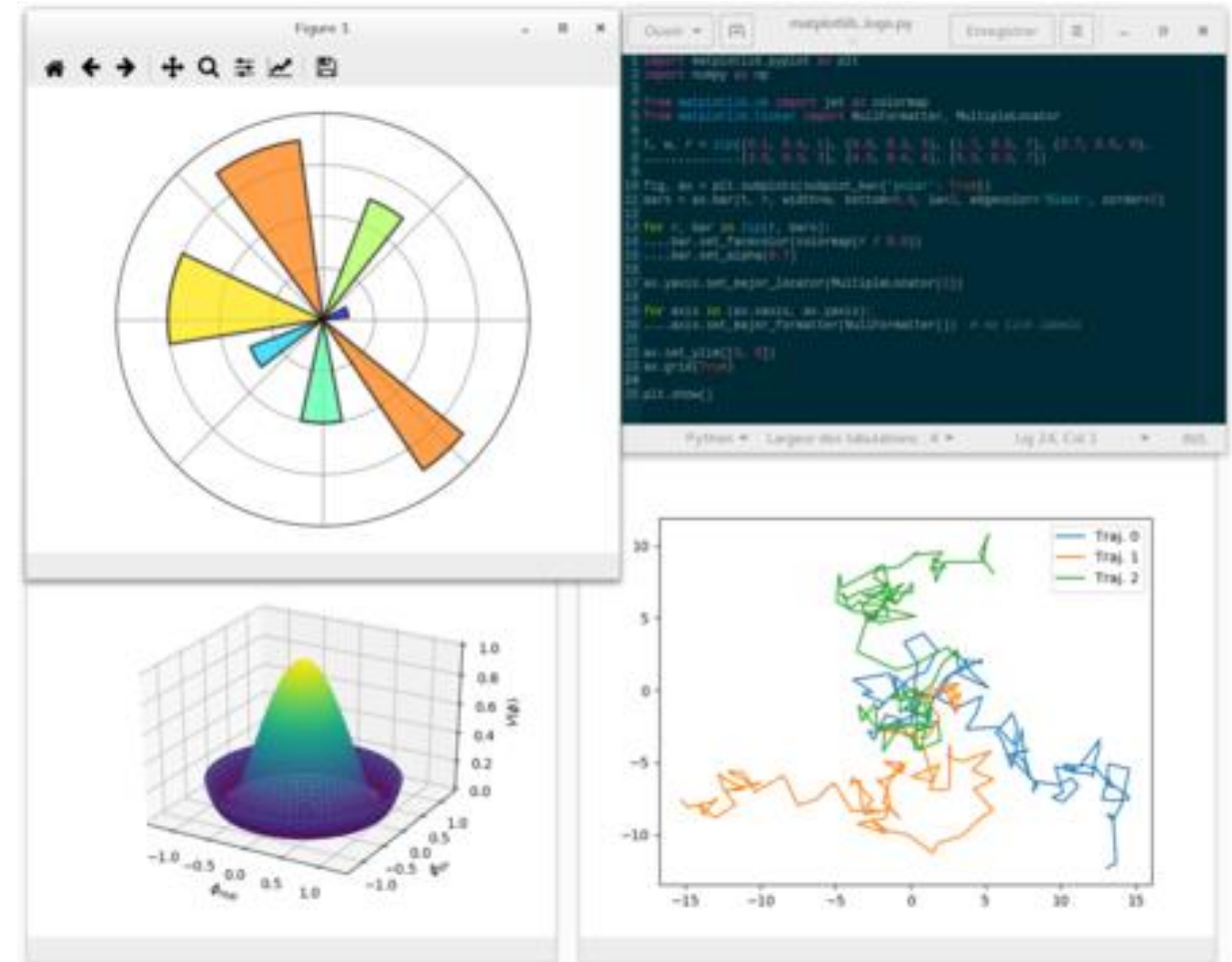
SciPy (not ScientificPython but close enough)

- SciPy is a Python library used for scientific computing and technical computing.

- SciPy builds on the NumPy array class and the capabilities of matplotlib.

- SciPy contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering.

- SciPy and NumPy contain some duplicate capabilities (e.g., FFT). A common way to look as this is that the NumPy version is more bare-bones than the equivalent SciPy version.

- SciPy documentation can be found at https://docs.scipy.org/doc/scipy/reference/

Optimization (scipy.optimize)
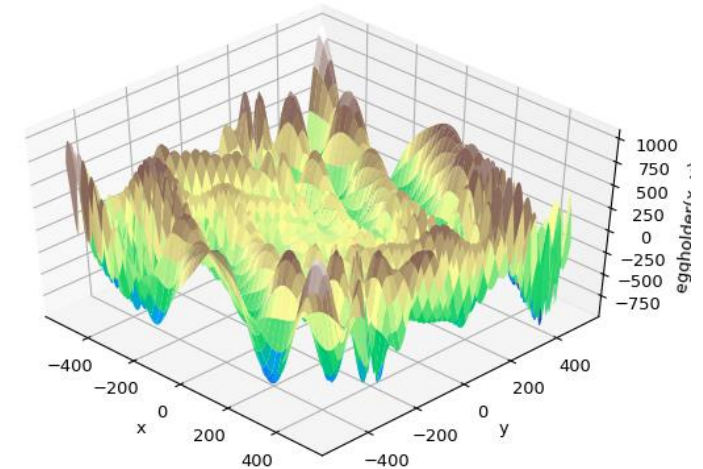
Signal Processing (scipy.signal)
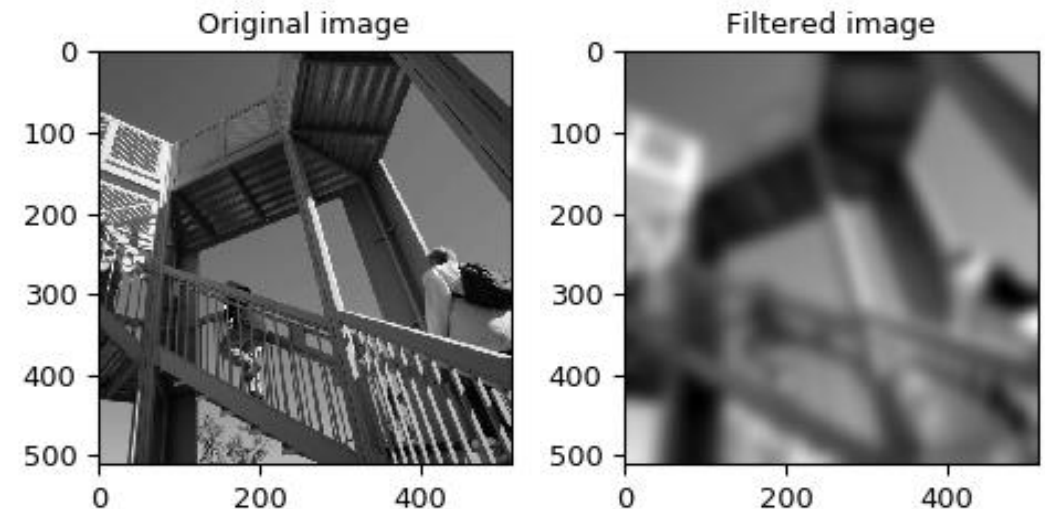
scikit-learn (also called sklearn)

- Sklearn is a machine learning library for Python.
- It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN.
- Sklearn is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.
- Sklearn documentation can be found at
- https://scikit-learn.org/stable/user_guide.html
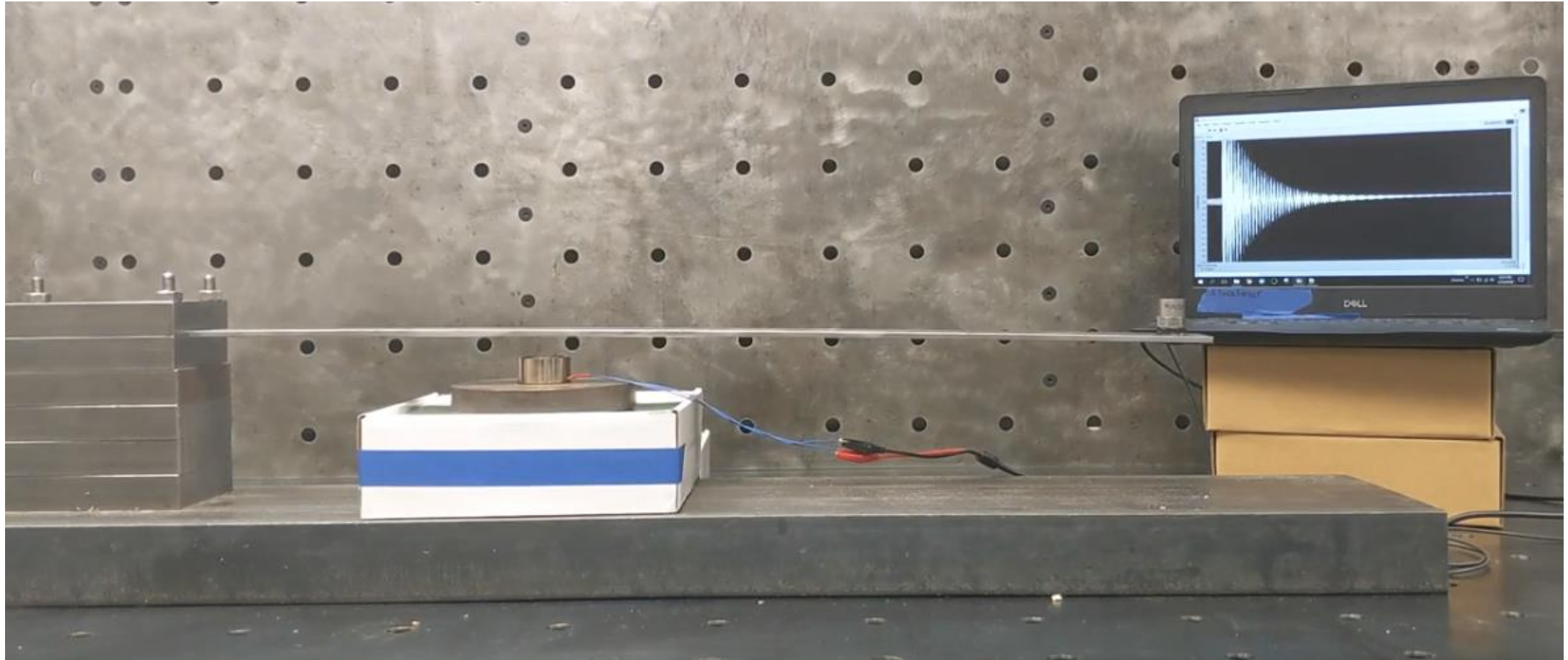
Decision Trees

Clustering
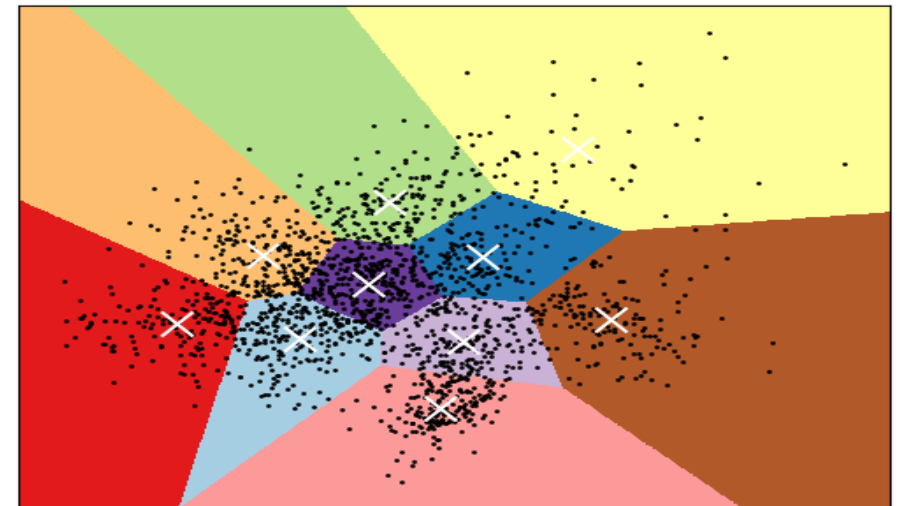
Conclusions:

- Python is great, but not useful for scientific computing.

- Scientific modules have been developed for python to expand its usefulness into scientific computing. The major packages are:
  - NumPy
  - Matplotlib
    - Pyplot
  - SciPy
  - Sklearn

- We code in a Integrated Development Environment (IDE) of which there are lots, we used Spyder in our examples.

```python
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  """
4  Created on Fri Nov 15 16:05:28 2019
5
6  @author: austin
7  """
8
9
10 import IPython as IP
11 IP.get_ipython().magic('reset -sf')
12
13 import numpy as np
14 import scipy as sp
15 import matplotlib as mpl
16 import matplotlib.pyplot as plt
17
18 plt.close('all')
19
20 #%% Load and plot data
21 D = np.loadtxt('vibration_data/Vibration_measurement.csv',delimiter=',')
22
23 tt = D[:,0]
24 dd = D[:,1]
25
26 plt.figure('beam data',figsize=(3.5,3))
27 plt.plot(tt,dd,'-',label='data 1')
28 plt.grid(True)
29 plt.xlabel('time (s)')
30 plt.ylabel('voltage (v)')
31 plt.title('beam data')
32 plt.xlim([-0.1,1.6])
33 plt.legend(framealpha=1,loc=8)
34 plt.tight_layout()
35 plt.savefig('plot.png',dpi=300)
36 array = np.ones((5,5))
37
38 #%% Plot an FFT of the data
39 |
```

Example #3 – Data processing (FFT) in SciPy

THANKS!

Name

Title

Email

Social