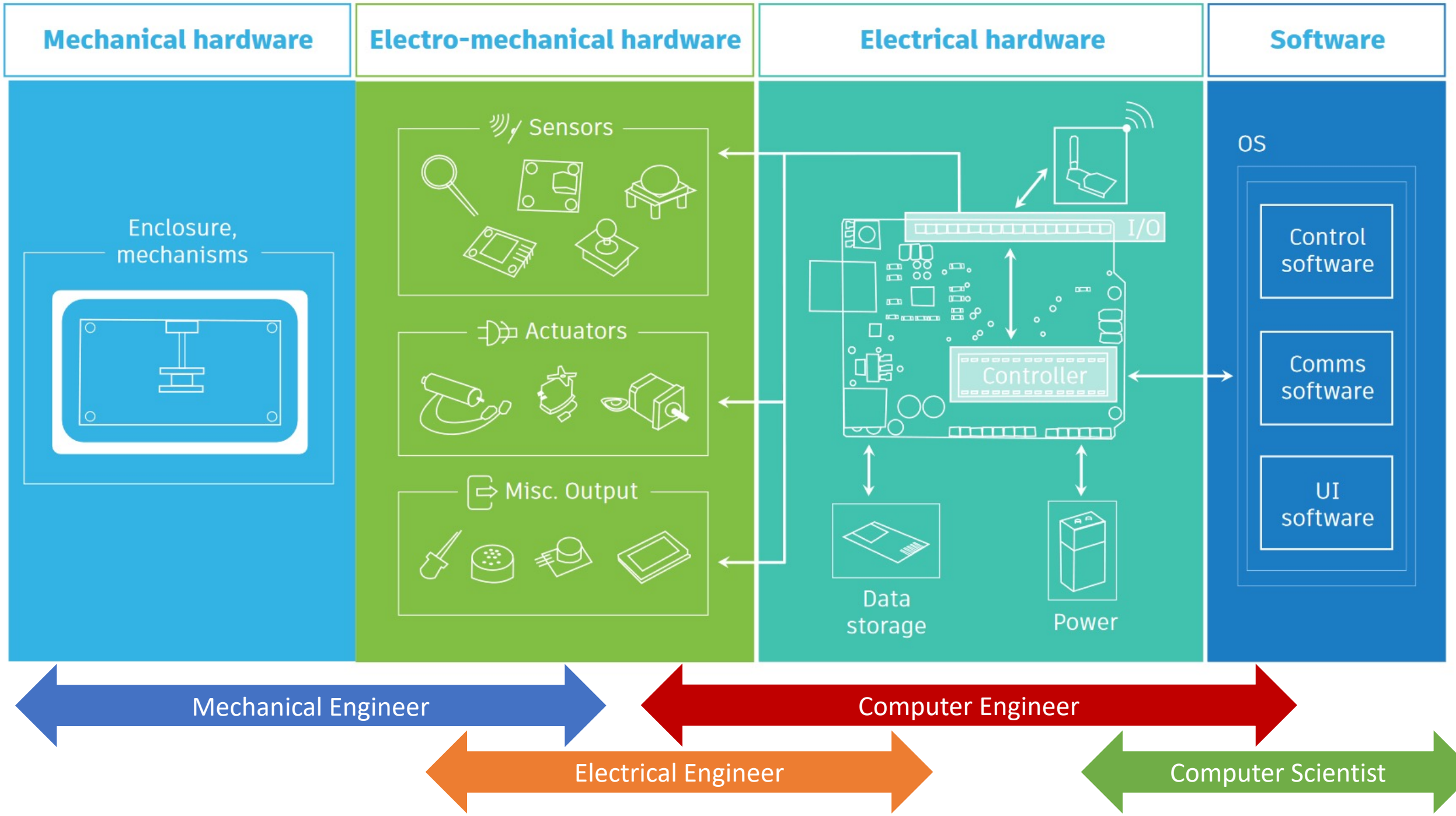
ENGR 298: Engineering Analysis and Decision Making

Dr. Jason Forsyth
Department of Engineering
James Madison University

So, we need to program. Why do engineers care?

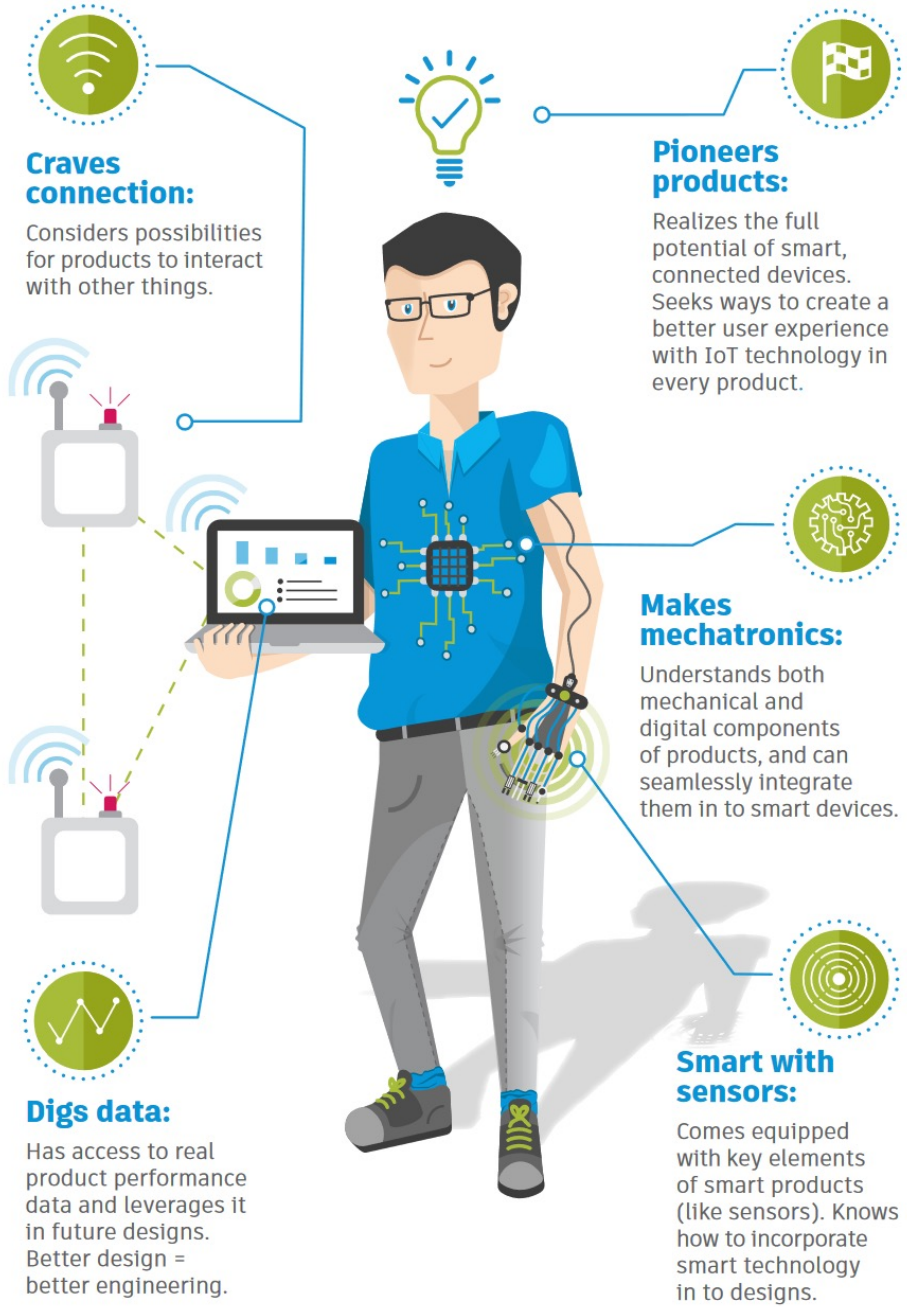

Mechanical hardware
Electro-mechanical hardware
Electrical hardware
Software
Enclosure, mechanisms
Sensors
Actuators
Misc. Output
I/O
Controller
Data storage
Power
OS
Control software
Comms software
UI software
Mechanical Engineer
Computer Engineer
Electrical Engineer
Computer Scientist

INTRODUCTION

The smart product ~~mechanical~~ engineer

IoT is inevitably changing how engineers design products. While the mechanical engineer of the future needs the same foundation of technical skills and savvy for creative problem-solving as always, additional characteristics will soon be necessary.

Be a full stack engineer



So why this course?

Why not visit our friends in CS?

Engineers + Programming + Data

- Focus is on applied programming. We need answers to questions. Gather data, perform analysis, present results, move on.
- Don’t really care if the code is “good”. But it needs to provide accurate results, insights, and be maintainable.
- We could do this with any “data”. Have collected sets from personal experience and various faculty members to exhibit various challenges / analysis techniques across engineering disciplines.
- We’ll learn how to write/share code and use modern software tools.

Some Particular Challenges

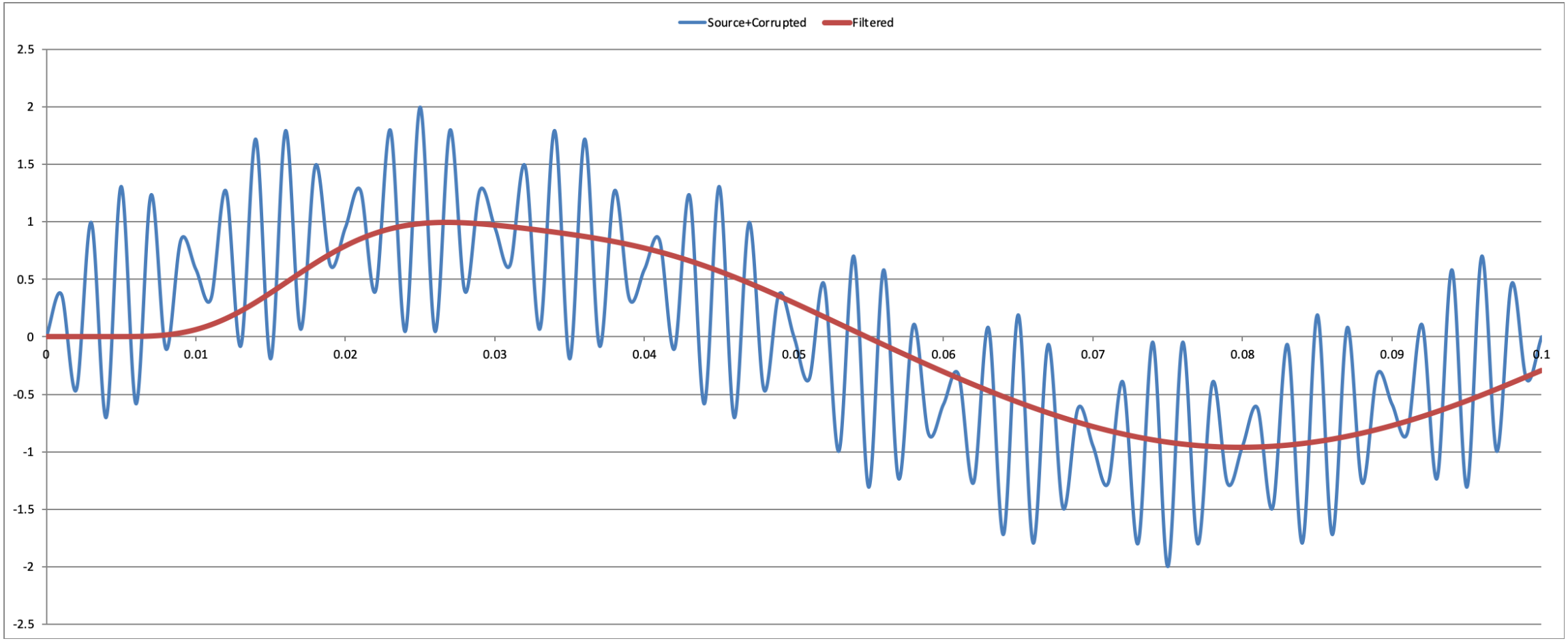
- Data is noisy, incomplete, and at times, mislabeled. How do we handle these situations?

- Your code will be written for a particular domain. There may (or may not) be templates, patterns, or references to follow. You may need to develop it new for each situation.

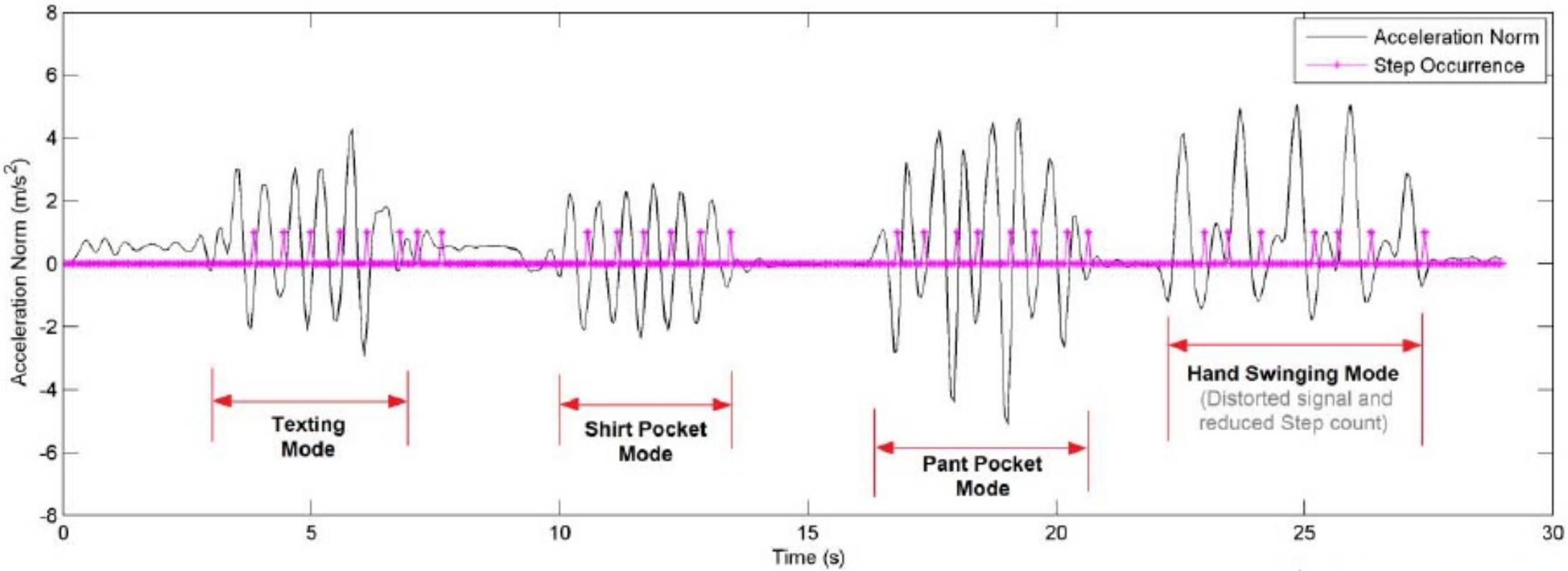
- Every domain will format, encode, and represent information differently. May have to “reboot” each time a new problem is tackled but can pull from experience.

Data Examples

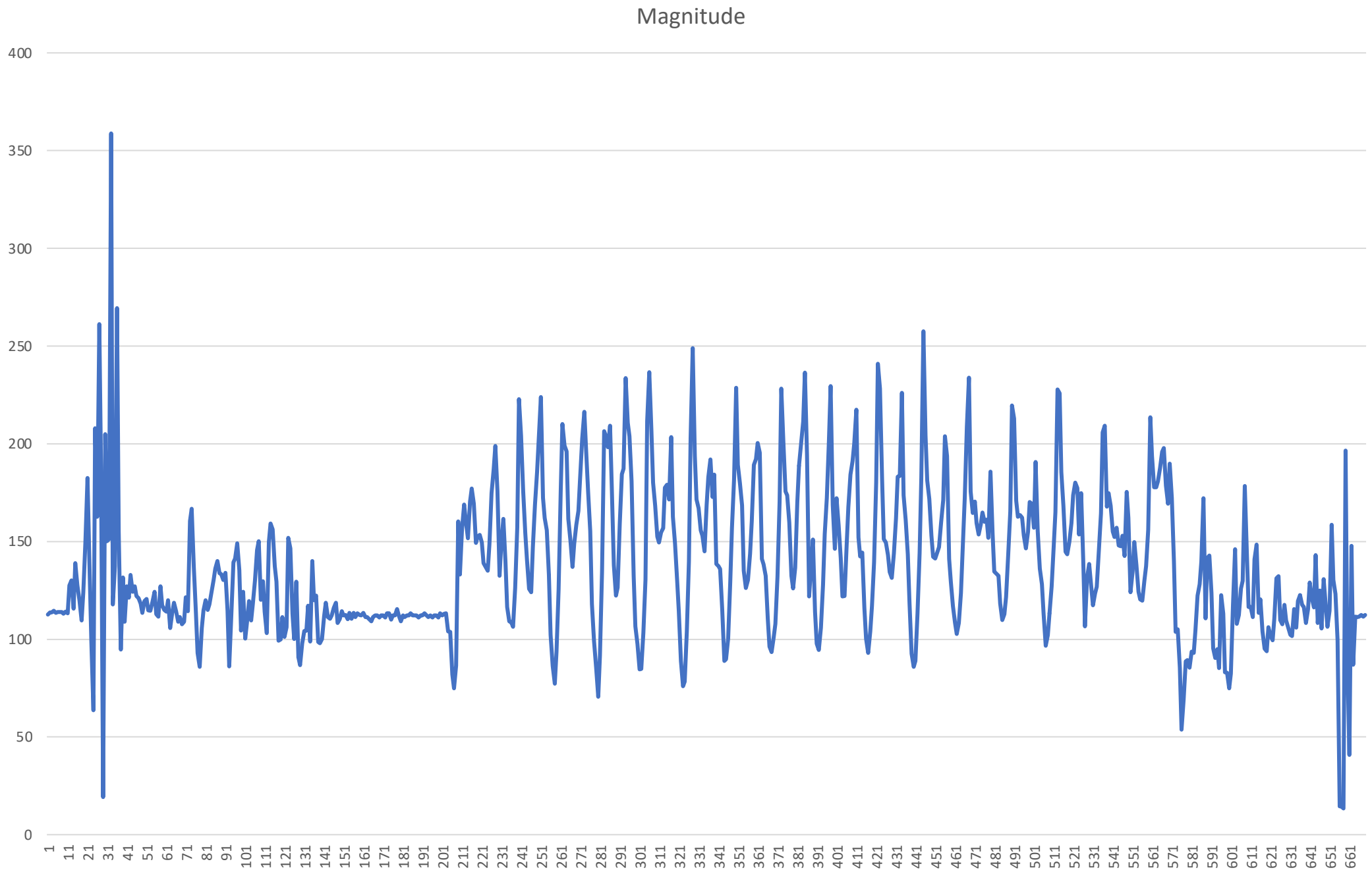
Time Series Data - Biometric

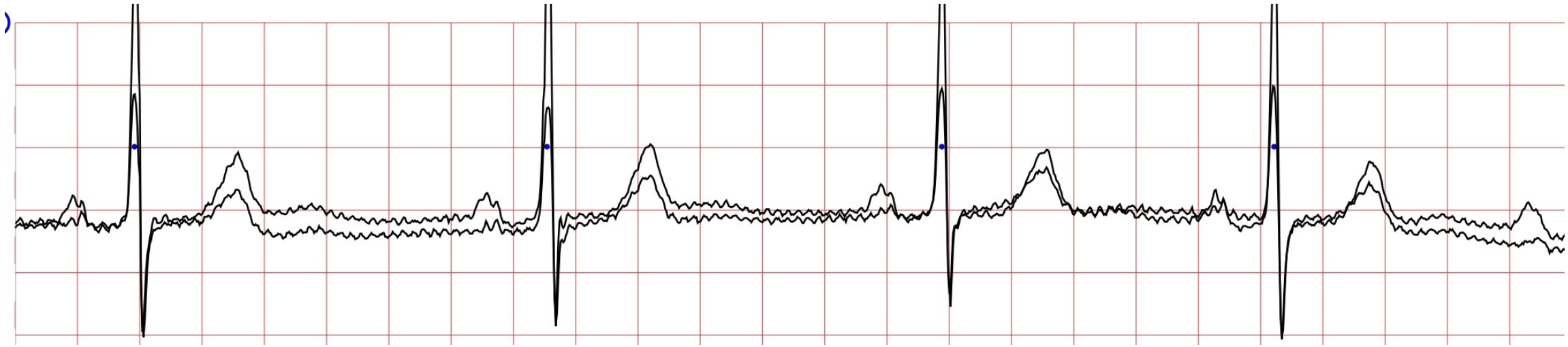

Source+Corrupted
Filtered
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
0
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.1

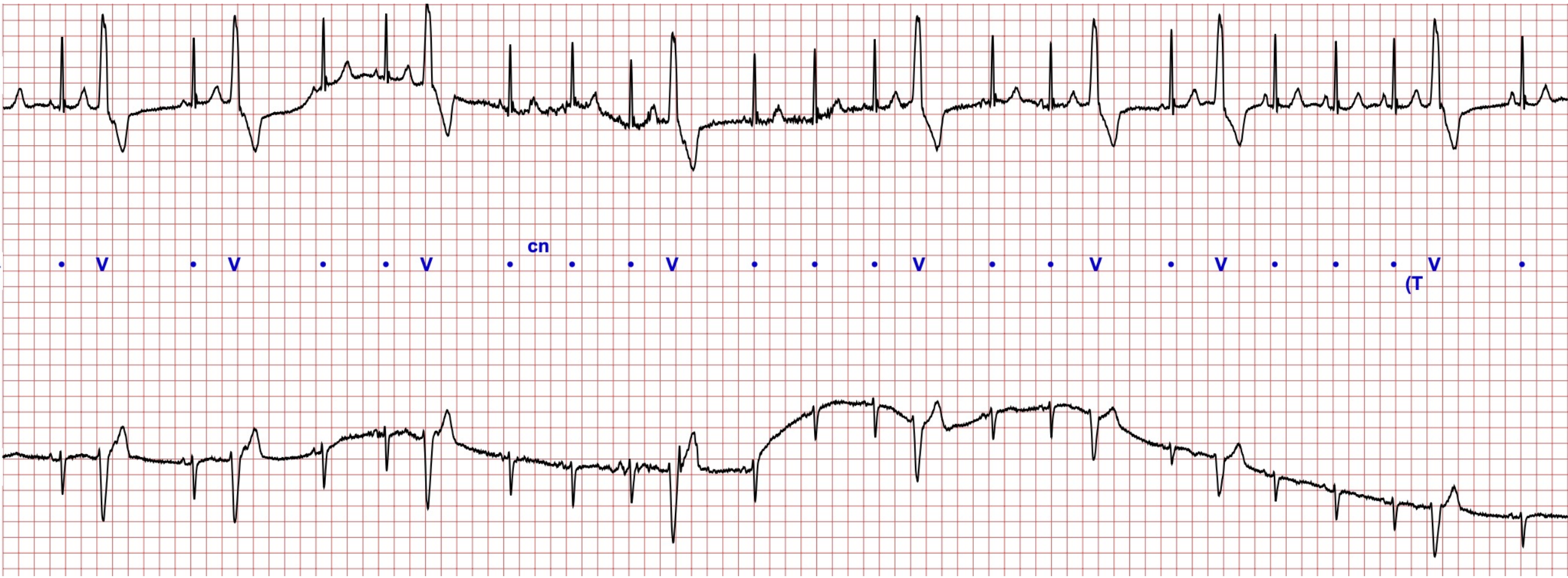
Anatomy of a Step – Acceleration



$$G_{NORM} = \sqrt{G_x^2 + G_y^2 + G_z^2}$$


Magnitude
400
350
300
250
200
150
100
50
0
1
11
21
31
41
51
61
71
81
91
101
111
121
131
141
151
161
171
181
191
201
211
221
231
241
251
261
271
281
291
301
311
321
331
341
351
361
371
381
391
401
411
421
431
441
451
461
471
481
491
501
511
521
531
541
551
561
571
581
591
601
611
621
631
641
651
661




V V V V V V V V V V V
cn
(T

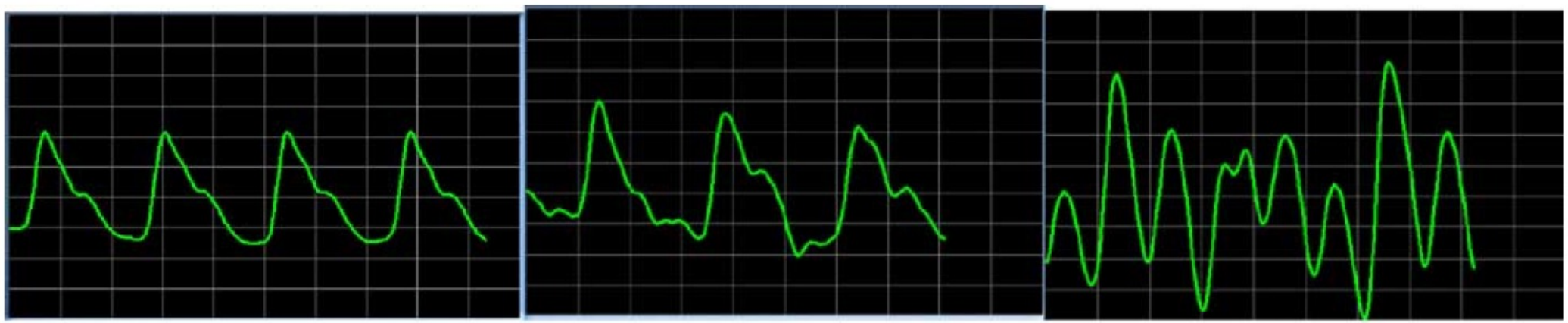


Fig. 1. Pulse oximeter output showing volumetric changes in blood over several heart beats. (a) Typical signal with no errors present. (b) Signal with errors when sensor location moves from side to side and (c) up and down.

Other Datasets Available

- **Dr. Miller**: protein identification and forces within cellular images
- **Dr. Prins**: tensile strength material testing and automatic failure calculation
- **Dr. Castaneda**: modeling saturation and humidity in concrete.
- **Dr. Forsyth**: motion artifacts from wearable PPG
- **Virginia Clean Cities**: determining EV charger placement in Virginia; analyzing biofuel sales; mapping truck stops with EV chargers.

How is this course structured…

- Course meets 3x each week. Anticipate that Monday and ~Wednesday will be lecture/content delivery. Remaining time is for you to achieve weekly/module deliverables.
- Would like to incorporate peer instruction; if you have completed the weekly assignments then you become a TA to help your colleagues. (Will develop some incentive structure here and rules).
- Each large “module” will end with a small culminating project that will occur outside of class. Final report/demonstration will show mastery of an independent project.
- Grading distribution subject to change; will not adjust to your “disadvantage”.

Homework / In-class Assignments (10x)	20%
Culminating Assignment Per Module (6x)	60%
Final Project Report and Source Code	10%
Final Project Presentation/Demo	10%

Modules with culminating deliverables…

1. **Introduction to Python (2 - 3 weeks)**
    a. Description
        i. An opening introduction to Python that ensures the student can load the development environment, write basic programs, manage packages, test their programs against automated tools, and create appropriate documentation.
    b. Learning Objectives
        i. Describe fundamental programming concepts such as variables, objects, methods, and classes in Python.
        ii. Use automated software tools to test programs for correctness and functionality.
        iii. Provide appropriate documentation and appropriation in the use of open-source software.
    c. Culminating Assignment
        i. Create your own PyCharm installation, install Pandas, and write code that passes tests in autograder. Publish code to GitHub.

Final Presentation / Demo

Task 1 - Examining Data of Interest (may be done in parallel with Task 2)

Browse the websites listed below to find a dataset of interest that you will use for this project. The particular data and challenge you select is open, however, consider the following items: (1) the data should be sharable and publicly available for access by the instructors and TAs. (2) it should be provided in a common format (.txt, .csv..etc.) that can easily be read by MATLAB and Excel. (3) the file size should be manageable by either program (a few MegaBytes rather than GigaBytes or larger...).

The following database sites should be used for this task:

- PhysioNet: https://physionet.org/
- Data Gov: https://www.data.gov/
- UCI Machine Learning Repository: https://archive.ics.uci.edu/ml/index.php

As 1-2 slides in the SlideDoc, identify the dataset of interest including a URL for download. Describe how the data was collected, why the data was collected, and what information in the dataset might be useful for your investigations.

Task 2 - Developing Questions of Interest (may be done in parallel with Task 1)

Develop 5 - 10 questions that will drive your investigation through this deliverable. By the end of the assignment, these will be narrowed down to 2-3 questions. These questions must be insightful, open-ended questions, that provide an *inference* or *conjecture* about some underlying dataset or problem. The questions should not be answerable by a simple yes/no response. Recalling our examples from the Week 7 and 8 lectures, some inferential questions on Housing data might be "How did regions of the country that responded well to the 2008 crisis react to the 2020 pandemic?" For the Birth Rate data, a powerful question might be "How does the change in birth rate within the US compare with other countries of similar economic development? What about countries with different economic development?"

The questions developed do not need to be as "deep" as these but they should: (1) ask about fundamental relationships between different "variables" within the data set; (2) be sufficiently interesting to carry you through the assignment, and (3) be "answerable" or "calculatable" within your abilities and the context of the course. We, as of yet, do not have an algebraic function to calculate "happiness".

In the slide doc, place each question in its own slide. On each slide describe: why this question is interesting to you, why it is an inferential question, and what might be the variables of interest that are being compared in the question.


python™

https://docs.python.org/3/faq/general.html#id20

So Why Python? Why Not C/C++, Java, GO, MATLAB….

It’s free, portable, and popular.

IEEE SPECTRUM

Rank	Language	Type	Score
1	Python	web, desktop, embedded	100.0
2	Java	web, mobile, desktop	95.3
3	C	mobile, desktop, embedded	94.6
4	C++	mobile, desktop, embedded	87.0
5	JavaScript	web	79.5
6	R	desktop	78.6
7	Arduino	embedded	73.2
8	Go	web, desktop	73.1
9	Swift	mobile, desktop	70.5
10	Matlab	desktop	68.4

Summary

Quick Facts: Software Developers	
2018 Median Pay	$105,590 per year $50.77 per hour
Typical Entry-Level Education	Bachelor's degree
Work Experience in a Related Occupation	None
On-the-job Training	None
Number of Jobs, 2018	1,365,500
Job Outlook, 2018-28	21% (Much faster than average)
Employment Change, 2018-28	284,100

Resources

Language Reference and Tutorials

Learn Python Tutorials

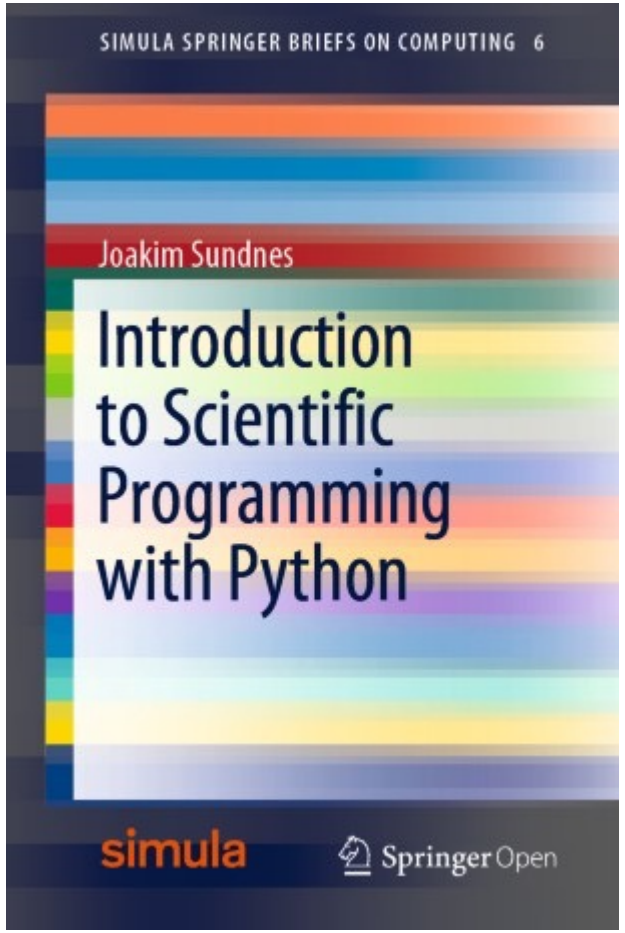
- Solo Learn (https://www.sololearn.com/learning/1169)
- Code Academy: Python 3 (https://www.codecademy.com/learn/learn-python-3)
- Code Academy: Data Analysis in Python (https://www.codecademy.com/learn/paths/analyze-data-with-python)
- Udacity (https://www.udacity.com/course/introduction-to-python--ud1110)
- Official Python Tutorial (https://docs.python.org/3/tutorial/)

Electronic Textbooks (Python)

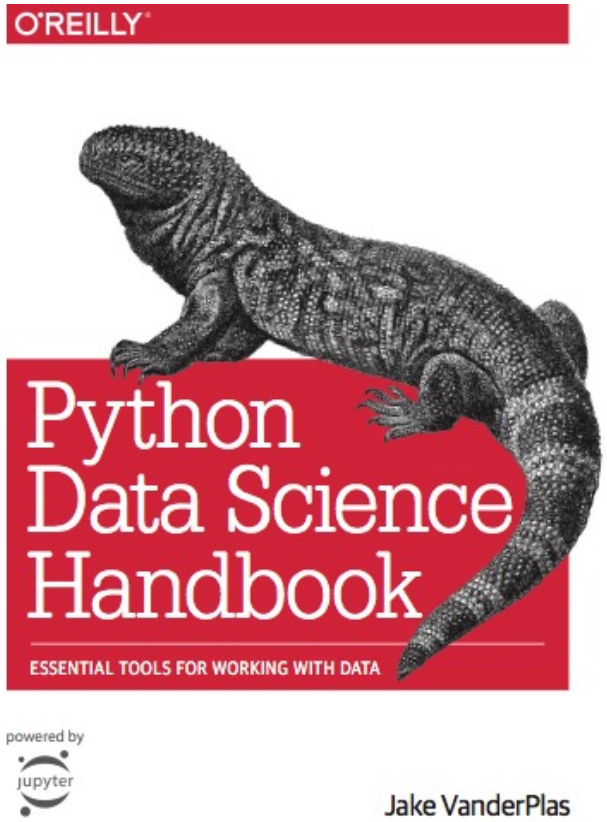
- An Introduction to Scientific Computing in Python (https://link.springer.com/book/10.1007/978-3-030-50356-7)
  - Companion GitHub repo (https://github.com/sundnes/python_intro)
  - Course PDF (link)
- A Primer on Scientific Programming with Python (https://link.springer.com/book/10.1007/978-3-662-49887-3)
- Python for Data Analysis (3rd edition) (https://www.oreilly.com/library/view/python-for-data/9781491957653/)
  - Available via JMU Libraries (link)
- Python Data Science Handbook (https://jakevdp.github.io/PythonDataScienceHandbook/)

I'm not the language/package expert but I will guide you to expert resources

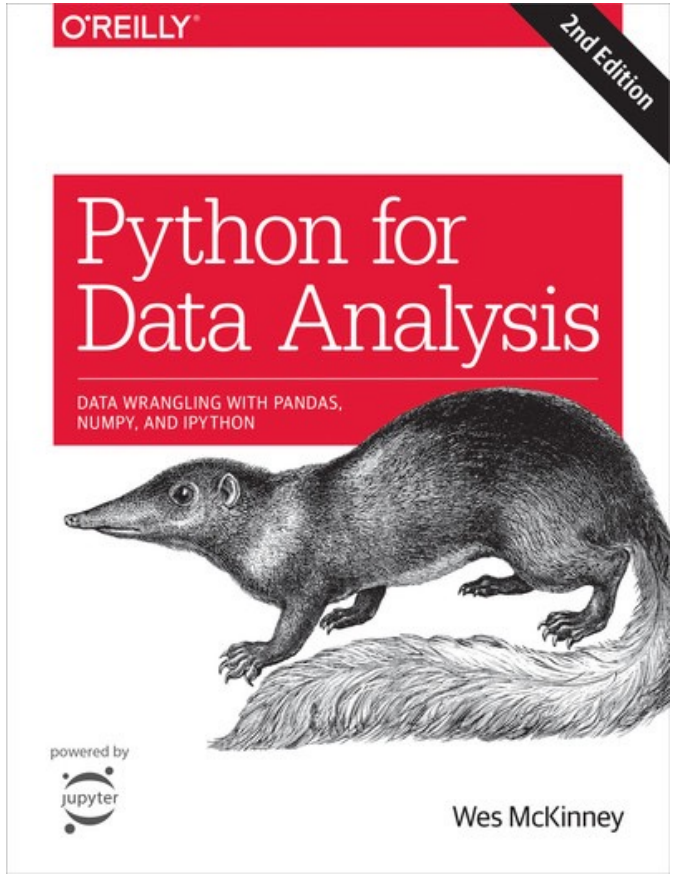
Goto References / Guides



On Canvas as PDF (text for CS 145)



Link on Canvas



Online Via JMU Library

Quick Handouts

datacamp

Python For Data Science NumPy Cheat Sheet

Learn NumPy online at www.DataCamp.com

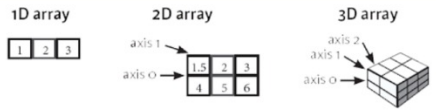
Numpy

The NumPy library is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays

Use the following import convention:

```
>>> import numpy as np
```

NumPy Arrays



> Creating Arrays

```
>>> a = np.array([1,2,3])
>>> b = np.array([(1.5,2,3), (4,5,6)], dtype = float)
>>> c = np.array([[(1.5,2,3), (4,5,6)],[(3,2,1), (4,5,6)]], dtype = float)
```

Initial Placeholders

```
>>> np.zeros((3,4)) #Create an array of zeros
>>> np.ones((2,3,4),dtype=np.int16) #Create an array of ones
>>> d = np.arange(10,25,5) #Create an array of evenly spaced values (step value)
>>> np.linspace(0,2,9) #Create an array of evenly spaced values (number of samples)
>>> e = np.full((2,2),7) #Create a constant array
>>> f = np.eye(2) #Create a 2X2 identity matrix
>>> np.random.random((2,2)) #Create an array with random values
>>> np.empty((3,2)) #Create an empty array
```

> I/O

Saving & Loading On Disk

```
>>> np.save('my_array', a)
>>> np.savez('array.npz', a, b)
>>> np.load('my_array.npy')
```

Saving & Loading Text Files

```
>>> np.loadtxt("myfile.txt")
>>> np.genfromtxt("my_file.csv", delimiter=',')
>>> np.savetxt("myarray.txt", a, delimiter=" ")
```

> Asking For Help

```
>>> np.info(np.ndarray.dtype)
```

> Inspecting Your Array

```
>>> a.shape #Array dimensions
>>> len(a) #Length of array
>>> b.ndim #Number of array dimensions
>>> e.size #Number of array elements
>>> b.dtype #Data type of array elements
>>> b.dtype.name #Name of data type
>>> b.astype(int) #Convert an array to a different type
```

> Data Types

```
>>> np.int64 #Signed 64-bit integer types
>>> np.float32 #Standard double-precision floating point
>>> np.complex #Complex numbers represented by 128 floats
>>> np.bool #Boolean type storing TRUE and FALSE values
>>> np.object #Python object type
>>> np.string_ #Fixed-length string type
>>> np.unicode_ #Fixed-length unicode type
```

> Array Mathematics

Arithmetic Operations

```
>>> g = a - b #Subtraction
 array([[-0.5, 0. , 0. ],
        [-3. , -3. , -3. ]])
>>> np.subtract(a,b) #Subtraction
>>> b + a #Addition
 array([[ 2.5, 4. , 6. ],
        [ 5. , 7. , 9. ]])
>>> np.add(b,a) #Addition
>>> a / b #Division
 array([[ 0.666666667, 1. , 1. ],
        [ 0.25 , 0.4 , 0.5 ]])
>>> np.divide(a,b) #Division
>>> a * b #Multiplication
 array([[ 1.5, 4. , 9. ],
        [ 4. , 10. , 18. ]])
>>> np.multiply(a,b) #Multiplication
>>> np.exp(b) #Exponentiation
>>> np.sqrt(b) #Square root
>>> np.sin(a) #Print sines of an array
>>> np.cos(b) #Element-wise cosine
>>> np.log(a) #Element-wise natural logarithm
>>> e.dot(f) #Dot product
 array([[ 7., 7.],
        [ 7., 7.]])
```

Comparison

```
>>> a == b #Element-wise comparison
 array([[False, True, True],
        [False, False, False]], dtype=bool)
>>> a < 2 #Element-wise comparison
 array([True, False, False], dtype=bool)
>>> np.array_equal(a, b) #Array-wise comparison
```

Aggregate Functions

```
>>> a.sum() #Array-wise sum
>>> a.min() #Array-wise minimum value
>>> b.max(axis=0) #Maximum value of an array row
>>> b.cumsum(axis=1) #Cumulative sum of the elements
>>> a.mean() #Mean
>>> np.median(b) #Median
>>> np.corrcoef(a) #Correlation coefficient
>>> np.std(b) #Standard deviation
```

> Copying Arrays

```
>>> h = a.view() #Create a view of the array with the same data
>>> np.copy(a) #Create a copy of the array
>>> h = a.copy() #Create a deep copy of the array
```

> Sorting Arrays

```
>>> a.sort() #Sort an array
>>> c.sort(axis=0) #Sort the elements of an array's axis
```

> Subsetting, Slicing, Indexing

Subsetting

```
>>> a[2] #Select the element at the 2nd index
 3
>>> b[1,2] #Select the element at row 1 column 2 (equivalent to b[1][2])
 6.0
```

Slicing

```
>>> a[0:2] #Select items at index 0 and 1
 array([1, 2])
>>> b[0:2,1] #Select items at rows 0 and 1 in column 1
 array([ 2., 5.])
>>> b[:1] #Select all items at row 0 (equivalent to b[0:1, :])
 array([[1.5, 2., 3.]])
>>> c[1,...] #Same as [1,:,:]
 array([[[ 3., 2., 1.],
         [ 4., 5., 6.]]])
>>> a[ : :-1] #Reversed array a array([3, 2, 1])
```

Boolean Indexing

```
>>> a[a<2] #Select elements from a less than 2
 array([1])
```

Fancy Indexing

```
>>> b[[1, 0, 1, 0],[0, 1, 2, 0]] #Select elements (1,0),(0,1),(1,2) and (0,0)
 array([ 4. , 2. , 6. , 1.5])
>>> b[[1, 0, 1, 0]][:,[0,1,2,0]] #Select a subset of the matrix's rows and columns
 array([[ 4. ,5. , 6. , 4. ],
        [ 1.5, 2. , 3. , 1.5],
        [ 4. , 5. , 6. , 4. ],
        [ 1.5, 2. , 3. , 1.5]])
```

> Array Manipulation

Transposing Array

```
>>> i = np.transpose(b) #Permute array dimensions
>>> i.T #Permute array dimensions
```

Changing Array Shape

```
>>> b.ravel() #Flatten the array
>>> g.reshape(3,-2) #Reshape, but don't change data
```

Adding/Removing Elements

```
>>> h.resize((2,6)) #Return a new array with shape (2,6)
>>> np.append(h,g) #Append items to an array
>>> np.insert(a, 1, 5) #Insert items in an array
>>> np.delete(a,[1]) #Delete items from an array
```

Combining Arrays

```
>>> np.concatenate((a,d),axis=0) #Concatenate arrays
 array([ 1, 2, 3, 10, 15, 20])
>>> np.vstack((a,b)) #Stack arrays vertically (row-wise)
 array([[ 1. , 2. , 3. ],
        [ 1.5, 2. , 3. ],
        [ 4. , 5. , 6. ]])
>>> np.r_[e,f] #Stack arrays vertically (row-wise)
>>> np.hstack((e,f)) #Stack arrays horizontally (column-wise)
 array([[ 7., 7., 1., 0.],
        [ 7., 7., 0., 1.]])
>>> np.column_stack((a,d)) #Create stacked column-wise arrays
 array([[ 1, 10],
        [ 2, 15],
        [ 3, 20]])
>>> np.c_[a,d] #Create stacked column-wise arrays
```

Splitting Arrays

```
>>> np.hsplit(a,3) #Split the array horizontally at the 3rd index
 [array([1]),array([2]),array([3])]
>>> np.vsplit(c,2) #Split the array vertically at the 2nd index
 [array([[[ 1.5, 2. , 1. ],
          [ 4. , 5. , 6. ]]]),
 array([[[ 3., 2., 3.],
         [ 4., 5., 6.]]])]
```

Learn Data Skills Online at www.DataCamp.com

Special Course Rules

Collaboration and Online Resources Policy: The focus of this semester will be to complete individual milestones and a final project. Given the complexity of these tassk, students may seek one another for: (1) general advice debugging and troubleshooting their solutions and (2) receiving feedback on the merits of their current design or approach. However, any assistance or contributions received from another student must be duly noted in any submission or milestone. Failure to note another student's contribution will be considered plagiarism and constitute a violation of the Academic Integrity Policy. Furthermore, under no circumstances will any student transmit or provide any software, circuit, diagram, or other electronic resource to another student where that is not already publicly available.

Students may also utilize any online resource (diagrams, software, tools…etc.) that is publicly available. However, students must clearly identify what elements of their project utilize online resources. These external sources must be clearly annotated in any code, diagram, or other artifact generated in this course. Failure to do so will be considered plagiarism and constitute a violation of the Academic Integrity Policy. Furthermore, students are liable and responsible for the effects of any external software. Failure to meet deadlines due to code "found online" that "doesn't work" is not an acceptable solution. Additionally, students will be liable for any damage to course equipment and supplies due to any external code.

Final Thoughts

- I view my role as guide, facilitator, evaluator, tech support.. I will setup the experience for you to be successful. Less 313, more 498.

- To do that, you should put in the work. Grab resources. Read. Try things out. Don’t just complete the problem, understand why/how things work.

- This course exists because students asked for it. If we’re successful it may become a permanent fixture in the department. Be the change you want to see.

Questions?

Getting Started in the Course

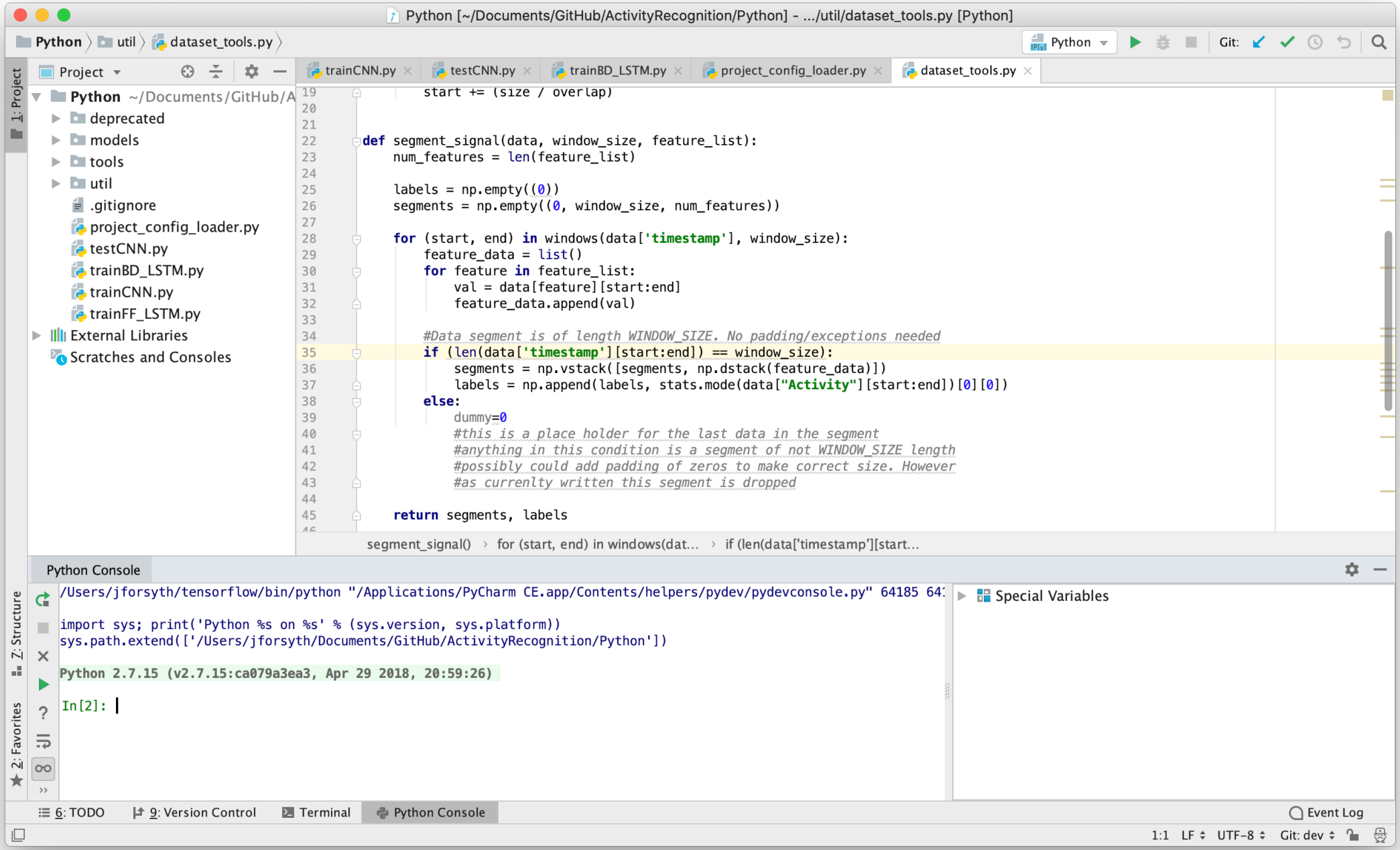
There are two versions of Python…?? It's all about the strings.

Why was Python 3 made incompatible with Python 2?

According to Guido, he initiated the Python 3 project to clean up a variety of issues with Python 2 where he didn't feel comfortable with fixing them through the normal deprecation process. This included the removal of classic classes, changing integer division to automatically promote to a floating point result (retaining the separate floor division operation) and changing the core string type to be based on Unicode by default. With a compatibility break taking place anyway, the case was made to just include some other changes in that process (like converting print to a function), rather than going through the full deprecation process within the Python 2 series.

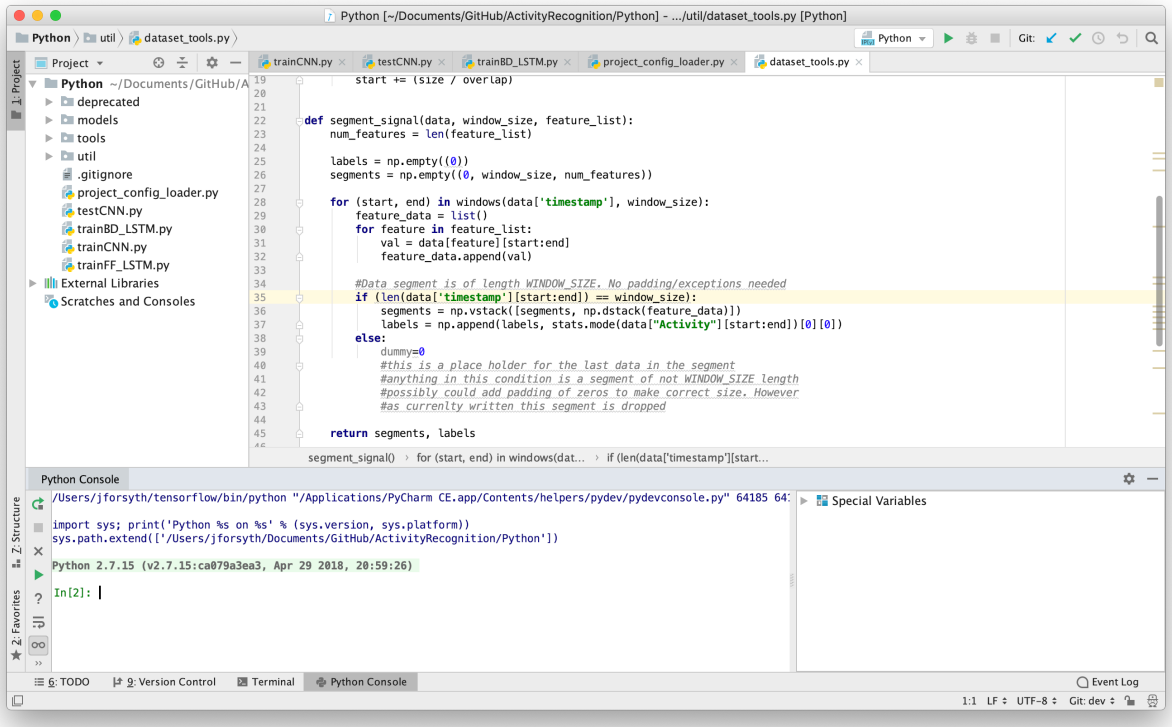
http://python-notes.curiousefficiency.org/en/latest/python3/questions_and_answers.html

PyCharm: Your Friendly Neighborhood IDE

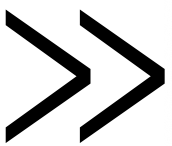

Python [~/Documents/GitHub/ActivityRecognition/Python] - .../util/dataset_tools.py [Python]
Python
util
dataset_tools.py
Git:
Project
Python ~/Documents/GitHub/A
deprecated
models
tools
util
.gitignore
project_config_loader.py
testCNN.py
trainBD_LSTM.py
trainCNN.py
trainFF_LSTM.py
External Libraries
Scratches and Consoles
trainCNN.py
testCNN.py
trainBD_LSTM.py
project_config_loader.py
dataset_tools.py
start += (size / overlap)
def segment_signal(data, window_size, feature_list):
num_features = len(feature_list)
labels = np.empty((0))
segments = np.empty((0, window_size, num_features))
for (start, end) in windows(data['timestamp'], window_size):
feature_data = list()
for feature in feature_list:
val = data[feature][start:end]
feature_data.append(val)
#Data segment is of length WINDOW_SIZE. No padding/exceptions needed
if (len(data['timestamp'][start:end]) == window_size):
segments = np.vstack([segments, np.dstack(feature_data)])
labels = np.append(labels, stats.mode(data["Activity"][start:end])[0][0])
else:
dummy=0
#this is a place holder for the last data in the segment
#anything in this condition is a segment of not WINDOW_SIZE length
#possibly could add padding of zeros to make correct size. However
#as currenlty written this segment is dropped
return segments, labels
segment_signal() › for (start, end) in windows(dat... › if (len(data['timestamp'][start...
Python Console
/Users/jforsyth/tensorflow/bin/python "/Applications/PyCharm CE.app/Contents/helpers/pydev/pydevconsole.py" 64185 64
import sys; print('Python %s on %s' % (sys.version, sys.platform))
sys.path.extend(['/Users/jforsyth/Documents/GitHub/ActivityRecognition/Python'])
Python 2.7.15 (v2.7.15:ca079a3ea3, Apr 29 2018, 20:59:26)
In[2]:
Special Variables
1: Project
Z: Structure
2: Favorites
6: TODO
9: Version Control
Terminal
Python Console
Event Log
1:1
LF
UTF-8
Git: dev

Welcome to PyCharm

- **PyCharm** is an *integrated development environment* (IDE). It contains all your software engineering tools in one place.
- **Code editor** with style checking, refractoring, git integration…etc.
- **Virtual environment**: a “virtual” installation of Python to manage various package installations.
- **Debug facilities** to step through code.



PyCharm



Idle

Some First Deliverables and Assignments

Diagnostic Assignments

- Available on the course GitHub. Download them, complete in Pycharm, and submit in Gradescope.

- Install PyCharm, Python 3, and complete these tasks:
  - Lists.py: can you handle lists
  - Stats.py: can you do math in Python
  - Calculator: how well do you handle horrible class objects

- I think everyone can handle them. If not, let me know. That's the point of the Diagnostics.