SIGNAL PROCESSING & ML IN UBICOMP

CSE 590 Ubiquitous Computing | Lecture 3 | April 12

Jon Froehlich • Liang He (TA)

DESIGN USE BUILD

MAKEABILITY LAB

PAUL G. ALLEN SCHOOL OF COMPUTER SCIENCE & ENGINEERING

UNIVERSITY of WASHINGTON

SCHEDULE TODAY: 6:30-9:20

06:35-06:55: Carlos Burkle’s disc. of the Bao *et al.*, 2004 activity rec reading

06:55-07:05: Joe Wandyez’s disc. of the Dourish, 2004 context reading (link)

07:05-07:55: SP & ML in ubicomp systems (w/Jupyter Notebook exercises)

07:55-08:05: Break (begin graded demos with Liang)

08:05-08:35: A1: Step Tracker Presentations

08:35-08:45: Introduce and Discuss A2: Gesture Recognition assignment

08:45-09:20: Work on A2: Gesture Recognition in Jupyter Notebook

SCHEDULE TODAY: 6:30-9:20

06:35-06:55: Carlos Burkle's disc. of the Bao *et al.*, 2004 activity rec reading

06:55-07:05: Joe Wandyez's disc. of the Dourish, 2004 context reading (link)

07:05-07:55: SP & ML in ubicomp systems (w/Jupyter Notebook exercises)

07:55-08:05: Break (begin graded demos with Liang)

08:05-08:35: A1: Step Tracker Presentations

08:35-08:45: Introduce and Discuss A2: Gesture Recognition assignment

08:45-09:20: Work on A2: Gesture Recognition in Jupyter Notebook

TODAY'S LEARNING GOALS

Note: We will likely be moving pretty fast today. But all of your learning will be reinforced by the assignment!

Some basics of **Python** and **Jupyter Notebook**

How do we **analyze**, **process**, & **visualize signals** in Jupyter Notebook?

What are **FFTs** and why are they useful?

What are **shape-matching** algorithms?

How should we evaluate the **performance of our classifiers**?

*Much of today's lecture slides are courtesy of Professor Keogh's tutorials

WHAT MIGHT ML HELP US DO IN UBICOMP SYSTEMS?

Classify low- or high-level activities (*e.g.*, gestures, playing baseball)

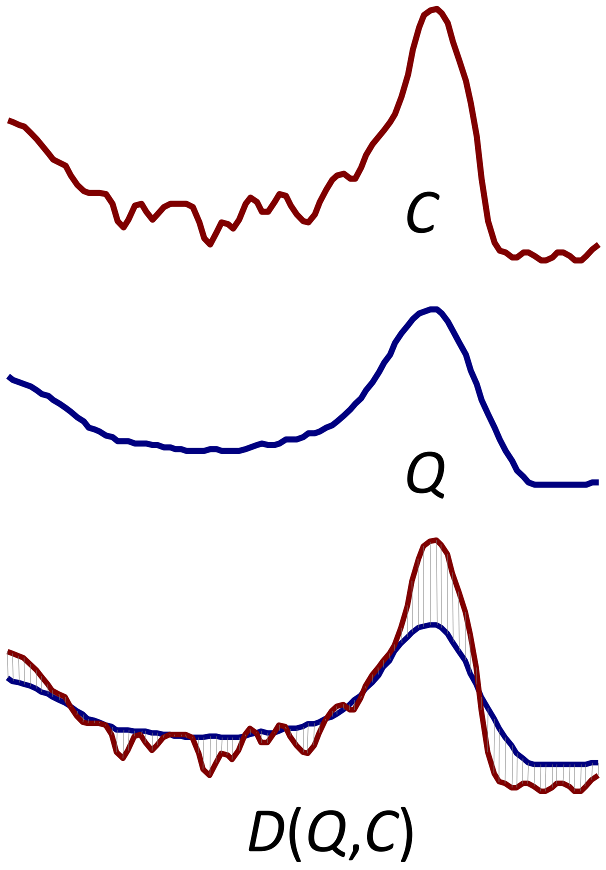
Cluster similar signals together (*e.g.*, how many categories of things exist in this dataset)

Search. (*e.g.*, I have signal A & I want to find all other signals like this in my data)

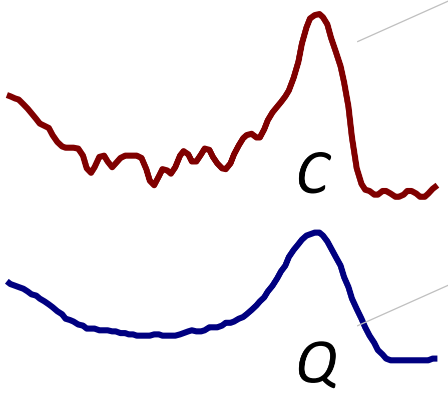
Novelty detection (*i.e.*, anomaly detection)

TWO PREVAILING APPROACHES

SHAPE-MATCHING



FEATURE-BASED APPROACH



Raw Signal	1st Derivative	FFT
Mean	Mean	Top N freqs
Variance	Variance	Fundamental
Skewness	Skewness	
Kurtosis	Kurtosis	

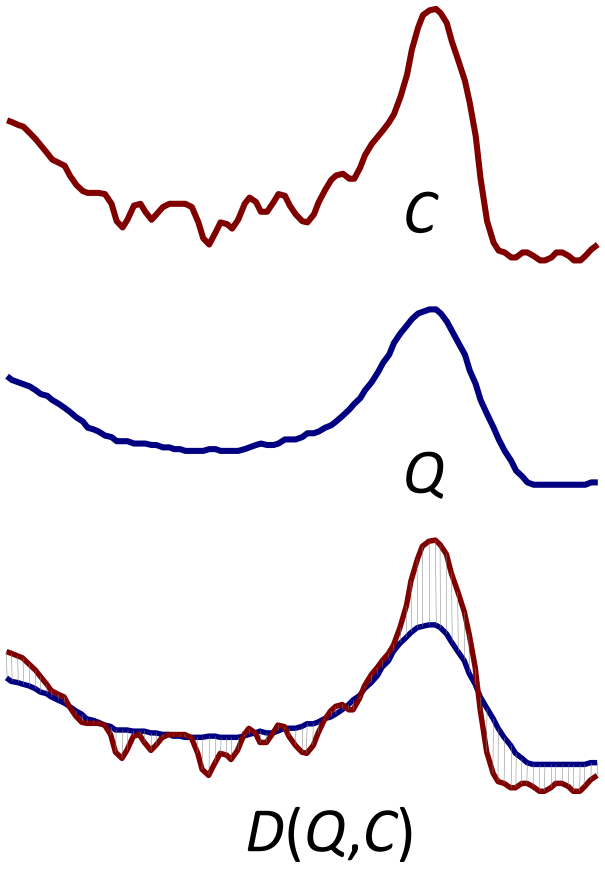
Raw Signal	1st Derivative	FFT
Mean	Mean	Top N freqs
Variance	Variance	Fundamental
Skewness	Skewness	
Kurtosis	Kurtosis	

Supervised Learning Model

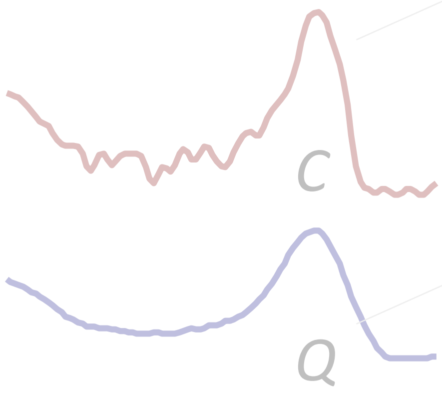
Output

TWO PREVAILING APPROACHES

SHAPE-MATCHING



FEATURE-BASED APPROACH



Raw Signal	1st Derivative	FFT
Mean	Mean	Top N freqs
Variance	Variance	Fundamental
Skewness	Skewness	
Kurtosis	Kurtosis	

Raw Signal	1st Derivative	FFT
Mean	Mean	Top N freqs
Variance	Variance	Fundamental
Skewness	Skewness	
Kurtosis	Kurtosis	

Supervised Learning Model

Output

INTRO TO SHAPE-MATCHING

SIMILARITY METRICS

How do we formally define similarity?

DEFINING DISTANCE MEASURES

Let O_1 and O_2 be two objects from the universe of possible objects.

The distance (dissimilarity) is denoted by $D(O_1,O_2)$

Properties of a desirable distance metric:

D(A,B) = D(B,A)	Symmetry
D(A,A) = 0	Constancy
D(A,B) = 0 IIf A= B	Positivity

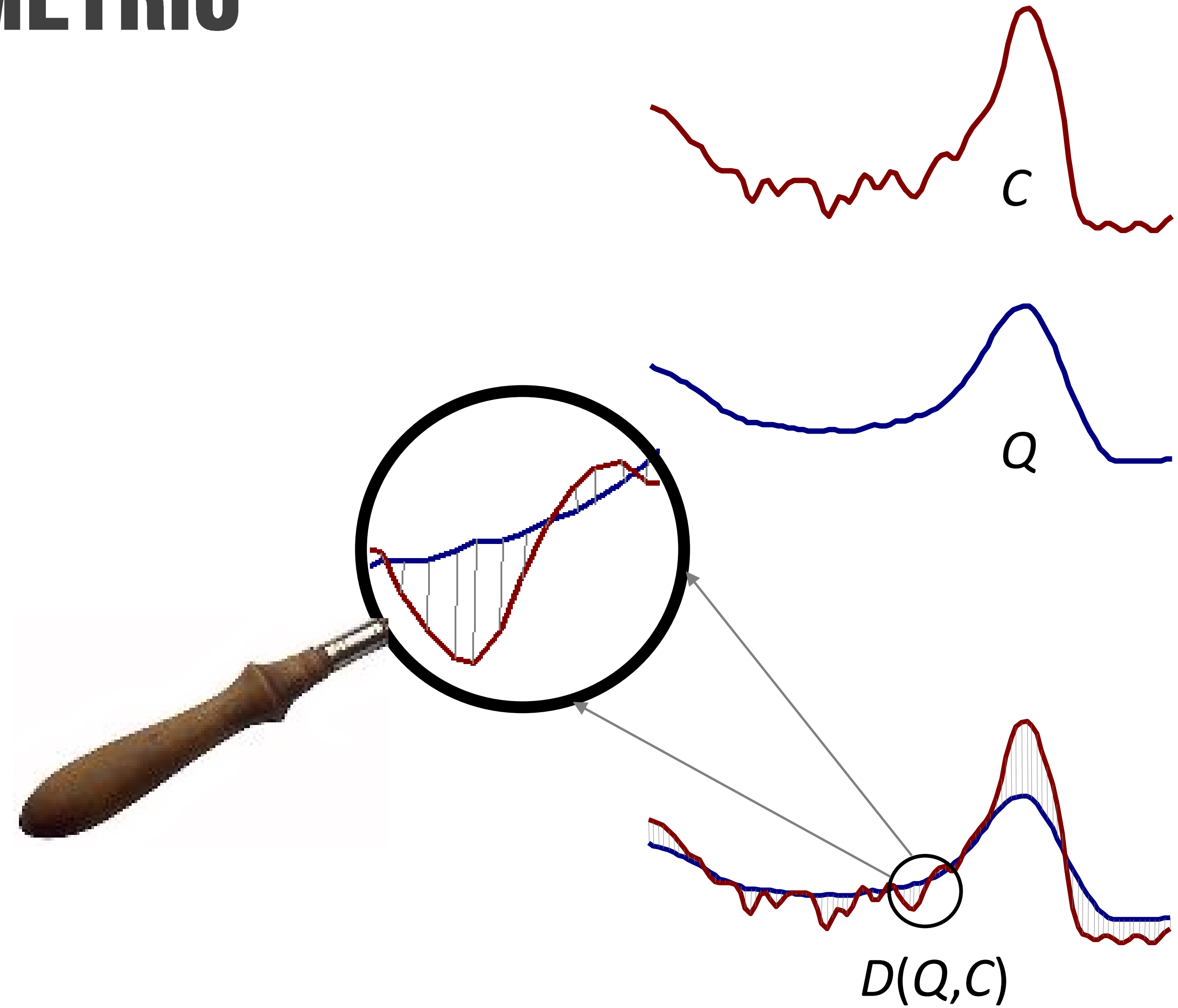
EUCLIDEAN DISTANCE METRIC

Given two time series:

$$Q = q_1 \ldots q_n$$

$$C = c_1 \ldots c_n$$

$$D(Q,C) \equiv \sqrt{\sum_{i=1}^{n} (q_i - c_i)^2}$$



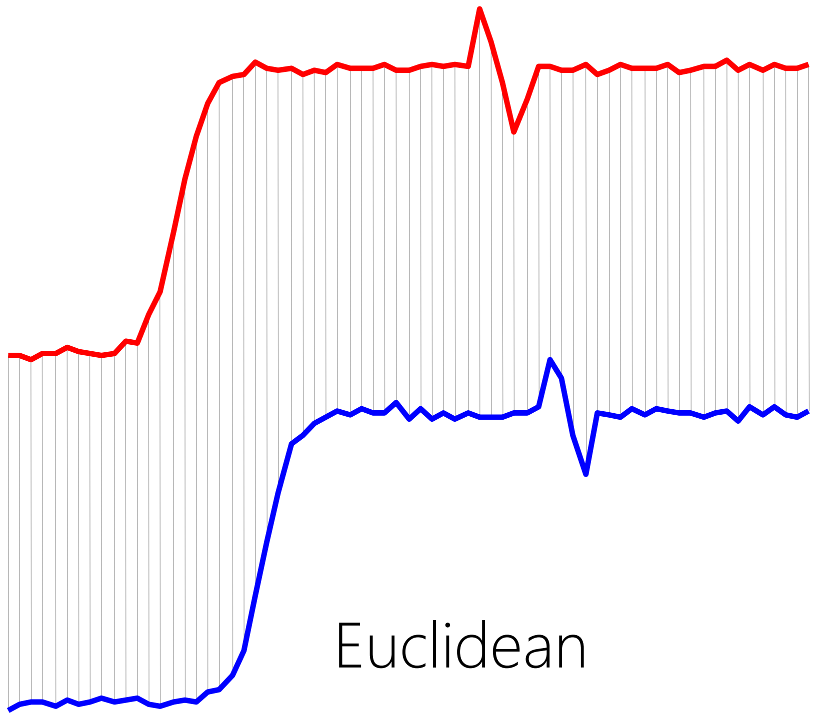
COMMON OPTIMIZATION OF EUCLID DISTANCE

Instead of calculating the raw Euclidean distance, calculate the squared Euclidean distance (to save CPU time)

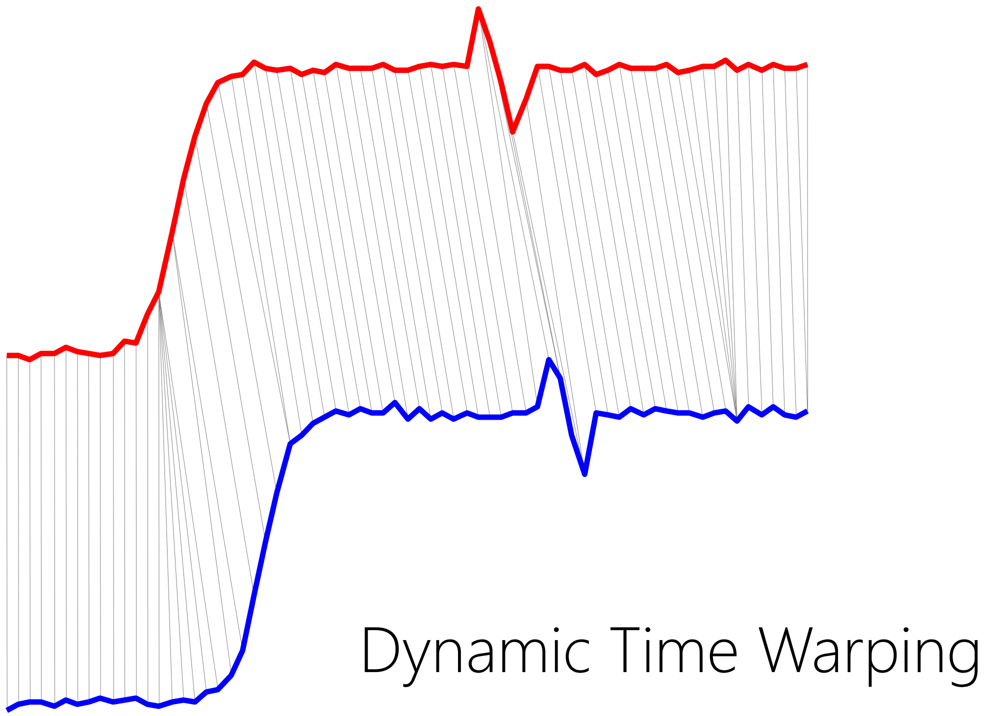
$$D(Q,C) \equiv \sqrt{\sum_{i=1}^{n}(q_i - c_i)^2} \quad \Rightarrow \quad D_{squared}(Q,C) \equiv \sum_{i=1}^{n}(q_i - c_i)^2$$

DYNAMIC TIME WARPING

Dynamic Time Warping is a more sophisticated similarity approach; however, it is $O(N^2)$



Fixed Time Axis
Sequences are aligned "one-to-one"



"Warped" Time Axis
Non-linear alignments are possible

PREPROCESSING TIME SERIES DATA

Shape-matching algorithms are particularly sensitive to distortions in the data that don't matter but greatly impact performance

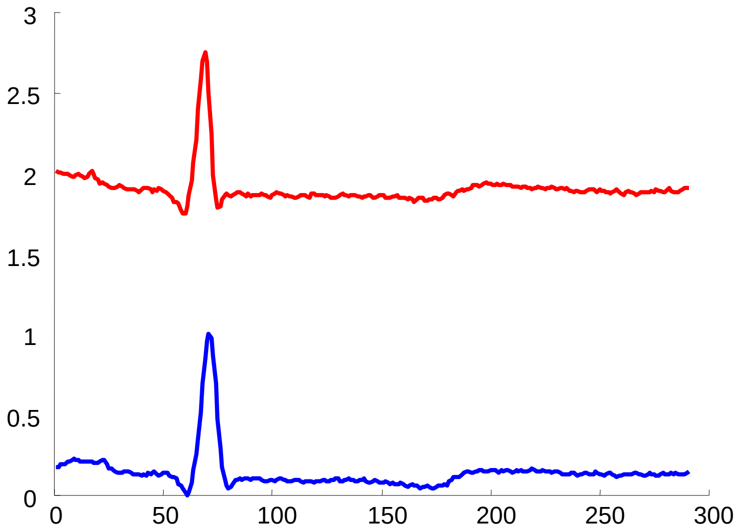
Offset translation (*e.g.*, demeaning)

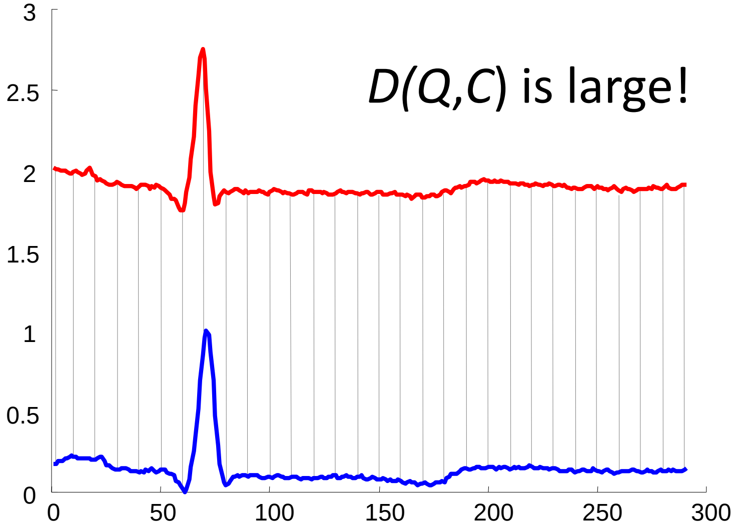
Amplitude scaling

Detrending (*e.g.*, removing linear trend)

Removing noise

OFFSET TRANSLATION

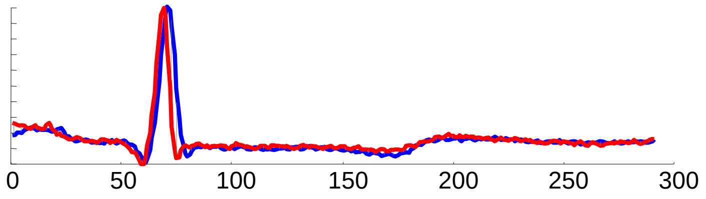




$D(Q,C)$ is large!

$Q = Q$ - mean(Q)

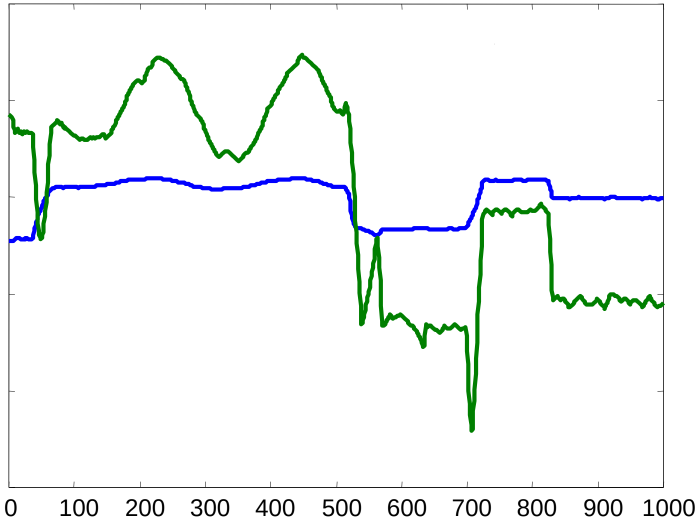
$C = C$ - mean(C)

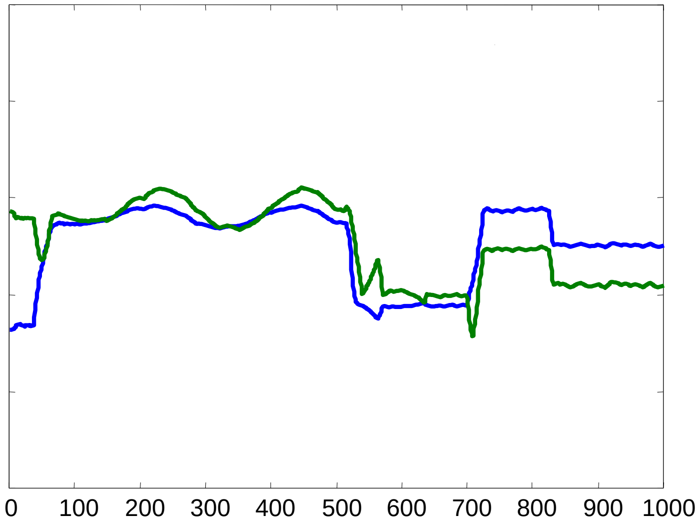


$D(Q,C)$ is now small!

PREPROCESSING TRANSFORMATIONS

AMPLITUDE SCALING



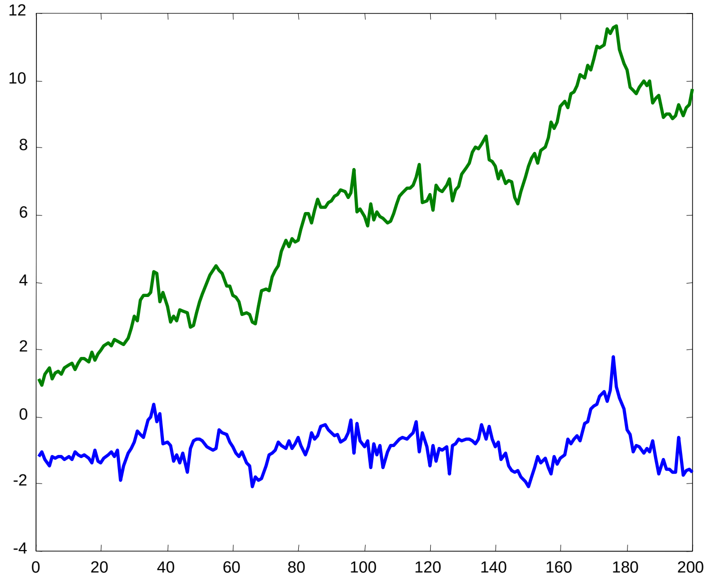


Q = (Q - mean(Q)) / std(Q)

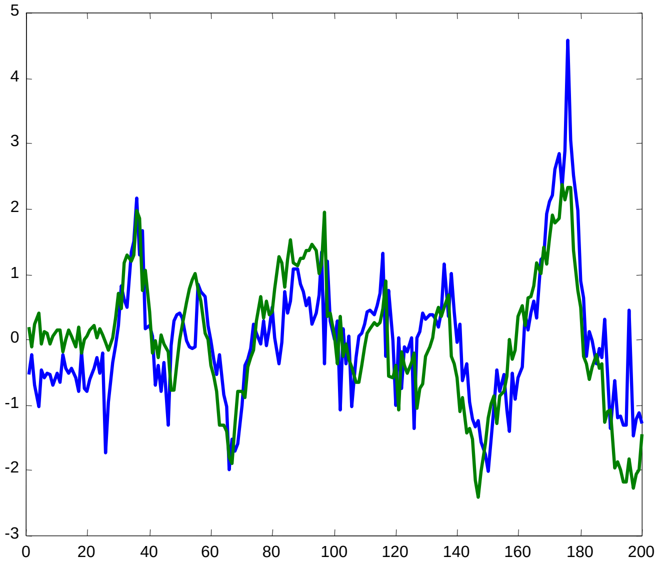
C = (C - mean(C)) / std(C)

PREPROCESSING TRANSFORMATIONS

DETRENDING



Find best fit line to time series, then subtract that line from the signal

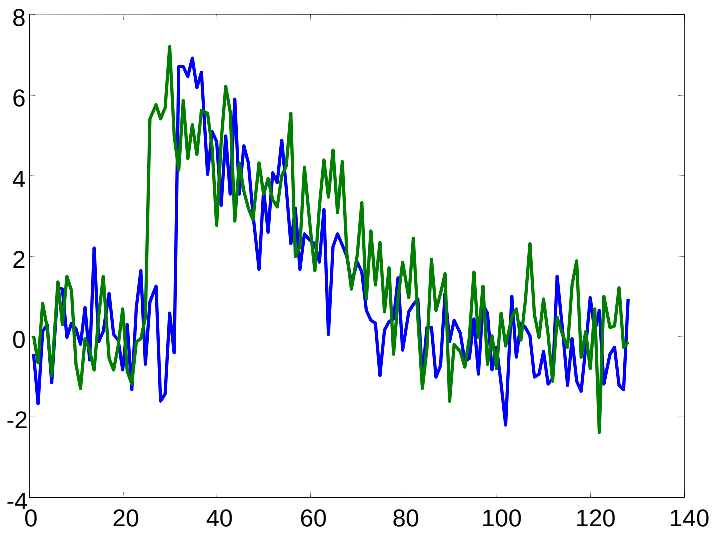


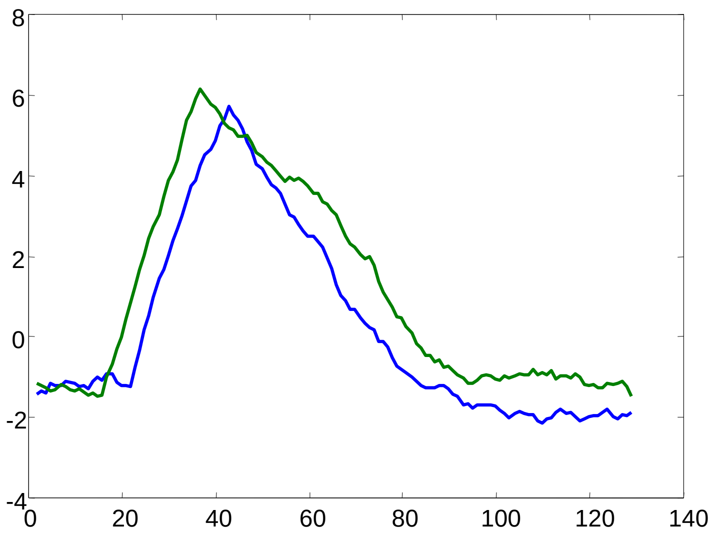
$Q = \mathrm{detrend}(Q)$

$C = \mathrm{detrend}(C)$

PREPROCESSING TRANSFORMATIONS

SMOOTHING





$$Q = \text{smooth}(Q)$$

$$C = \text{smooth}(C)$$

Project Jupyter | Home

jupyter.org

jupyter

Install About Us Community Documentation NBViewer Widgets Blog

jupyter

Project Jupyter exists to develop open-source software, open-standards, and services for interactive computing across dozens of programming languages.

JupyterCon 2018

August 21 - 25

Installing Jupyter

Get up and running with the Jupyter Notebook on your computer within minutes!

Prerequisite: Python

While Jupyter runs code in many programming languages, **Python** is a requirement (Python 3.3 or greater, or Python 2.7) for installing the Jupyter Notebook itself.

Installing Jupyter using Anaconda

We strongly recommend installing Python and Jupyter using the Anaconda Distribution, which includes Python, the Jupyter Notebook, and other commonly used packages for scientific computing and data science.

First, download Anaconda. We recommend downloading Anaconda's latest Python 3 version.

Second, install the version of Anaconda which you downloaded, following the instructions on the download page.

Congratulations, you have installed Jupyter Notebook! To run the notebook, run the following command at the Terminal (Mac/Linux) or Command Prompt (Windows):

```
jupyter notebook
```

See Running the Notebook for more details.

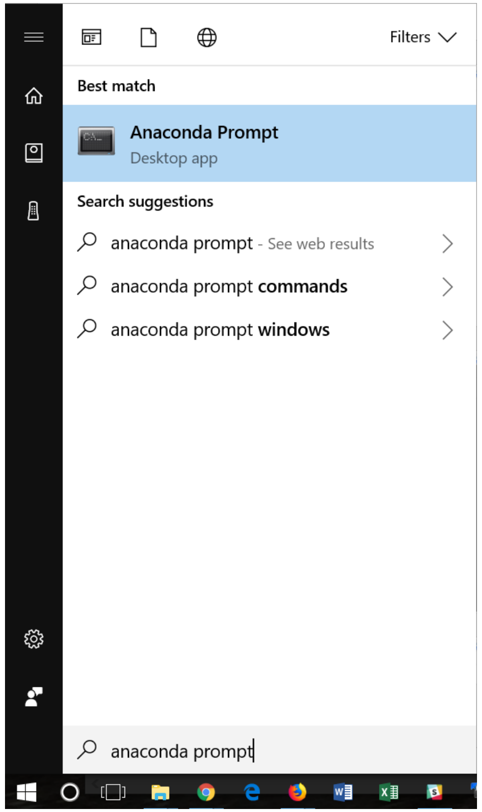
Installing Jupyter with pip

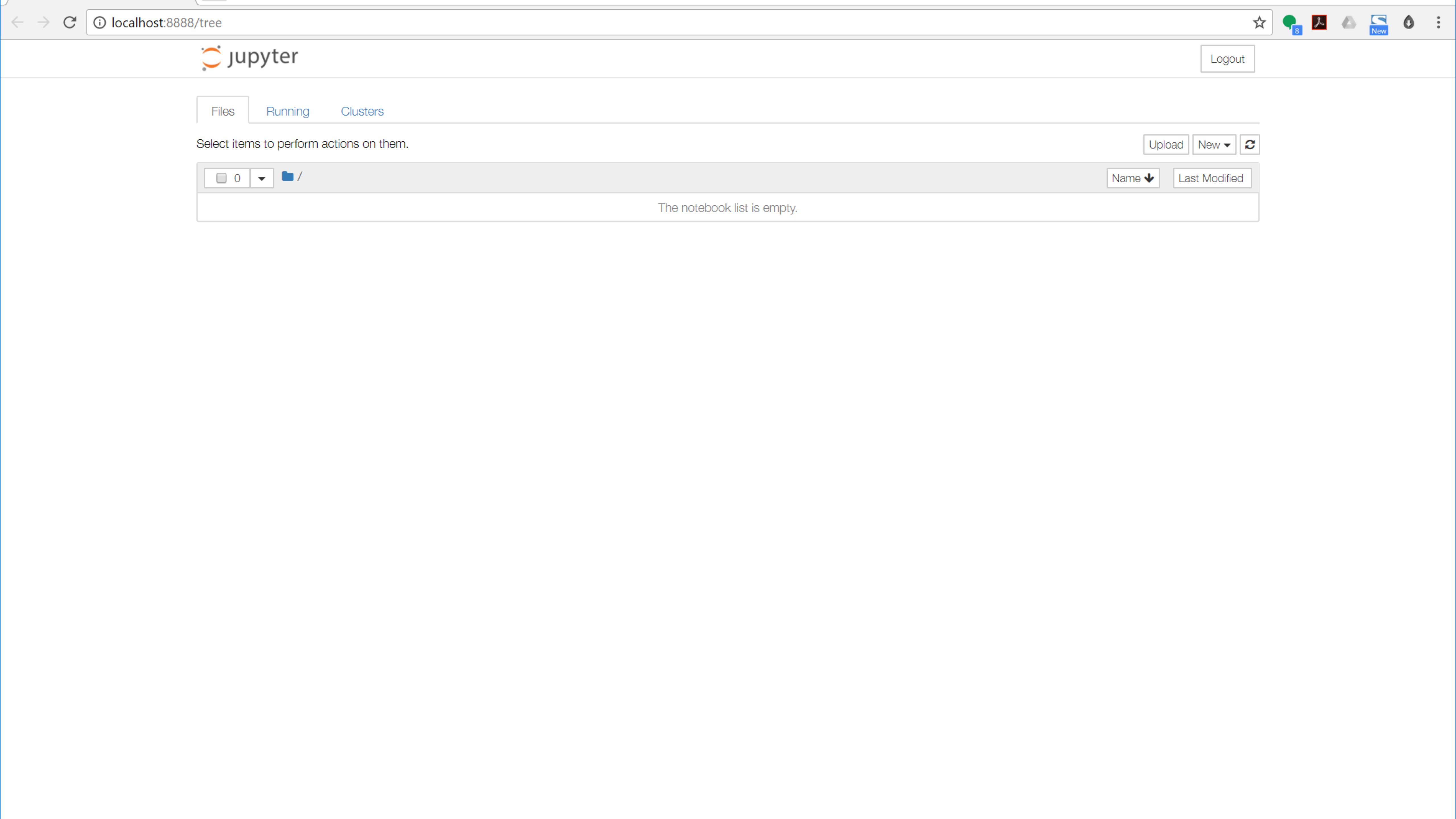
STARTING A FRESH JUPYTER NOTEBOOK

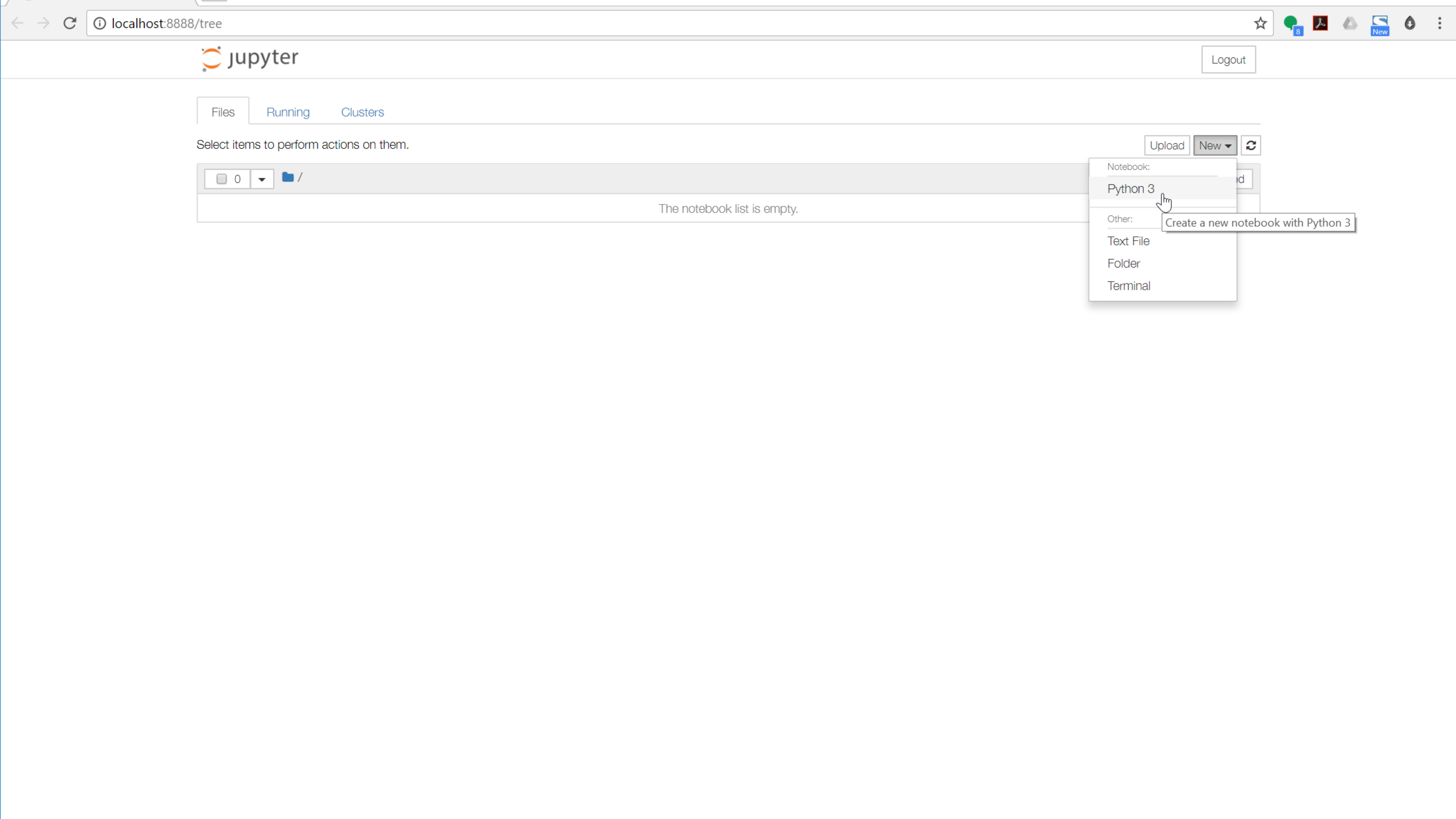
On a Mac

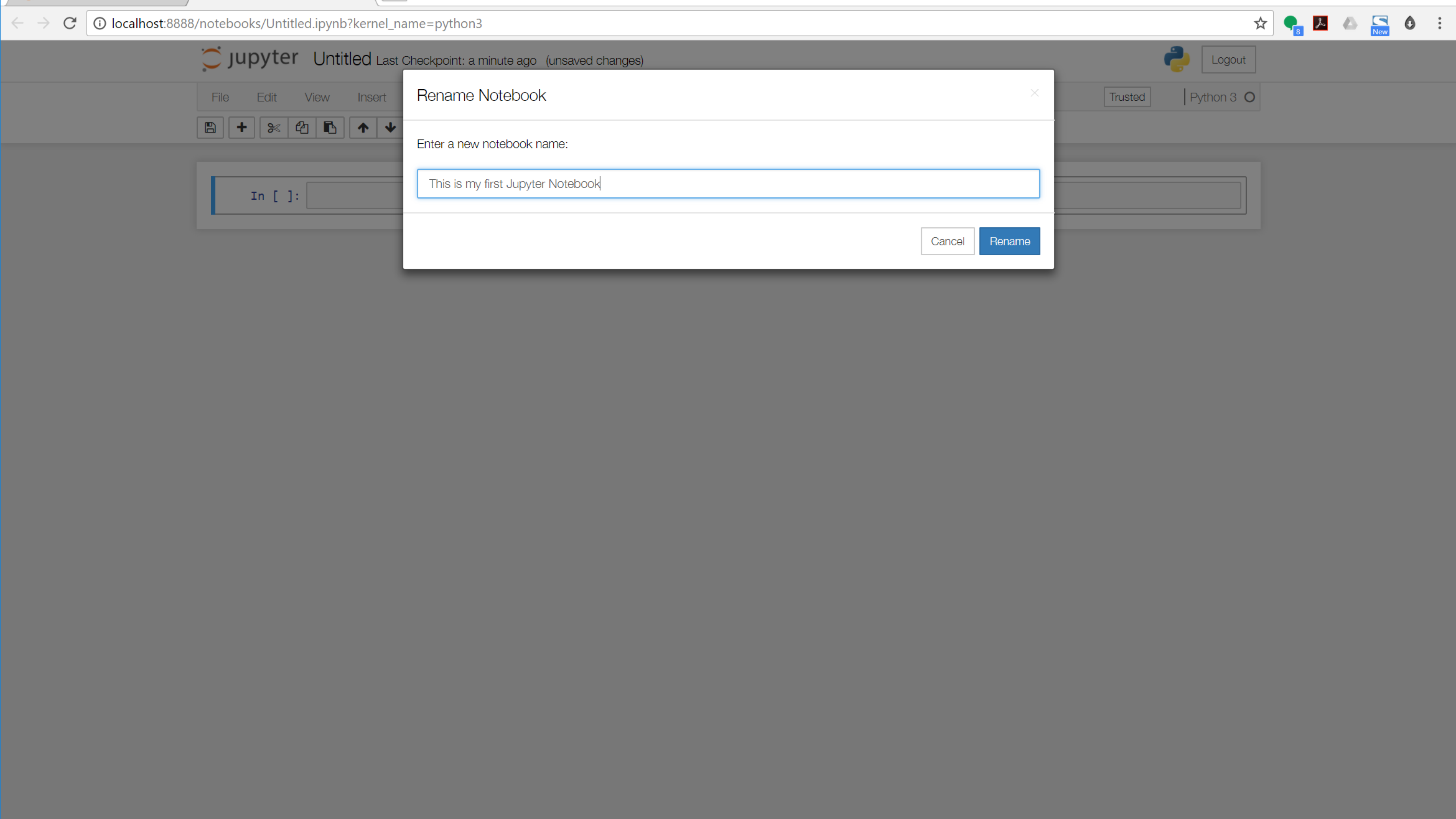
```
> mkdir funtimes

> cd funtimes

> jupyter notebook
```

On Windows




localhost:8888/tree
jupyter
Logout
Files
Running
Clusters
Select items to perform actions on them.
Upload
New
0
/
Name
Last Modified
The notebook list is empty.


localhost:8888/tree
jupyter
Logout
Files
Running
Clusters
Select items to perform actions on them.
Upload
New
0
/
The notebook list is empty.
Notebook:
Python 3
Create a new notebook with Python 3
Other:
Text File
Folder
Terminal


localhost:8888/notebooks/Untitled.ipynb?kernel_name=python3
jupyter
Untitled Last Checkpoint: a minute ago (unsaved changes)
Logout
File
Edit
View
Insert
Trusted
Python 3
In [ ]:
Rename Notebook
Enter a new notebook name:
This is my first Jupyter Notebook
Cancel
Rename

localhost:8888/notebooks/This%20is%20my%20first%20Jupyter%20Notebook.ipynb

jupyter This is my first Jupyter Notebook Last Checkpoint: 2 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted Python 3

Run Code

In [1]:
```
"Hello World"
```

Out[1]:
```
'Hello World'
```

In [ ]:

localhost:8888/notebooks/This%20is%20my%20first%20Jupyter%20Notebook.ipynb

jupyter This is my first Jupyter Notebook Last Checkpoint: 3 minutes ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Run Code

In [1]:
```
"Hello World"
```
Out[1]:
```
'Hello World'
```

In [3]:
```
myList = [1, 2, 3, 4, 5]
print(myList)
```
```
[1, 2, 3, 4, 5]
```

In [ ]:

localhost:8888/notebooks/This%20is%20my%20first%20Jupyter%20Notebook.ipynb

jupyter This is my first Jupyter Notebook Last Checkpoint: 8 minutes ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

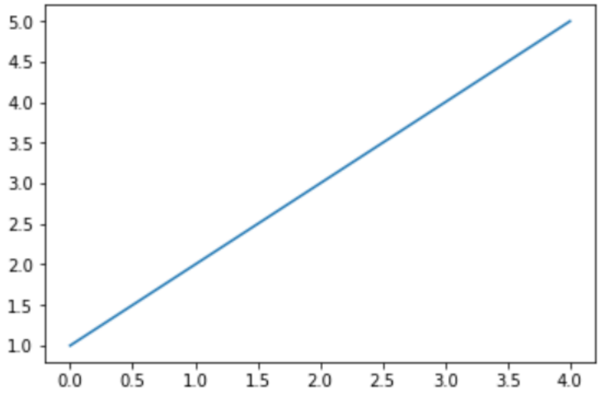
Trusted | Python 3

Run Code

In [1]:
```
"Hello World"
```
Out[1]:
```
'Hello World'
```

In [3]:
```
myList = [1, 2, 3, 4, 5]
print(myList)
```
```
[1, 2, 3, 4, 5]
```

In [6]:
```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
ax.plot(myList)
```
Out[6]:
```
[<matplotlib.lines.Line2D at 0x1e822d0e8d0>]
```



In [ ]:

localhost:8888/notebooks/This%20is%20my%20first%20Jupyter%20Notebook.ipynb

jupyter This is my first Jupyter Notebook Last Checkpoint: 10 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

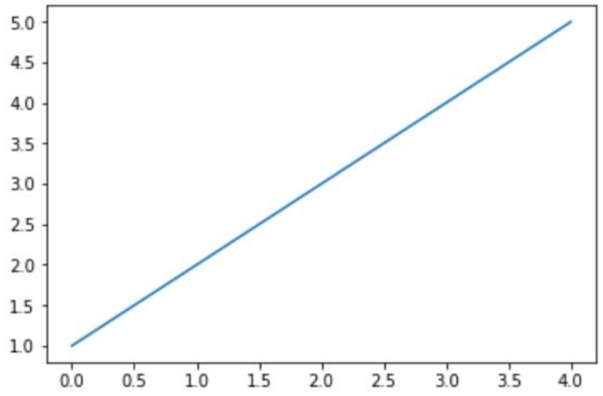
Run Code

In [6]:

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
ax.plot(myList)
```

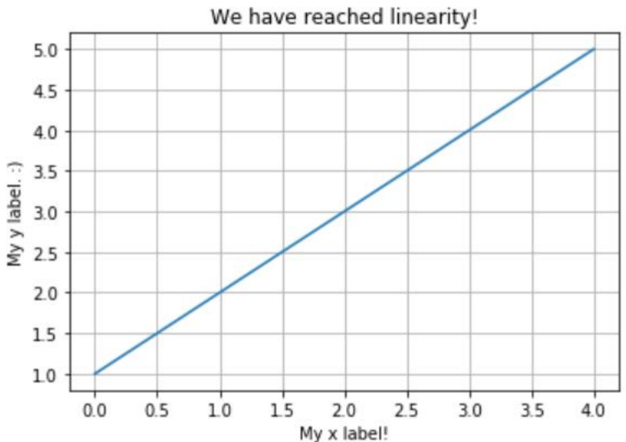
Out[6]:

```
[<matplotlib.lines.Line2D at 0x1e822d0e8d0>]
```



In [8]:

```
fig, ax = plt.subplots()
ax.plot(myList)
ax.set(xlabel='My x label!', ylabel='My y label. :)',
       title='We have reached linearity!')
ax.grid()
```



By this point in the lecture, Jon has switched to using Jupyter Notebook.

See: L03-PlayingWithSignals in https://github.com/jonfroehlich/CSE590Sp2018

SCHEDULE TODAY: 6:30-9:20

06:35-06:55: Carlos Burkle's discussion of the Bao *et al.*, 2004 activity rec reading

06:55-07:05: Joe Wandyez's discussion of the Dourish, 2004 context reading (link)

07:05-07:55: Signal processing & ML in ubicomp systems (with Jupyter Notebook exercises)

07:55-08:05: Break (begin graded demos with Liang)

08:05-08:35: A1: Step Tracker Presentations

08:35-08:45: Introduce and Discuss A2: Gesture Recognition assignment

08:45-09:20: Work on A2: Gesture Recognition in Jupyter Notebook

Quick Feedback about A1: Step Tracker

As mentioned previously, this is my first time teaching this course and my first time teaching PMP students. As such, I am continuously calibrating and revising my teaching approach. Your honest feedback here is appreciated and will be used to improve the course.

Quiz Type	Ungraded Survey
Points	
Shuffle Answers	No
Time Limit	No Time Limit
Multiple Attempts	No
View Responses	Always
Show Correct Answers	No
One Question at a Time	No
Anonymous Submissions	No

Due	For	Available from	Until
-	Everyone	-	-

Preview

Moderate This Survey

SCHEDULE TODAY: 6:30-9:20

06:35-06:55: Carlos Burkle’s discussion of the Bao *et al.*, 2004 activity rec reading

06:55-07:05: Joe Wandyez’s discussion of the Dourish, 2004 context reading (link)

07:05-07:55: Signal processing & ML in ubicomp systems (with Jupyter Notebook exercises)

07:55-08:05: Break

08:05-08:35: A1: Step Tracker Presentations

08:35-08:45: Introduce and Discuss A2: Gesture Recognition assignment

08:45-09:20: Work on A2: Gesture Recognition in Jupyter Notebook

A2: Gesture Recognizer

Overview

Imagine working for a company that would like to use the phone as an input device to an interactive video game system like the XBox or Playstation--for example, using the phone as a paddle in tennis or as a "ball" in bowling. In this assignment, you will build your own 3D gesture recognizer to automatically recognize these gestures.

While in the "real world" you would ultimately need to create a real-time gesture recognizer, for this assignment you will make an *offline* version in Jupyter Notebook. Specifically, you will make two recognizers: (i) a *shape-matching* recognizer such as via a Euclidean distance metric or Dynamic Time Warping and (ii) a *feature-based* (or *model-based*) recognizer using a sliding-window support-vector machine (SVM), hidden-markov model (HMM), or an alternative supervised learning approach of your choosing. An initial version of your shape-matching recognizer and performance results are due next week for your check-in.

Within Jupyter Notebook, we will use Python 3 and these amazing libraries numpy, scipy, matplotlib, and scikit-learn. Numpy and scipy provide numeric array handling and signal processing, matplotlib provides visualization, and scikit-learn is the de facto machine learning library in Python. You are welcome to use other libraries as well (*e.g.*, this DTW library).

For your deliverables, you will turn in your Jupyter Notebook, your recorded gestures, and a brief (1-page) report on your algorithmic approaches and performance results.

Learning Goals

- Introduce and learn basic machine learning approaches popular in ubiquitous computing systems, including shape matching and feature-based classification
- Introduce and learn basic machine learning pipeline: data collection, signal processing, model training, and model testing using k-fold cross validation
- Introduce and learn popular data analytics tools and toolkits (*e.g.*, Jupyter Notebook, scipy). I hope you'll enjoy learning and using Jupyter Notebook as much as we do!

ASSIGNMENT

A2: GESTURE RECOGNIZER

1. Using Jupyter Notebook, design and implement two different (offline) gesture recognition approaches:
   1. Shape-matching approach (*e.g.*, Euclidean distance)
   2. A model-based approach (*e.g.*, a multi-class SVM)
2. You must test on two gesture sets: one gesture set that I pre-created and another that you create. You will use the A02-GestureLogger tool (see github)
3. To help get you started, I created an initial Jupyter Notebook with some basic data structures and parsing for analyzing the gesture data. (again, see github)
4. You will submit your Jupyter Notebooks and a brief report

Your gesture recognizer needs to recognize:

1. Backhand Tennis
2. Forehand Tennis
3. Underhand Bowling
4. Baseball Throw
5. At Rest (i.e., no motion)
6. Midair Clockwise 'O'
7. Midair Counter-clockwise 'O'
8. Midair Zorro 'Z'
9. Midair 'S'
10. Shake
11. A gesture of your own creation

SCHEDULE TODAY: 6:30-9:20

06:35-06:55: Carlos Burkle's discussion of the Bao *et al.*, 2004 activity rec reading

06:55-07:05: Joe Wandyez's discussion of the Dourish, 2004 context reading (link)

07:05-07:55: Signal processing & ML in ubicomp systems (with Jupyter Notebook exercises)

07:55-08:05: Break

08:05-08:35: A1: Step Tracker Presentations

08:35-08:45: Introduce and Discuss A2: Gesture Recognition assignment

08:45-09:20: Work on A2: Gesture Recognition in Jupyter Notebook