Working With Data

Phys 281 – Class 4

Grant Wilson

Answers to Exercises

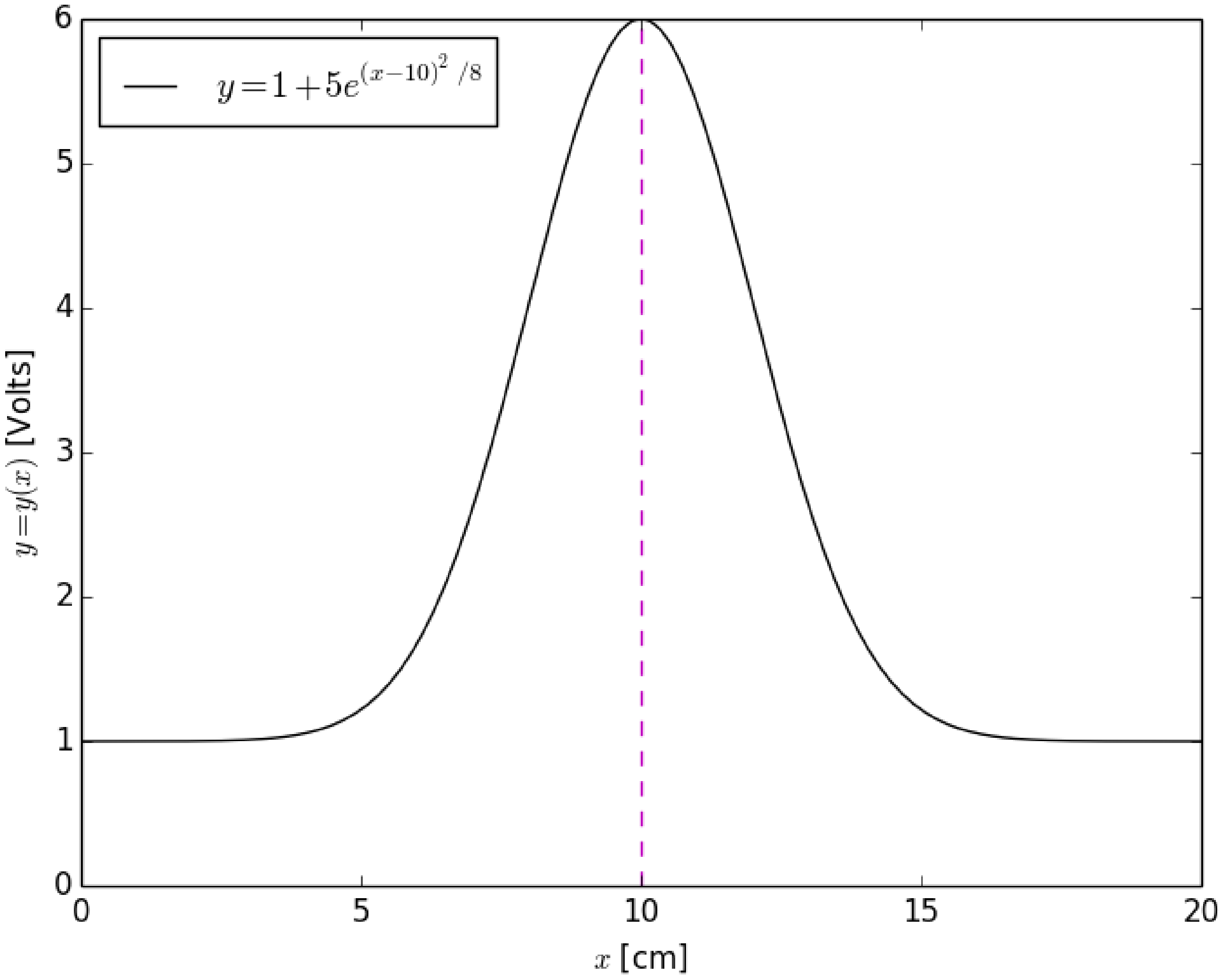
```
from matplotlib import pyplot as plt
import numpy as np
plt.ion()

#E3.1 - plot a gaussian
offset = 1.
amplitude = 5.
centroid = 10.
std = 2.

def Gaussian(offset, amplitude, centroid, std, x):
    return offset + amplitude*np.exp(-0.5*(x-centroid)**2/std**2)

#define our plotting vectors
x = np.linspace(0,20,200,endpoint=True)
g = Gaussian(offset, amplitude, centroid, std, x)

#make the plot
plt.plot(x,g,'k',label="$y = 1+5e^{(x-10)^2/8}$")
plt.xlabel('$x$ [cm]')
plt.ylabel('$y = y(x)$ [Volts]')
plt.plot([10,10],[0,g.max()],'m',linestyle='--')
plt.legend(loc='upper left')
plt.savefig("E3.1_plot.png")
```



$y = 1 + 5e^{(x-10)^2\ /8}$
$y = y(x)$ [Volts]
x [cm]
0
1
2
3
4
5
6
0
5
10
15
20

```
#E3.2 - reproducing bad correlation plot

#the data - pulled from Moodle
dates = np.array([1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010])
drownings = np.array([19, 16, 9, 12, 15, 10, 11, 7, 2, 5, 1, 1])
marriages = np.array([10.9, 9.8, 9, 9, 9.1, 8.8, 8.7, 8.4, 7.8, 7.9, 7.6,7.4])

dateshr = np.linspace(1999,2010,200,endpoint=True)
tck = interpolate.splrep(dates,drownings,s=0)
drowningshr = interpolate.splev(dateshr,tck,der=0)
tck = interpolate.splrep(dates,marriages,s=0)
marriageshr = interpolate.splev(dateshr,tck,der=0)


#the drownings plot
plt.figure(figsize=(12,4),dpi=80,facecolor='black')
plt.plot(dateshr,drowningshr,'-m',label='People who drowned after falling out of a fishing boat')
plt.plot(dates,drownings,'om')
plt.xlim(1999,2010)
plt.xticks(np.linspace(1999,2010,12,endpoint=True))
plt.ylim(0,20)
plt.yticks([0,10,20])
plt.ylabel("Deaths (US)")
```

```
#E3.2 - reproducing bad correlation plot (continued)
#at this point the plot is mostly black and yellow on black
#recolor the axes and the labels
ax1 = plt.gca()
ax1.spines['right'].set_color('none')
ax1.spines['top'].set_color('none')
ax1.spines['bottom'].set_color('yellow')
ax1.spines['left'].set_color('yellow')

#recolor the ticks
for ticks in ax1.xaxis.get_ticklines() + ax1.yaxis.get_ticklines():
        ticks.set_color('yellow')

#recolor the labels
for label in ax1.xaxis.get_ticklabels() + ax1.yaxis.get_ticklabels():
    label.set_color('yellow')
label = ax1.yaxis.get_label()
label.set_color('yellow')

#set the face color of the plot to black
rect = ax1.patch
rect.set_facecolor('k')
```

```
#E3.2 - reproducing bad correlation plot (continued)

#use twinx and twiny to deal with the other axes
ax2 = ax1.twinx()
plt.plot(dateshr,marriageshr,'-y', label="Marriage rate in Kentucky")
plt.plot(dates,marriages,'oy')
plt.ylim([7,11])
plt.yticks([7,8,9,10,11])
ax2.set_ylabel('Marriages per 1000 people', color='y')

#again, deal with the label and axis colors
for tl in ax2.get_yticklabels():
    tl.set_color('y')
for ticks in ax2.yaxis.get_ticklines():
    ticks.set_color('yellow')

#there are a few more ticks that show up yellow that we don’t want
#so color them black
ax3 = ax1.twiny()
for ticks in ax3.xaxis.get_ticklines():
    ticks.set_color('k')

#and finally, save the figure to a png.
plt.savefig("E3.2_plot.png")
```

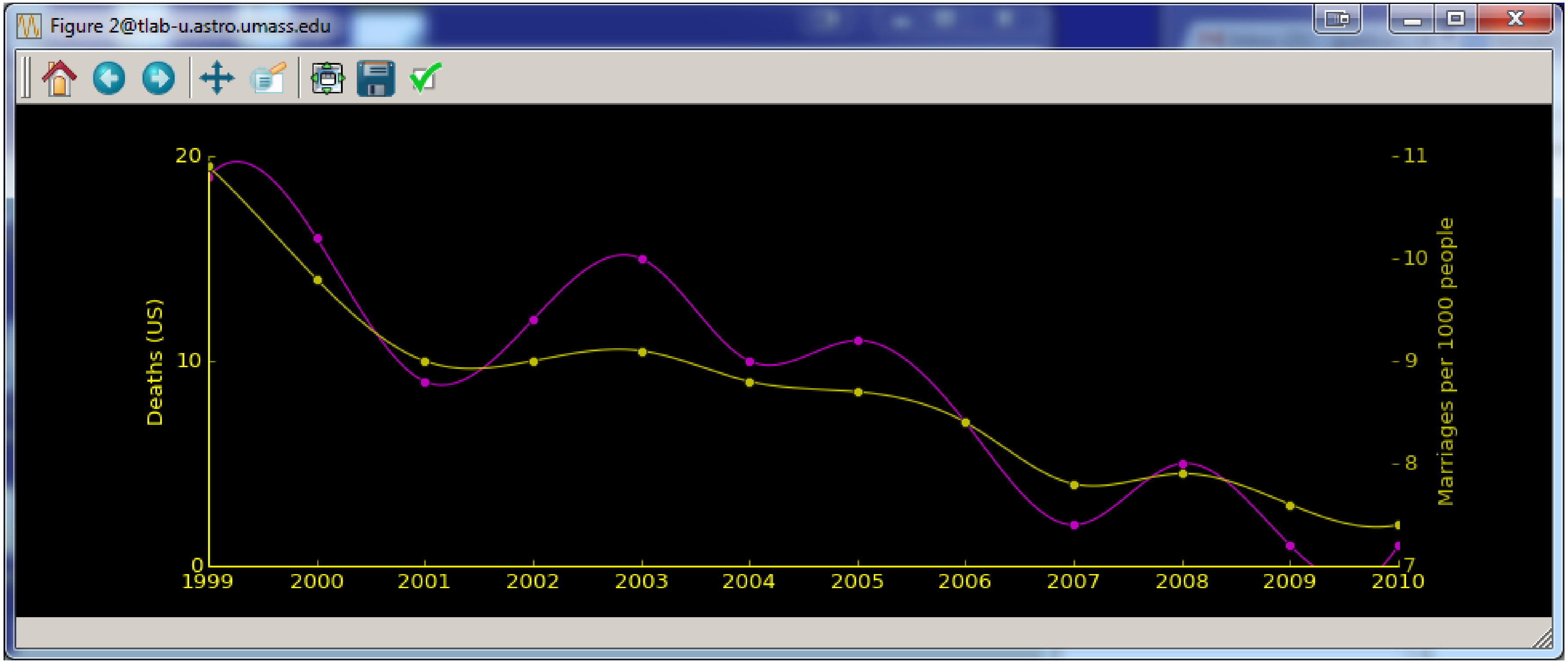


Figure 2@tlab-u.astro.umass.edu
Deaths (US)
Marriages per 1000 people
20
10
0
11
10
9
8
7
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010

Working With Data

Phys 281 – Class 4

Grant Wilson

Some Data Formats

ASCII (human readable)

- American Standard Code for Information Interchange
  - 128 characters (English alphabet)
  - numbers 0-9
  - some punctuation
  - some control codes
  - everything stored in 7-bit integers

- Bottom line:
  - Great because it is human readable!
  - Terribly inefficient for storage and very limited.

- Now Unicode (aka wide-body ascii) is the standard for the consistent encoding, representation, and handling of text.

XML

- Extensible Markup Language
- Human readable and machine readable
- Unicode-based
- Used for representation of data structures
- Eg.

```
<President>
        <fname> Barak </fname>
        <lname> Obama </lname>
        <number> 44 </number>
        <political affiliation> Democrat </political affiliation>
</President>
```

- Easy to read and decode, still very inefficient for large amounts of data.

Binary Formatted Files

- In a binary file the data has been encoded into a sequence of bytes (or bits, or words, etc.)

- Each binary file type has its own encoding scheme and so it has its own decoding scheme.

- The text editor “emacs” has a binary file reading mode (Esc – x hexl-mode) that can sometimes be helpful.

Two common data file formats in Physics and Astronomy

- CDF (or NetCDF) – binary format with many utilities for human exploration of data.
  - see netCDF4 package

- FITS – Flexible Image Transport System - Astronomy standard for storing images and other arrays.
  - see PyFITS package

To get info from terminal

- use the raw_input() method (note that raw_input() != input())
- Try this:
  - *name = raw_input("Enter your name: ")*
  - *print "Your name is "+name*

- But don't forget to type-convert:
  - *one = raw_input("Enter the number 1: ")*
  - *two = raw_input("Now enter the number 2: ")*
  - *print "1/2 = ", one/two*

  Instead:
  - *one = float(raw_input("Enter the number 1: "))*
  - *two = float(raw_input("Now enter the number 2: "))*
  - *print "1/2 = ", one/two*

Saving and Loading Arrays from a File

- numpy has specific methods for this
- .npy files are binary files using the numpy encoding scheme

- save(file, arr) – store a single array in a file
  - file is the filename string (in quotes)
  - arr is the numpy array
  - Try this:
    - *x = np.arange(100)*
    - *np.save(“x.npy”, x)*

    now quit ipython

    start ipython up again
    - *import numpy as np*
    - *x = np.load(“x.npy”)*
    - *print x*

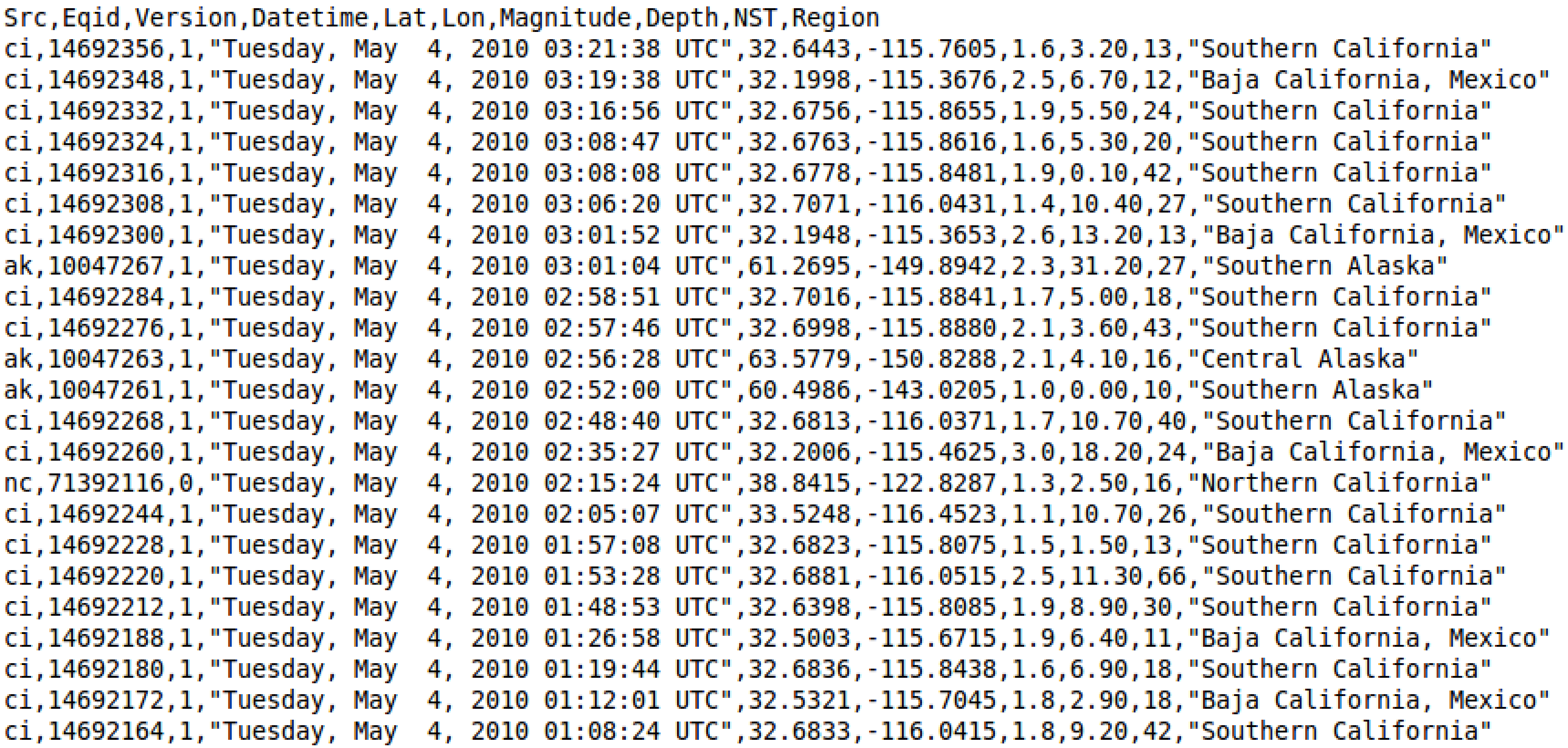
Saving and Loading Arrays from a File

- If you have more than one array you want to save in the file … use the .npz format
- savez(file, *args, **kwds) – save several arrays into a single file

- Try this:
  - *x = np.arange(10)*
  - *y = np.sin(x)*
  - *np.savez(“xy.npz”, x=x, y=y)*

  quit ipython and start up again
  - *import numpy as np*
  - *npz = np.load(“xy.npz”)*
  - *print npz.files*
  - *x1 = npz[‘x’]*
  - *y1 = npz[‘y’]*

Reading/Writing CSV files



```
Src,Eqid,Version,Datetime,Lat,Lon,Magnitude,Depth,NST,Region
ci,14692356,1,"Tuesday, May  4, 2010 03:21:38 UTC",32.6443,-115.7605,1.6,3.20,13,"Southern California"
ci,14692348,1,"Tuesday, May  4, 2010 03:19:38 UTC",32.1998,-115.3676,2.5,6.70,12,"Baja California, Mexico"
ci,14692332,1,"Tuesday, May  4, 2010 03:16:56 UTC",32.6756,-115.8655,1.9,5.50,24,"Southern California"
ci,14692324,1,"Tuesday, May  4, 2010 03:08:47 UTC",32.6763,-115.8616,1.6,5.30,20,"Southern California"
ci,14692316,1,"Tuesday, May  4, 2010 03:08:08 UTC",32.6778,-115.8481,1.9,0.10,42,"Southern California"
ci,14692308,1,"Tuesday, May  4, 2010 03:06:20 UTC",32.7071,-116.0431,1.4,10.40,27,"Southern California"
ci,14692300,1,"Tuesday, May  4, 2010 03:01:52 UTC",32.1948,-115.3653,2.6,13.20,13,"Baja California, Mexico"
ak,10047267,1,"Tuesday, May  4, 2010 03:01:04 UTC",61.2695,-149.8942,2.3,31.20,27,"Southern Alaska"
ci,14692284,1,"Tuesday, May  4, 2010 02:58:51 UTC",32.7016,-115.8841,1.7,5.00,18,"Southern California"
ci,14692276,1,"Tuesday, May  4, 2010 02:57:46 UTC",32.6998,-115.8880,2.1,3.60,43,"Southern California"
ak,10047263,1,"Tuesday, May  4, 2010 02:56:28 UTC",63.5779,-150.8288,2.1,4.10,16,"Central Alaska"
ak,10047261,1,"Tuesday, May  4, 2010 02:52:00 UTC",60.4986,-143.0205,1.0,0.00,10,"Southern Alaska"
ci,14692268,1,"Tuesday, May  4, 2010 02:48:40 UTC",32.6813,-116.0371,1.7,10.70,40,"Southern California"
ci,14692260,1,"Tuesday, May  4, 2010 02:35:27 UTC",32.2006,-115.4625,3.0,18.20,24,"Baja California, Mexico"
nc,71392116,0,"Tuesday, May  4, 2010 02:15:24 UTC",38.8415,-122.8287,1.3,2.50,16,"Northern California"
ci,14692244,1,"Tuesday, May  4, 2010 02:05:07 UTC",33.5248,-116.4523,1.1,10.70,26,"Southern California"
ci,14692228,1,"Tuesday, May  4, 2010 01:57:08 UTC",32.6823,-115.8075,1.5,1.50,13,"Southern California"
ci,14692220,1,"Tuesday, May  4, 2010 01:53:28 UTC",32.6881,-116.0515,2.5,11.30,66,"Southern California"
ci,14692212,1,"Tuesday, May  4, 2010 01:48:53 UTC",32.6398,-115.8085,1.9,8.90,30,"Southern California"
ci,14692188,1,"Tuesday, May  4, 2010 01:26:58 UTC",32.5003,-115.6715,1.9,6.40,11,"Baja California, Mexico"
ci,14692180,1,"Tuesday, May  4, 2010 01:19:44 UTC",32.6836,-115.8438,1.6,6.90,18,"Southern California"
ci,14692172,1,"Tuesday, May  4, 2010 01:12:01 UTC",32.5321,-115.7045,1.8,2.90,18,"Baja California, Mexico"
ci,14692164,1,"Tuesday, May  4, 2010 01:08:24 UTC",32.6833,-116.0415,1.8,9.20,42,"Southern California"
```

Reading/Writing CSV file

- There are several ways to do this.
  - numpy has methods for dealing with arrays
    - loadtxt
    - savetxt
  - There is also a csv module that is a bit more general
    - https://docs.python.org/2/library/csv.html

Example with csv module

- Download the csv file “my_first.csv” from the class Moodle page and put it in your class04 directory.
- Try this:

```
import csv
with open(‘my_first.csv’) as csvfile:
    csvreader = csv.reader(csvfile)
    for row in csvreader:
        print row
```

Example with csv module

```
import csv
with open(‘my_first.csv’) as csvfile:
    csvreader = csv.reader(csvfile)
    for row in csvreader:
        print row
```

Open the CSV file (referring to the open file as “csvfile”) and then properly close the file when the block of code below completes.

Example with csv module

```
import csv
with open(‘my_first.csv’) as csvfile:
    csvreader = csv.reader(csvfile)
    for row in csvreader:
        print row
```

Define a csv.reader object for the file. This is how we will have access to the file’s contents. The csv.reader accesses the file line by line.

Example with csv module

```
import csv
with open(‘my_first.csv’) as csvfile:
    csvreader = csv.reader(csvfile)
    for row in csvreader:
        print row
```

Now loop through the lines of the file (the “rows” in csvreader) and extract them as python lists.

Exercise

- Write a set of code to extract the x- and z-columns of data from my_first.csv file. Store the values as numpy arrays called x and z.

- Write the x and z numpy arrays to a binary .npz file called “my_first.npz” using the numpy methods.

- Restart ipython and read in the x and z arrays from “my_first.npz”. Make a plot of x –vs- z.