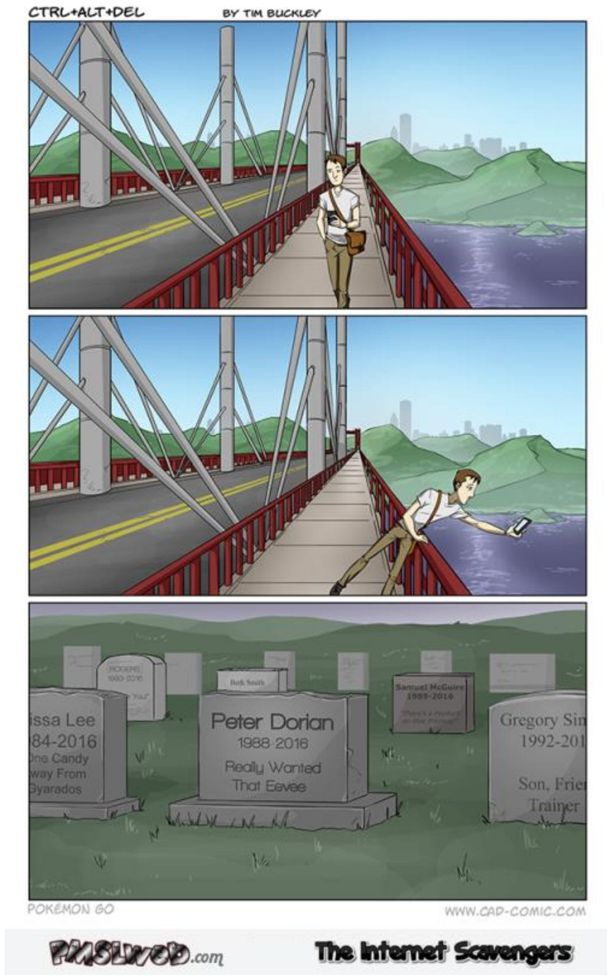


The Trouble with Files

(Hands on)

You'll need Jupyter.

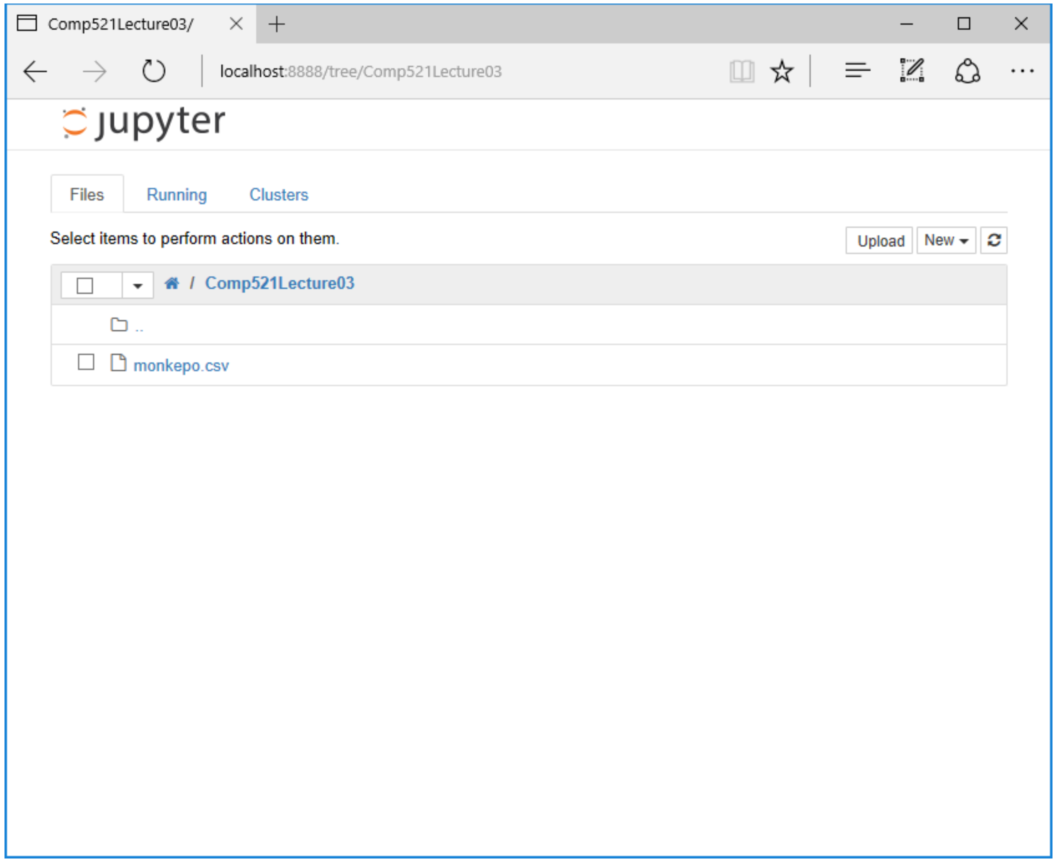
Warning: Today is easy. Mostly cut-and-paste. But, it is just a warm up for things to come. YOU WILL WRITE CODE ***IN*** this class.

They're here!

- Pandimensional aliens, called *Monkepo*, are among us! And, they're now *poking* into our universe.
- Recent technology has enabled their detection
- Millennials have been duped into detecting them using a popular smartphone app
- Open questions:
  - Where? What types? Can we find hotspots?

Let's look at the data

- A collection of "sightings" can be downloaded from: http://csbio.unc.edu/mcmillan/Media/monkepo.csv
- You can open it in a spreadsheet
- Save them into a directory/folder on your machine (other than "Downloads")
- Start up a Jupyter notebook in the same directory



Select items to perform actions on them.

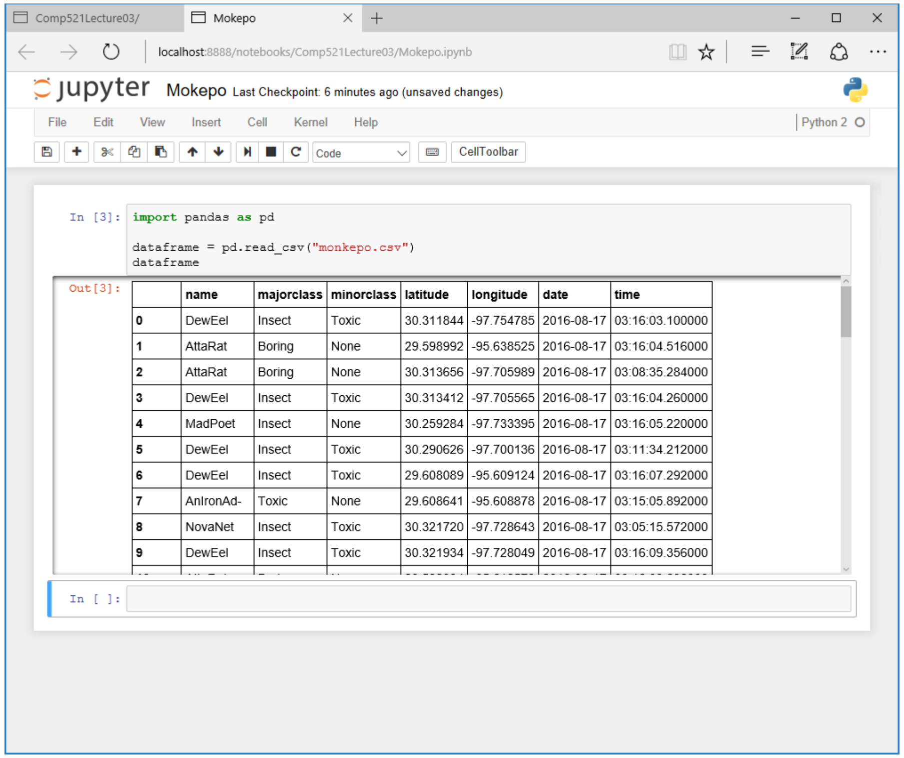
monkepo.csv

Read in the file

- Make a new Python2 notebook
- Rename it "Mokepo"
- Add 3 lines of code into a cell, and run it!

```
import pandas as pd

dataframe = pd.read_csv("monkepo.csv")
dataframe
```

- Scroll around. Get a sense for what the data looks like.

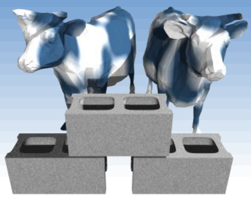

Comp521Lecture03/
Mokepo
localhost:8888/notebooks/Comp521Lecture03/Mokepo.ipynb
jupyter
Mokepo Last Checkpoint: 6 minutes ago (unsaved changes)
File
Edit
View
Insert
Cell
Kernel
Help
Python 2
Code
CellToolbar

```
In [3]: import pandas as pd

        dataframe = pd.read_csv("monkepo.csv")
        dataframe
```

Out[3]:

	name	majorclass	minorclass	latitude	longitude	date	time
0	DewEel	Insect	Toxic	30.311844	-97.754785	2016-08-17	03:16:03.100000
1	AttaRat	Boring	None	29.598992	-95.638525	2016-08-17	03:16:04.516000
2	AttaRat	Boring	None	30.313656	-97.705989	2016-08-17	03:08:35.284000
3	DewEel	Insect	Toxic	30.313412	-97.705565	2016-08-17	03:16:04.260000
4	MadPoet	Insect	None	30.259284	-97.733395	2016-08-17	03:16:05.220000
5	DewEel	Insect	Toxic	30.290626	-97.700136	2016-08-17	03:11:34.212000
6	DewEel	Insect	Toxic	29.608089	-95.609124	2016-08-17	03:16:07.292000
7	AnIronAd-	Toxic	None	29.608641	-95.608878	2016-08-17	03:15:05.892000
8	NovaNet	Insect	Toxic	30.321720	-97.728643	2016-08-17	03:05:15.572000
9	DewEel	Insect	Toxic	30.321934	-97.728049	2016-08-17	03:16:09.356000

```
In [ ]:
```

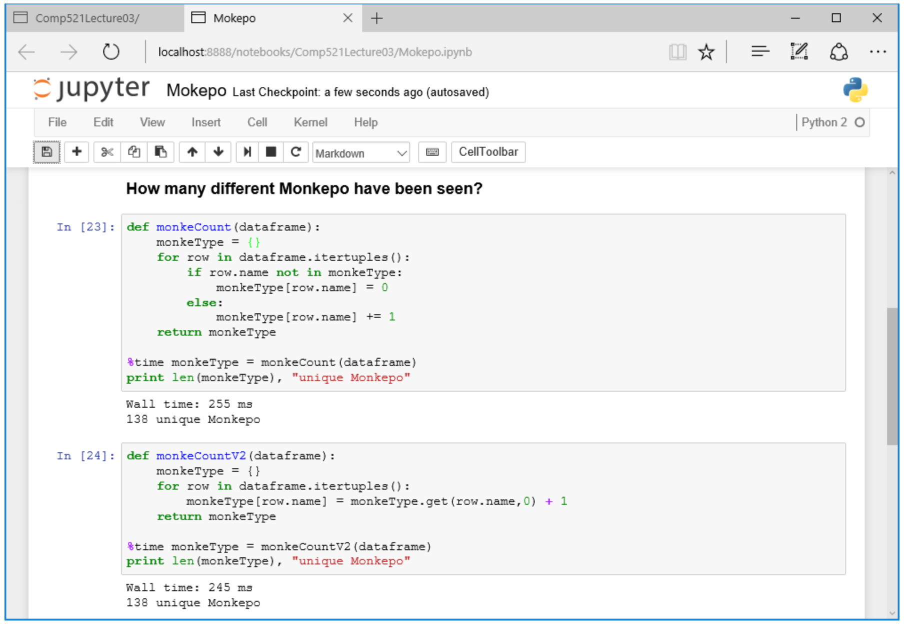


What Monkepo have been seen?

- Use a dictionary to count the occurrences of various Monkepo by their name.
- Dictionary

```
myDict = {'a' : 7, 'b' : 3, 'd' : 2 }

print myDict['a']
myDict['d'] += 6
print myDict
print 'c' in myDict
```

- Scan through a Pandas dataframe

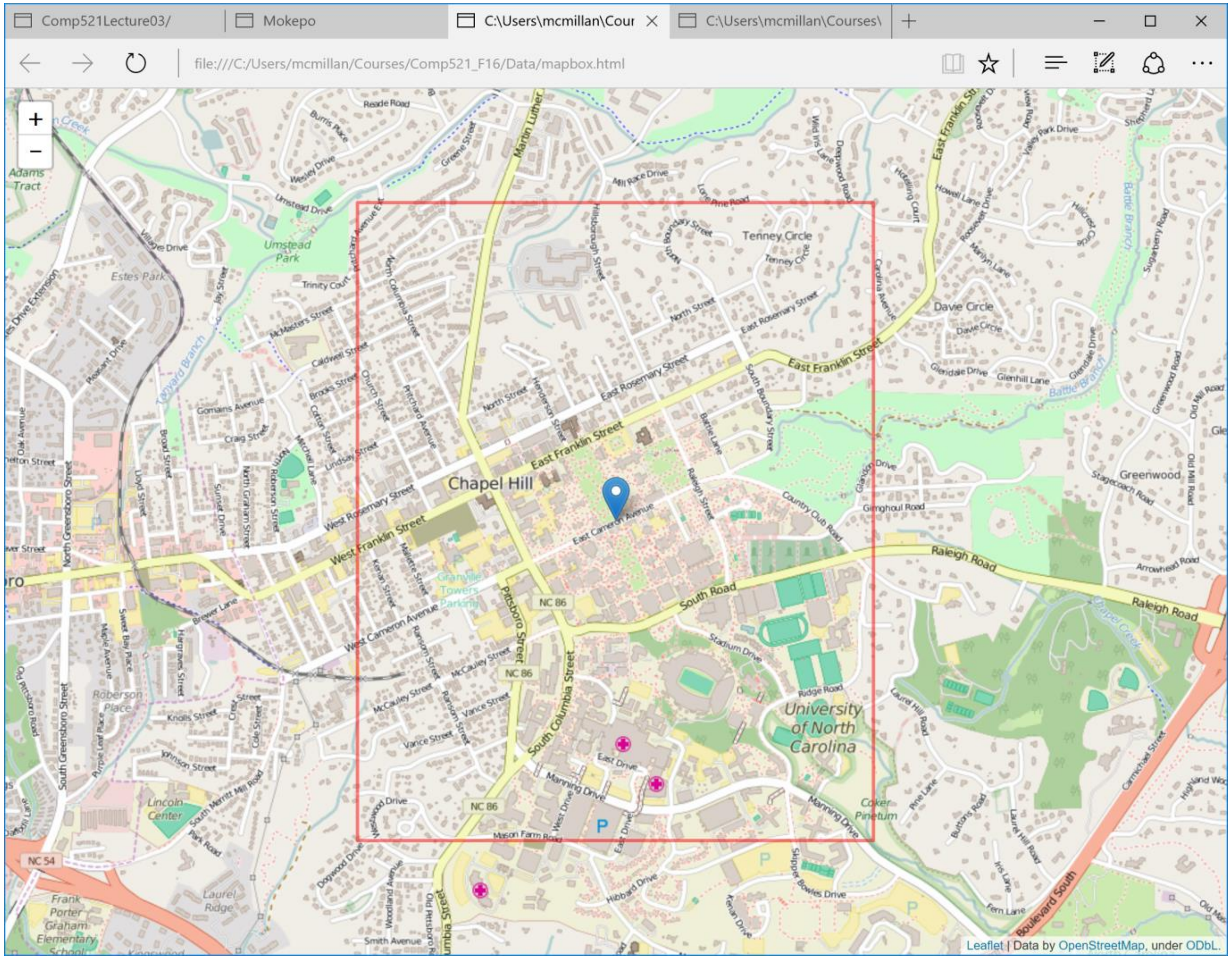

Comp521Lecture03/
Mokepo
localhost:8888/notebooks/Comp521Lecture03/Mokepo.ipynb
jupyter
Mokepo Last Checkpoint: a few seconds ago (autosaved)
File
Edit
View
Insert
Cell
Kernel
Help
Python 2
Markdown
CellToolbar

How many different Monkepo have been seen?

```
In [23]: def monkeCount(dataframe):
             monkeType = {}
             for row in dataframe.itertuples():
                 if row.name not in monkeType:
                     monkeType[row.name] = 0
                 else:
                     monkeType[row.name] += 1
             return monkeType

         %time monkeType = monkeCount(dataframe)
         print len(monkeType), "unique Monkepo"

         Wall time: 255 ms
         138 unique Monkepo

In [24]: def monkeCountV2(dataframe):
             monkeType = {}
             for row in dataframe.itertuples():
                 monkeType[row.name] = monkeType.get(row.name,0) + 1
             return monkeType

         %time monkeType = monkeCountV2(dataframe)
         print len(monkeType), "unique Monkepo"

         Wall time: 245 ms
         138 unique Monkepo
```

```
seenNearby = 0
for row in dataframe.itertuples():
    if (abs(row.latitude - 35.912) < 0.01) and (abs(row.longitude + 79.051) < 0.01):
        seenNearby += 1
```

Know this neighborhood?

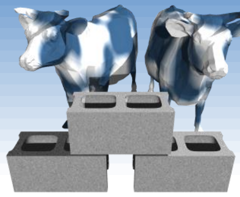

Comp521Lecture03/
Mokepo
C:\Users\mcmillan\Cour
C:\Users\mcmillan\Courses\
file:///C:/Users/mcmillan/Courses/Comp521_F16/Data/mapbox.html
Chapel Hill
East Franklin Street
West Franklin Street
Raleigh Road
South Road
University of North Carolina
Leaflet | Data by OpenStreetMap, under ODbL.

Now a real test…

- How many have distinct Monkepo species appear?
  - Builds a Python dictionary, 'monkeType', whose key is 'name' and 'value' is the number of times it appears in the file.

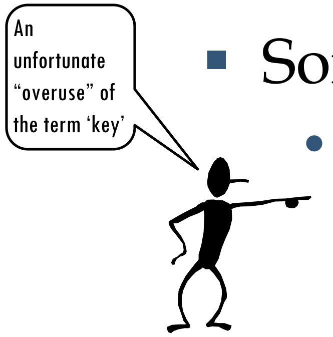
```
def monkeCount(dataframe):
    monkeType = {}
    for row in dataframe.itertuples():
        if row.name not in monkeType:
            monkeType[row.name] = 0
        else:
            monkeType[row.name] += 1
    return monkeType

monkeType = monkeCount(dataframe)
print len(monkeType), "unique Monkepo"
```

```
def monkeCountV2(dataframe):
    monkeType = {}
    for row in dataframe.itertuples():
        monkeType[row.name] = monkeType.get(row.name,0) + 1
    return monkeType

monkeType = monkeCountV2(dataframe)
print len(monkeType), "unique Monkepo"
```



Now which is the most common?

- Use the dictionary from last
  - Sort the 'keys' (names) by the 'values' (counts)
    - In Python, the 'sorted' iterator allows for an optional parameter, 'key' to specify the attribute to sort by, as well as a parameter 'reverse', which controls the order (increasing or decreasing)
    - In Python you can specify the attribute to sort by using a function to select it.
    - Python includes the ability to define simple "anonymous" functions inline using the keyword 'lambda' which takes a list of arguments followed by a colon and a single statement whose value is returned



```
for key, value in sorted(monkeType.items(), key=lambda tup:tup[1], reverse=True):
    print "%10s: %6d" % (key, value)
```

What's the most recent Monkepo?

- Math on dates and times can be tricky
  - Regional differences
  - Discontinuities
  - Variable-sized parts (60 secs/min, 24 hours/day, some months with 30, 31, 29 and 28 days, etc.)
- Python has a nice packages, 'datetime', and 'dateutil' to handle these issues cleanly

```
import datetime
import dateutil

maxdate = datetime.datetime(1970,1,1,0,0,0)
rowIndex = -1
for i, row in dataframe.iterrows():
    mpotime = dateutil.parser.parse("%s %s" % (row.date, row.time))
    if (mpotime > maxdate):
        maxdate = mpotime
        rowIndex = i

print maxdate, rowIndex
print datetime.datetime.now() - maxdate
```

Let's combine ideas

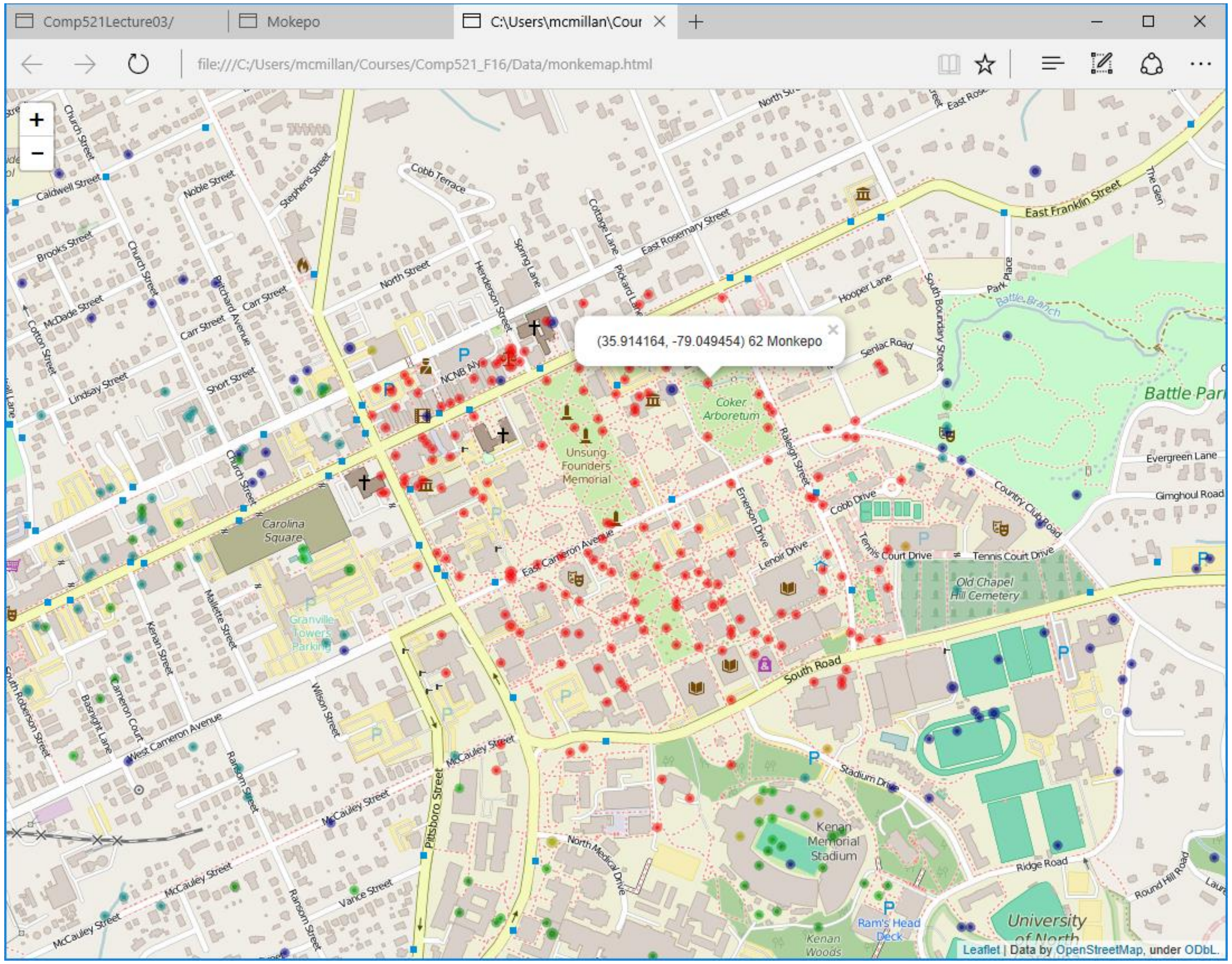
- We know the global frequency of Monkepo, but perhaps it differs locally. In other words, perhaps certain Monkepo are more apt to show up at particular places.
- For example around (35.914164, -79.049454)? (A secluded location near a babbling brook)

```
def monkepoNear(latitude, longitude, range=0.0001):
    monkeSeen = {}
    N = 0
    for row in dataframe.itertuples():
        if (abs(row.latitude - latitude) < range) and (abs(row.longitude - longitude) < range):
            monkeSeen[row.name] = monkeSeen.get(row.name, 0) + 1
            N += 1
    print "Saw", N, "monkepo in region", (latitude, longitude), '+/-', range
    for key, value in sorted(monkeSeen.items(), key=lambda tup:tup[1], reverse=True):
        print "%10s: %6d" % (key, value)
    print

monkepoNear(35.914164,-79.049454)
```

Other interesting questions…

- Do particular species of Monkepo appear at particular times of day?
- Do Monkepo appear anywhere with equal likelihood? Or, might there be hotspots?
- Are there patterns of MonkePo occurrences?
- If one has rough information about the whereabouts of a particular Monkepo, can we figure out where it is?

Do hotspots exist?



Every question requires new code

- Moreover, the various 'codes' fall into a common patterns
  - Scan through the file looking for instances that satisfy some test, and save the results in some other table/list/hash
  - As the file grows, so does the time required to answer our questions
- Rather than 'code', can we devise a way have the computer search through its 'databanks' and we just to ask questions? After all, that's how they worked on Star Trek.

Data Organization

- Some questions are hard resolve in one pass
  - What is the longest interval for which no Monkepo were detected?
  - At point on Campus am I most likely to find a Monkepo? A good Monkepo?
- However, if we reorganized the data they could be answered faster
  - Sort rows by date and time
    (Recall, finding the most recent Monkepo report)
  - Sort rows by their position
  - Sort rows by the frequency of the Monkepo type

Enter Databases

- Rather than devise a new algorithm for any question you might have, devise a "Query Language" and a flexible "Data Organization Scheme" that is easy to scan and search.
- Let the computer "*figure out*" the best method for approaching any given query or question.
- Suppose 1000's of people are adding new sightings to our file, how can that be managed?

Next Time

- The Relational Model