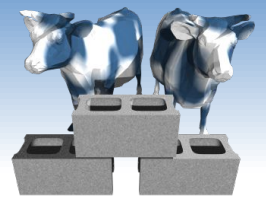
The Trouble with Files

(Hands on)

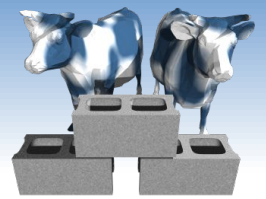
Warning: Today is easy.
Mostly cut-and-paste. But, it is just a warm up for things to come. YOU WILL WRITE CODE ***IN*** this class.



Only a few days to go!

- Only 9 more days until the season starts!
- Time to draft a Fantasy Football team!

- Open questions:
  - What is American/ NFL football?
  - Can't I just pick New England and be done with it?
  - What is fantasy about it?
  - What has this got to do with databases?



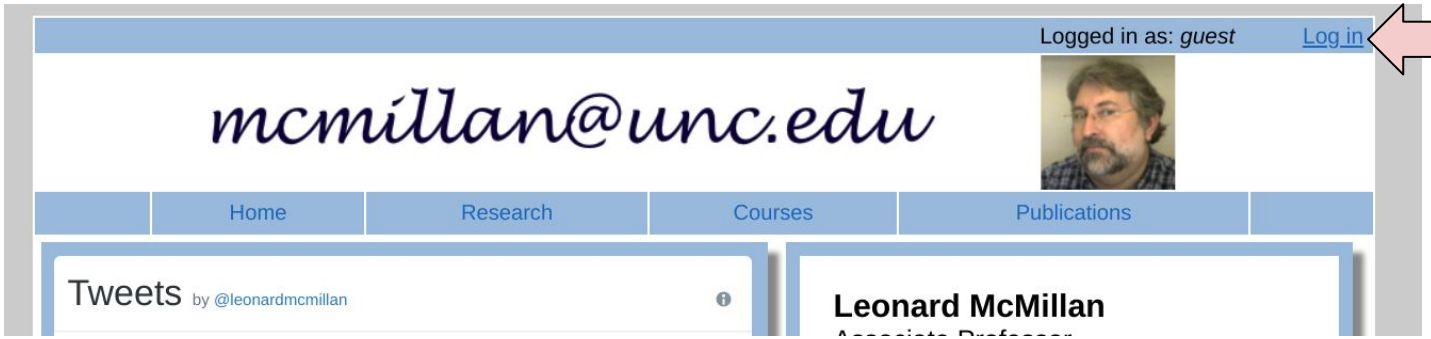
Let's login

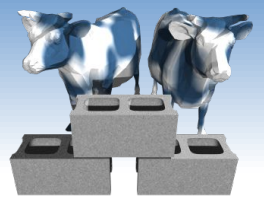
If you were here last Thursday, you should have

- A course website login
- A Jupyter Hub login

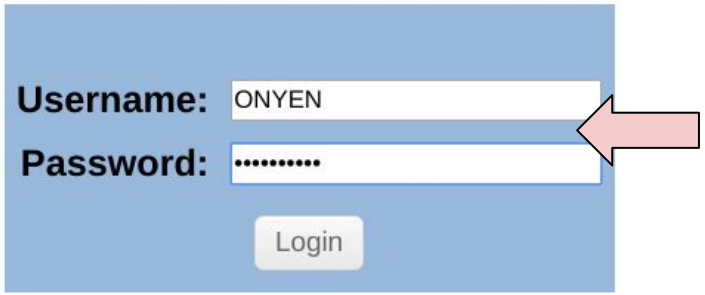
Let's try each.

First goto https://csbio.unc.edu/mcmillan/





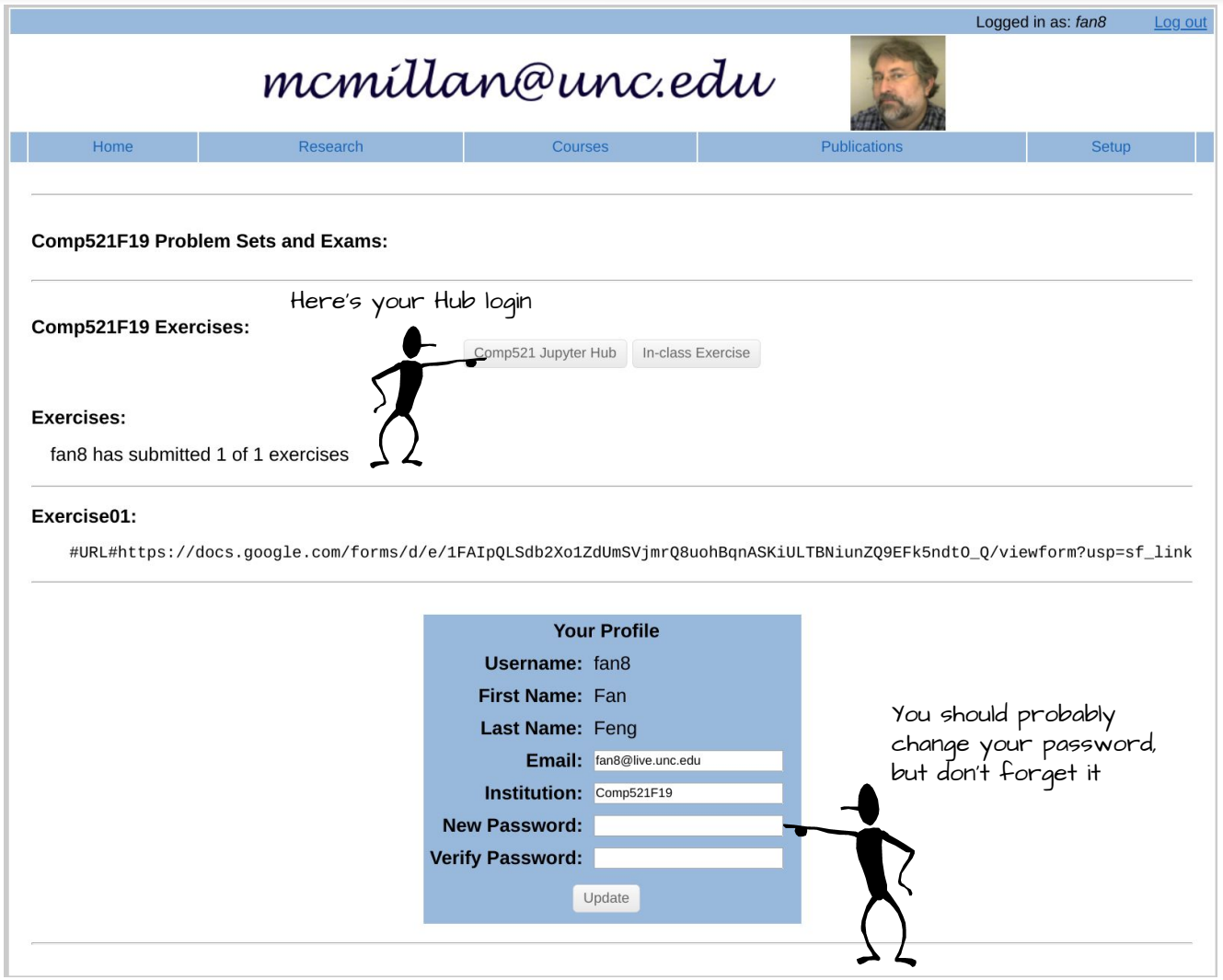
Course website login

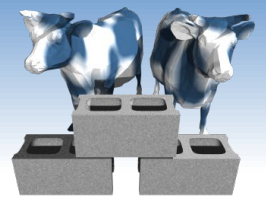


Enter your ONYEN as your username, and your PID as your password

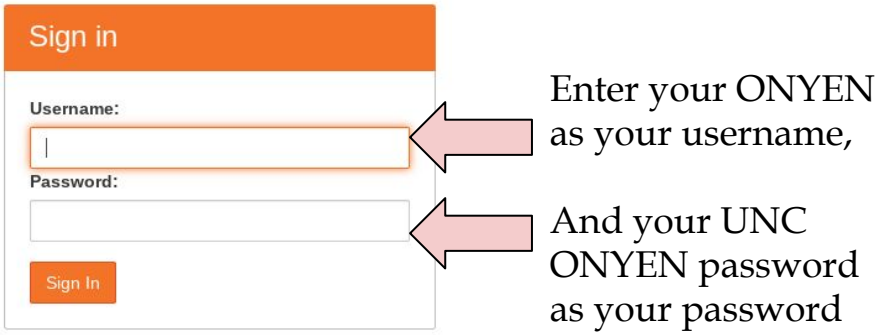
Your login should then show up as "Verified"
Next press "Continue"; you should then see "Setup" as a menu option. Press it.

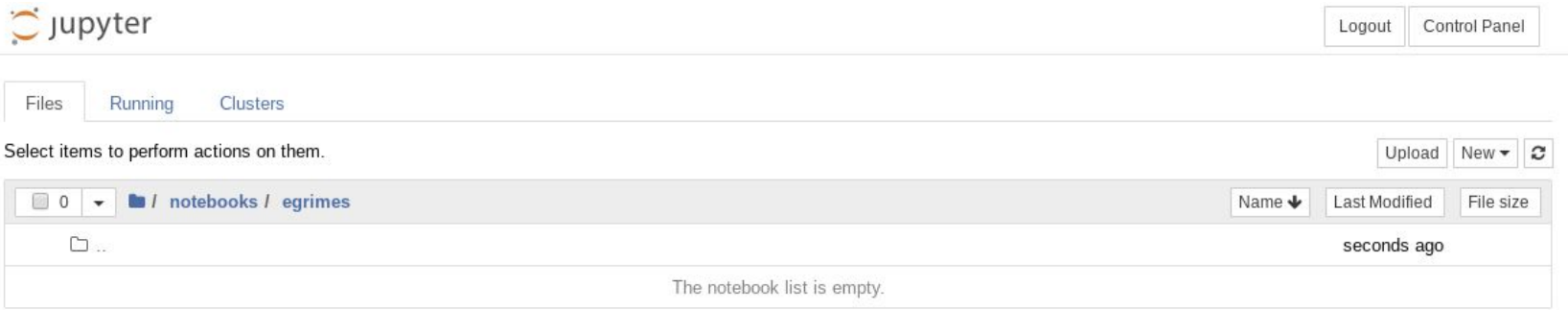
Course website portal



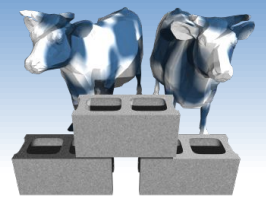


Your Jupyter Hub login





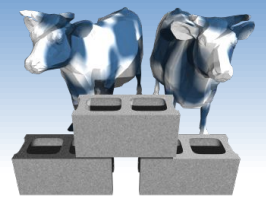
This should lead you to a place that resembles what is shown above



Now, let's look at data

The team rosters for the last 10 years can be downloaded from: http://csbio.unc.edu/mcmillan/Media/NFLRosters.tsv

- Save it to the Downloads folder on your machine
- Then upload it to your Jupyter hub
- You can also open it in a spreadsheet, or "head", "tail", or "more" it using the hub terminal.

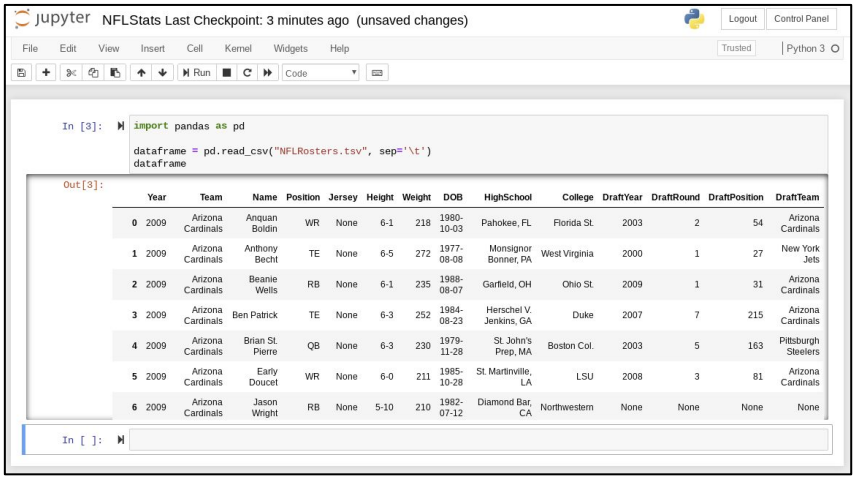


Read in the file

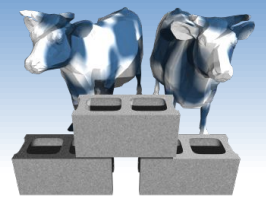
- Make a new Python3 notebook
- Rename it "NFLStats"
- Add 3 lines of code into a cell, and run it!

```
import pandas as pd

dataframe = pd.read_csv("NFLRosters.tsv", sep='\t')
dataframe
```

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



	Year	Team	Name	Position	Jersey	Height	Weight	DOB	HighSchool	College	DraftYear	DraftRound	DraftPosition	DraftTeam
0	2009	Arizona Cardinals	Anquan Boldin	WR	None	6-1	218	1980-10-03	Pahokee, FL	Florida St.	2003	2	54	Arizona Cardinals
1	2009	Arizona Cardinals	Anthony Becht	TE	None	6-5	272	1977-08-08	Monsignor Bonner, PA	West Virginia	2000	1	27	New York Jets
2	2009	Arizona Cardinals	Beanie Wells	RB	None	6-1	235	1988-08-07	Garfield, OH	Ohio St.	2009	1	31	Arizona Cardinals
3	2009	Arizona Cardinals	Ben Patrick	TE	None	6-3	252	1984-08-23	Herschel V. Jenkins, GA	Duke	2007	7	215	Arizona Cardinals
4	2009	Arizona Cardinals	Brian St. Pierre	QB	None	6-3	230	1979-11-28	St. John's Prep, MA	Boston Col.	2003	5	163	Pittsburgh Steelers
5	2009	Arizona Cardinals	Early Doucet	WR	None	6-0	211	1985-10-28	St. Martinville, LA	LSU	2008	3	81	Arizona Cardinals
6	2009	Arizona Cardinals	Jason Wright	RB	None	5-10	210	1982-07-12	Diamond Bar, CA	Northwestern	None	None	None	None



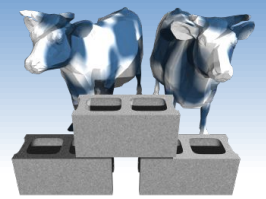
Jumping into the data

- Let's look at this dataset.
- An example of scanning through a dataframe

```
quarterbacks = 0
for index, row in dataframe.iterrows():  # index is a number, row is a dictionary
    if (row["Position"] == "QB"):
        quarterbacks += 1
print(quarterbacks)
```

- One way to programically keep track of and organize things is to use dictionaries.
- Python dictionaries are stores for "*name-value*" pairs, where a hash is used to disambiguate names

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

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



Counting distinct occurences

Scan a given column name and count the number of distinct entries

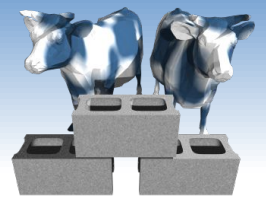
Scan the dataframe while counting the number of distinct values in a given column

In [48]:
```
def CountOccurences(df, field):
    """ Return a dictionary of distinct field entries with their number of occurences"""
    fieldCounts = {}
    for index, row in df.iterrows():
        if row[field] not in fieldCounts:
            fieldCounts[row[field]] = 0
        fieldCounts[row[field]] += 1
    return fieldCounts
```

In [ ]:
```
teams = CountOccurences(dataframe, 'Team')

for name, count in sorted(teams.items()):
    print(name, count)
```

In [ ]:
```
players = CountOccurences(dataframe, 'Name')

N = 0
for name, count in players.items():
    if (count >= 10):
        print(name, count)
        N += 1
print(N, 'players appear in ten or more rows')
```

What's going on with this data?



```
Arizona Cardinals 234
Atlanta Falcons 201
Baltimore Ravens 206
Buffalo Bills 233
Carolina Panthers 200
Chicago Bears 195
Cincinnati Bengals 205
Cleveland Browns 216
Dallas Cowboys 196
Denver Broncos 206
Detroit Lions 215
Green Bay Packers 219
Houston Texans 212
Indianapolis Colts 217
Jacksonville Jaguars 237
Kansas City Chiefs 217
Los Angeles Chargers 77
Los Angeles Rams 92
Miami Dolphins 204
Minnesota Vikings 196
New England Patriots 211
New Orleans Saints 211
New York Giants 221
New York Jets 228
Oakland Raiders 227
Philadelphia Eagles 206
Pittsburgh Steelers 202
Saint Louis Rams 115
San Deigo Chargers 130
San Francisco 49ers 218
Seattle Seahawks 212
Tampa Bay Buccaneers 216
Tennessee Titans 208
Washington Redskins 228
```

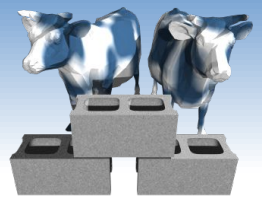
- Patterns here?
- What's the deal with New York and Los Angeles?
- Are Arizona, Carolina, Minnesota, and New England cities?
- Where do Chargers and Rams live?
- Difference between a team and a franchise?

Let's look another one

```
Larry Fitzgerald 10
Matt Ryan 10
Joe Flacco 10
Ryan Fitzpatrick 10
Jonathan Stewart 10
Steve Smith 12
Adrian Peterson 11
Greg Olsen 10
Brandon Marshall 11
Matthew Stafford 10
Aaron Rodgers 10
Pierre Garcon 10
Marcedes Lewis 10
Zach Miller 12
Jamaal Charles 10
Matt Cassel 10
Ted Ginn 10
Brian Hoyer 10
Tom Brady 10
Drew Brees 10
Eli Manning 10
Darrius Heyward-Bey 10
Alex Smith 16
DeSean Jackson 10
LeSean McCoy 10
Ben Roethlisberger 10
David Johnson 11
Mike Wallace 10
Danny Amendola 10
Antonio Gates 10
Darren Sproles 10
Philip Rivers 10
Delanie Walker 10
Frank Gore 10
Michael Crabtree 10
Vernon Davis 10
Jared Cook 10
Demaryius Thomas 10
Golden Tate 10
39 players appear in ten or more rows
```

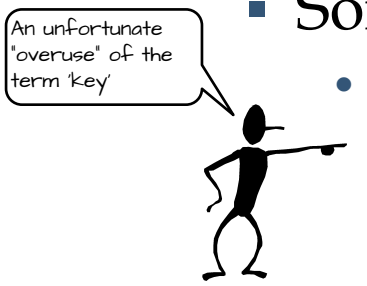
- How many players appeared in 10 or more rosters?
- Can we be sure there aren't two players with the same name? (Alex Smith)

- How do we disambiguate? name+team, name+birthday, name+college
- Is the roster data wrong?

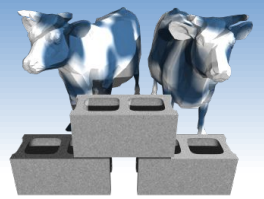


Now which is the most common?

- Using 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



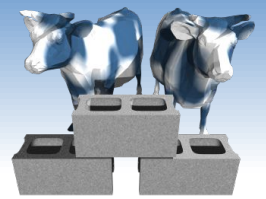

```
for key, value in sorted(players.items(), key = lambda playerCount: playerCount[1],
reverse=True):
    if (value <= 10):
        break
    print("%20s: %6d" % (key, value))
```



Exercise Time

- How many distinct "Positions" appear in the NFLRoster list
  ('QB', 'RB', 'WR', etc)

  - Are there ambiguities?
  - Inconsistencies?
  - What is the difference?

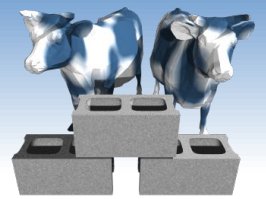


How many #1 draft picks?

- How many appear in our roster?
  When and where do they play?

```
NumberOnes = {}
for index, row in dataframe.iterrows():
    if (row["DraftRound"] == "1") and (row["DraftPosition"] == "1"):
        pick = (row['DraftYear'],row['Name'],row['Position'])
        NumberOnes[pick] = NumberOnes.get(pick,[]) + [(row['Year'],row['Team'])]
for year, player, position in sorted(NumberOnes):
    print(player, year, position)
    for year, team in NumberOnes[year,player,position]:
        print("    ", year, team)
```

- What's up with 2005? "Alex Smith" again?
- Any other fishy results?
- Many subtle bugs arise from "slight" misunderstanding of the underlying data



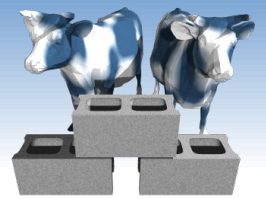
How to separate Alex Smiths?

- We could make sure our player counts consider the combination of name and birthdates
  (Actually, will not work here because the birthdates were added to the given Rosters based on the names)

```
In [7]: !head Data/team_2009_rosters.csv
```

```
Season,Player,Team,Pos,name,GSIS_ID
2009,Aaron Rodgers,GB,QB,A.Rodgers,00-0023459
2009,Alex Smith,SF,QB,A.Smith,00-0023436
2009,Ben Roethlisberger,PIT,QB,B.Roethlisberger,00-0022924
2009,Billy Volek,SD,QB,B.Volek,00-0019041
2009,Brady Quinn,CLE,QB,B.Quinn,00-0025409
2009,Brett Favre,MIN,QB,B.Favre,00-0005106
2009,Brian Brohm,BUF,QB,B.Brohm,00-0026196
2009,Brian Hoyer,NE,QB,B.Hoyer,00-0026625
2009,Brian St. Pierre,ARI,QB,B.Pierre,00-0022101
```

- Player details were appended to the original "sparse" roster file. Perhaps, position should have been considered at this point (It wasn't :( )
  Maybe the GSIS_ID would have helped?

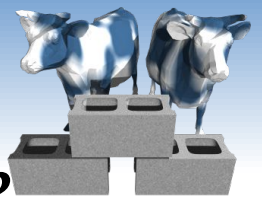


Let's combine ideas

- What Quarterbacks have played on multiple teams?
- Whats going on the the ".split(' ')[-1]" on "Team"
- What is a set() and why and how is it being used?

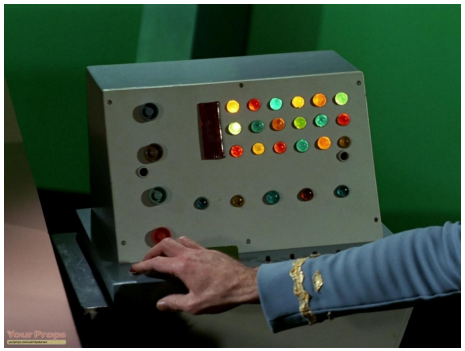
```
quarterbacks = {}
for index, row in dataframe.iterrows():
    if (row['Position'] == 'QB'):
        quarterbacks[row['Name']] = quarterbacks.get(row['Name'],[]) + [row['Team'].split(' ')[-1]]

print(len(quarterbacks), "Quarterbacks")

N = 1
for name in quarterbacks:
    different = set(quarterbacks[name])
    if (len(different) > 1):
        print("%3d %20s: %s" % (N, name, different))
        N += 1
```

- Can it be done in a single scan?

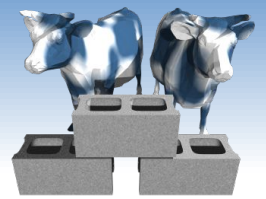


Every question requires new code

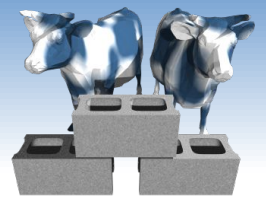
- Moreover, the various 'codes' fall into a common patterns
  - Scan through the file looking for instances that satisfy some test, and save the results in some other table/list/hash
  - As the file grows, so does the time required to answer our questions
- Rather than write 'code', can we devise a way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



Data Organization Matters

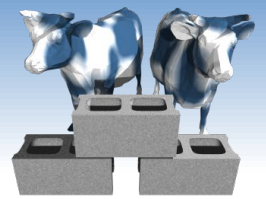


- Some questions are hard resolve in one pass
  - Has anyone every changed playing positions in one roster relative to another?
  - What Colleges are best represented in the NFL?
  - Are College names unambigous?
    ("Miami", "Miami (Ohio)")
  - Are College names consistent?
    ("Penn St." or "Pennsylvania State University")
- If we reorganized the data could questions be answered faster
  - Sort rows by Year and Team
  - Sort rows by Position and Team



Enter Databases

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



Next Time

- The Relational Model