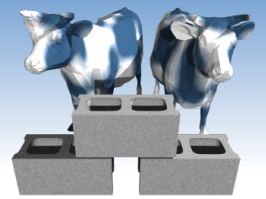
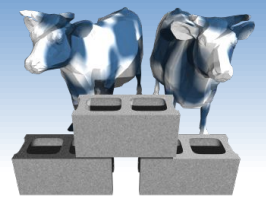
The Trouble with Files

(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.



After only 2 days…

- It looks like we're back to the ultimate in social distancing…. Zoom
- Two qestions have defined our year thus far:
  - Can Americans save themselves from two diseases?
  - A virus that propagates via non-symptomatic carriers
  - A legacy of systematic racism
- Let's look at the data



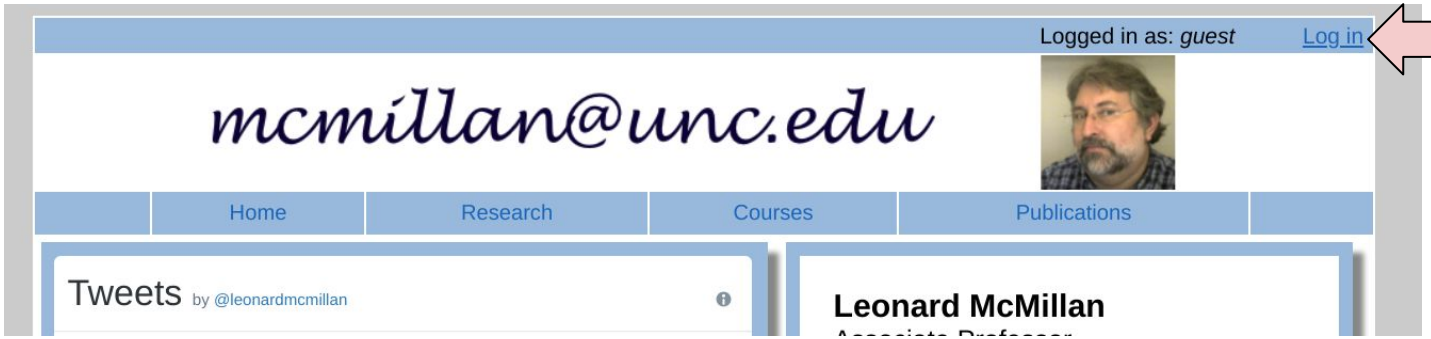
Let's login

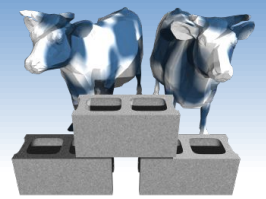
If you were here last Thursday, you should have

- A course website login
- A Jupyter Hub login

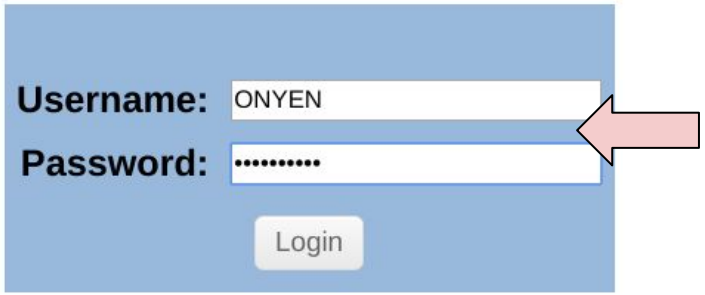
Let's try each.

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





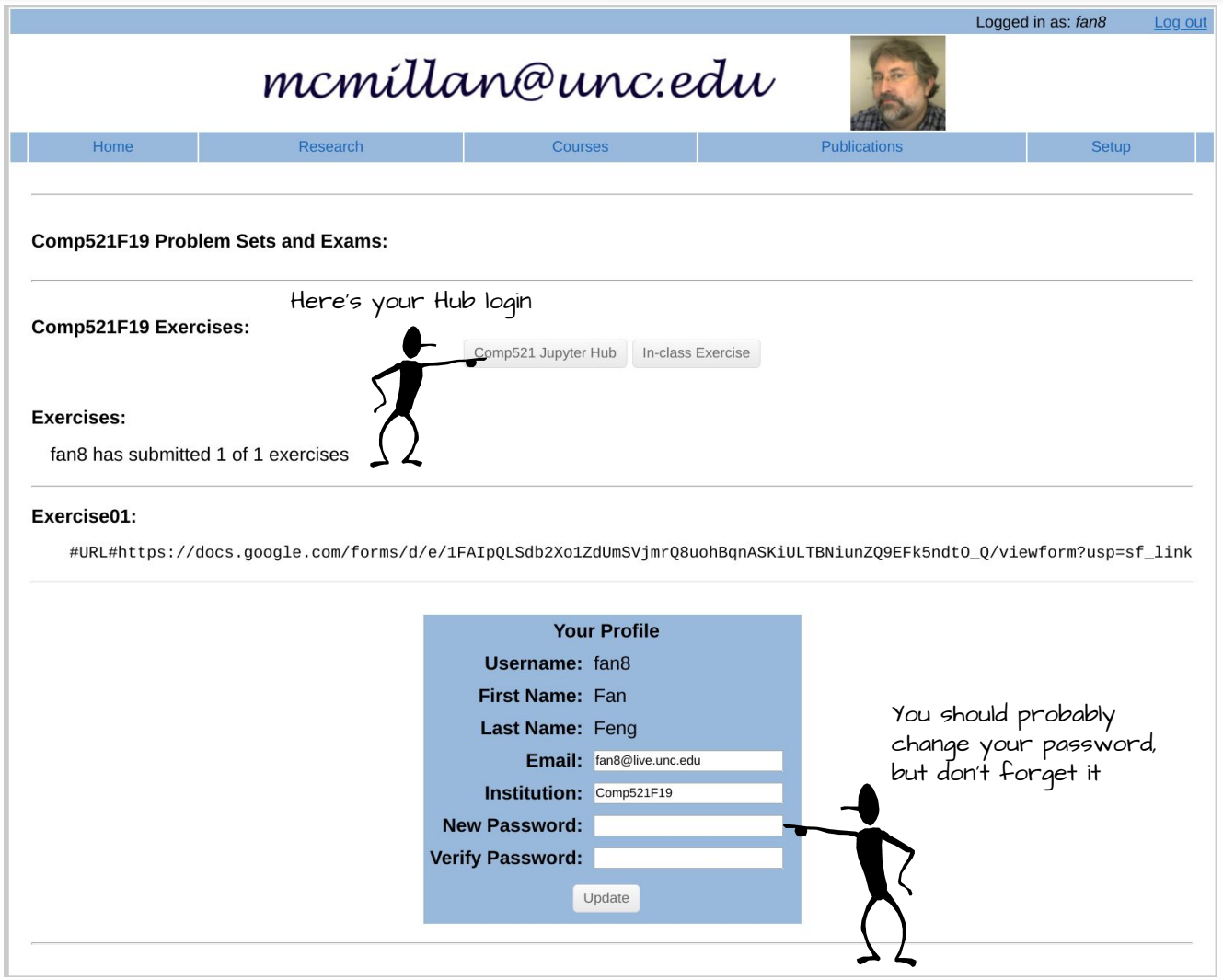
Course website login

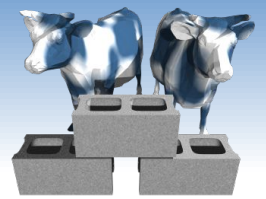


Enter your ONYEN as your username, and your PID as your password

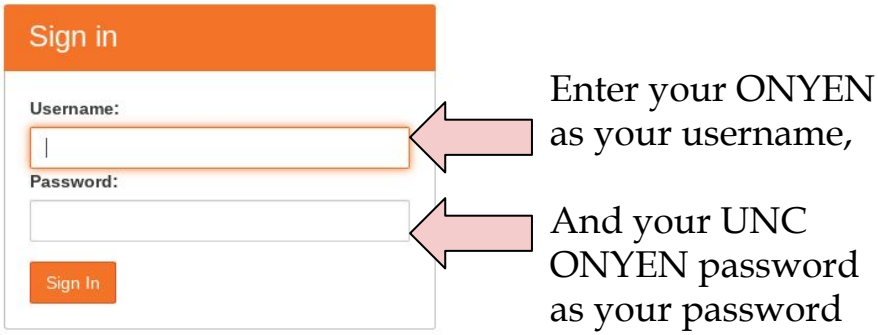
Your login should then show up as "Verified"
Next press "Continue"; you should then see "Setup" as a menu option. Press it.

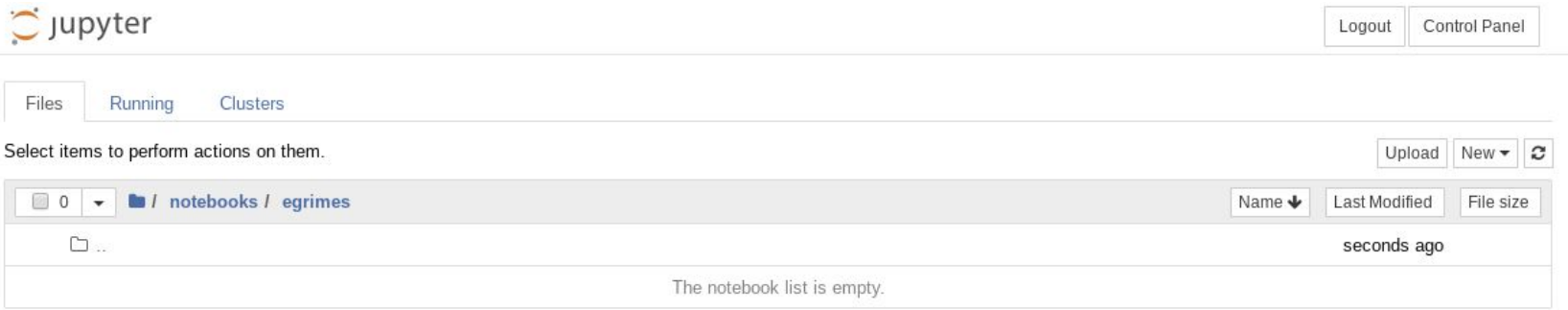
Course website portal



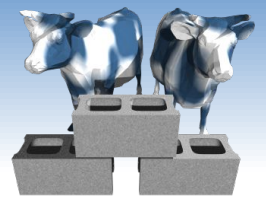


Your Jupyter Hub login





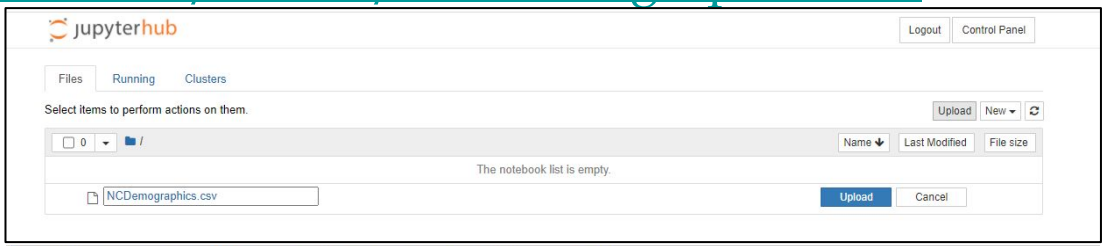
This should lead you to a place that resembles what is shown above



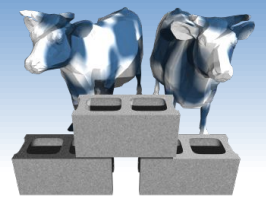
Now, let's look at data

The population makeup of all NC counties over the past 10 years can be downloaded from:
http://csbio.unc.edu/mcmillan/Media/NCDemographics.csv

- 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.




```
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

mcmillan@comp521-2fa20:~$ pwd
/home/mcmillan
mcmillan@comp521-2fa20:~$ ls
NCDemographics.csv
mcmillan@comp521-2fa20:~$ head NCDemographics.csv
fips,County,Region,COG,MSA,Year,Race,Sex,Age 0 to 2,Age 3 to 4,Age 5,Age 6 to 9,Age 10 to 13,Age 14,Age 15,Age 16 to 17,Age
 18 to 19,Age 20 to 24,Age 25 to 34,Age 35 to 44,Age 45 to 54,Age 55 to 59,Age 60 to 64,Age 65 to 74,Age 75 to 84,Age 85 to
 99,Age 100
37045,Cleveland,Western North Carolina,Isothermal Planning and Development Commission,Non-Metropolitan,2000,aian,female,,,,
,,,,,,,,,,,,,,
37057,Davidson,Central North Carolina,Piedmont Triad Council of Governments,Winston-Salem,2000,aian,female,14,8,6,19,19,4,6
,11,6,27,54,57,45,14,5,9,1,2,0
37087,Haywood,Western North Carolina,Southwestern Commission,Asheville,2000,aian,female,,,,,,,,,,,,,,,,,,
37095,Hyde,Eastern North Carolina,Albermarle Commission,Non-Metropolitan,2000,aian,female,,,,,,,,,,,,,,,,,,
37109,Lincoln,Central North Carolina,Centralina Council of Governments,Charlotte-Concord-Gastonia (NC),2000,aian,female,,,,
,,,,,,,,,,,,,,
37113,Macon,Western North Carolina,Southwestern Commission,Non-Metropolitan,2000,aian,female,,,,,,,,,,,,,,,,,,
37117,Martin,Eastern North Carolina,Mid-East Commission,Non-Metropolitan,2000,aian,female,,,,,,,,,,,,,,,,,,
37123,Montgomery,Central North Carolina,Piedmont Triad Council of Governments,Non-Metropolitan,2000,aian,female,,,,,,,,,,,,
,,,,,,,
37125,Moore,Central North Carolina,Triangle J Council of Governments,Non-Metropolitan,2000,aian,female,12,10,8,11,8,1,7,9,5
,20,52,40,37,14,12,12,7,2,0
mcmillan@comp521-2fa20:~$
```



Read in the file

- Make a new Python3 notebook
- Rename it "NCStats"
- Add the following lines of code, and run it!

```
import pandas as pd

def emptyInts(value):
    ''' Convert string values to ints, treat empty strings as 0 '''
    return 0 if value == '' else int(value)

dataframe = pd.read_csv("NCDemographics.csv", converters={colIndex : emptyInts for colIndex in range(8,28)})
print(dataframe.shape)
dataframe
```

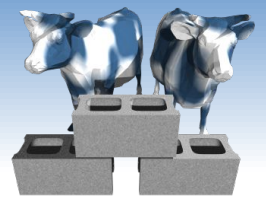
- Scroll around.
- Get a sense for what the data looks like.

In [1]:
```
import pandas as pd

def emptyInts(value):
    ''' Convert string values to ints, treat empty strings as 0 '''
    return 0 if value == '' else int(value)

dataframe = pd.read_csv("NCDemographics.csv", converters={colIndex : emptyInts for colIndex in range(8,28)})
print(dataframe.shape)
dataframe
```

```
(25200, 27)
```

Out[1]:

	fips	County	Region	COG	MSA	Year	Race	Sex	Age 0 to 2	Age 3 to 4	...	Age 20 to 24	Age 25 to 34	Age 35 to 44	Age 45 to 54	Age 55 to 59	Age 60 to 64	Age 65 to 74	Age 75 to 84	Age 85 to 99
0	37045	Cleveland	Western North Carolina	Isothermal Planning and Development Commission	Non-Metropolitan	2000	aian	female	0	0	...	0	0	0	0	0	0	0	0	0
1	37057	Davidson	Central North Carolina	Piedmont Triad Council of Governments	Winston-Salem	2000	aian	female	14	8	...	27	54	57	45	14	5	9	1	2
2	37087	Haywood	Western North Carolina	Southwestern Commission	Asheville	2000	aian	female	0	0	...	0	0	0	0	0	0	0	0	0
3	37095	Hyde	Eastern North Carolina	Albermarle Commission	Non-Metropolitan	2000	aian	female	0	0	...	0	0	0	0	0	0	0	0	0
4	37109	Lincoln	Central North Carolina	Centralina Council of Governments	Charlotte-Concord-Gastonia (NC)	2000	aian	female	0	0	...	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
25195	37165	Scotland	Eastern North Carolina	Lumber River Council of Governments	Non-Metropolitan	2020	Total	female	709	461	...	1116	2409	1886	2185	1216	1218	2267	1126	481



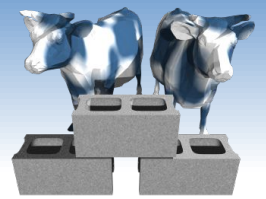
Jumping into the data

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

```
countySeen = set()
for index, row in dataframe.iterrows():  # index is a number, row is a dictionary
    if (row["County"] not in countySeen):
        countySeen.add(row["County"])
print(len(countySeen), "Counties")
```

- 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)
```



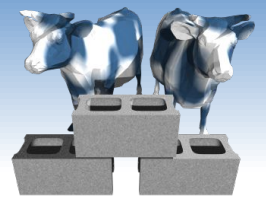
Counting distinct occurences

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

```
def CountOccurences(df, field):
    """ Return a dictionary of distinct field entries with their numbe 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
```

```
years = CountOccurences(dataframe, 'Region')

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

```
Central North Carolina 8316
Eastern North Carolina 10332
Western North Carolina 6552
```

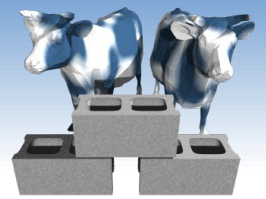


Operations over rows

- Total population of each County in 2020
  - Find rows where year is 2020
  - Sum counts across all age groups
  - Ignore Race, and Gender
- Output those counties with a population > 300,000

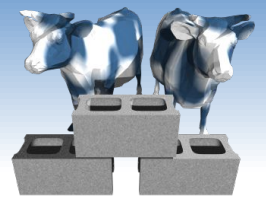
```
CountyPopulation = {}
for index, row in dataframe.iterrows():
    if row['Year'] != 2020:
        continue
    total = sum(row[8:])
    CountyPopulation[row['County']] = CountyPopulation.get(row['County'], 0) + total

for County, Pop in sorted(CountyPopulation.items()):
    if (Pop > 300000):
        print(County, Pop)
```

What's going on with this data?



```
Alamance 348110
Buncombe 534092
Cabarrus 433216
Catawba 321008
Cumberland 666418
Davidson 341776
Durham 640644
Forsyth 766246
Gaston 447684
Guilford 1090696
Iredell 368046
Johnston 424802
Mecklenburg 2262684
New Hanover 478544
Onslow 408714
Pitt 362010
Union 485314
Wake 2219766
```

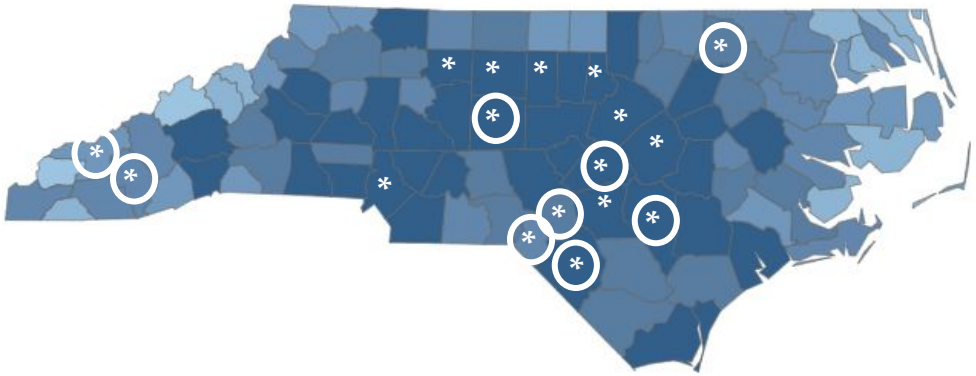
- Any surprizes?
- What do you expect to find in the MSA field in each of these counties?
- How do these populations break down by
  - Sex
  - Race
  - Age
- How might this list differ if the year was 2000?



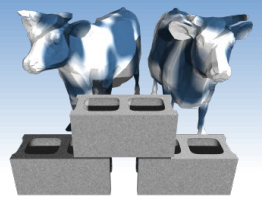
Let's look another one

```
Alamance 3706
Cumberland 5813
Durham 4604
Forsyth 4617
Guilford 4784
Halifax 2325
Harnett 3479
Hoke 5475
Jackson 4276
Johnston 2441
Mecklenburg 11401
Randolph 2251
Robeson 52518
Sampson 2843
Scotland 4973
Swain 3672
Wake 12868
```

- Which counties have an "aian" population > 2000?

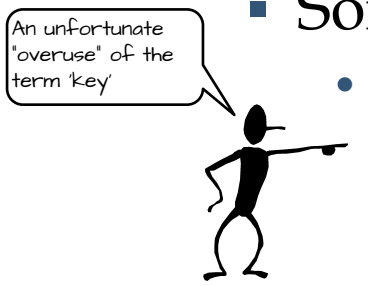


- Which of these aren't in the high population counties
- Which of these aren't high population counties?
- What's with the shades of blue?

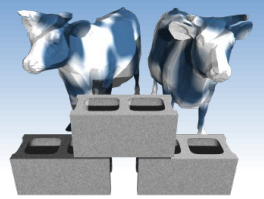


Now which is the most common?

- 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

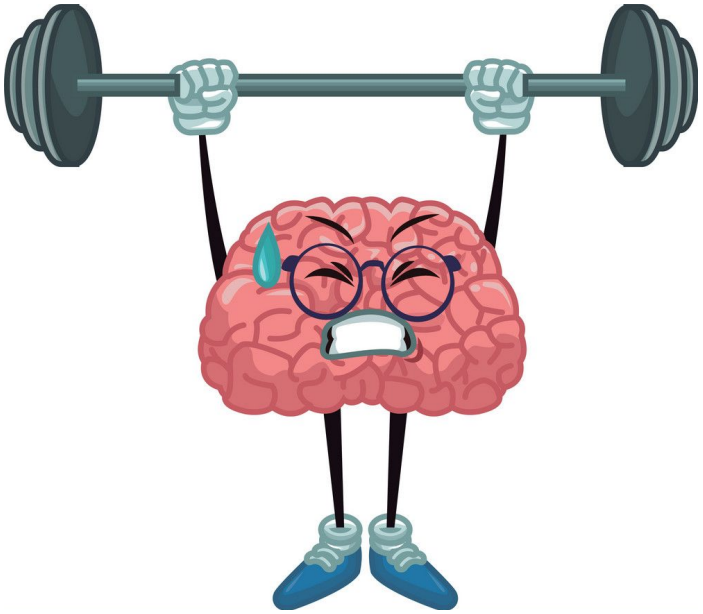


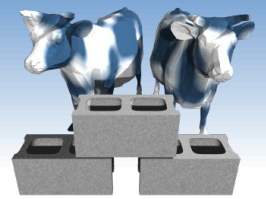

```
for key, value in sorted(AianPopulation.items(), key = lambda AianPopulation: AianPopulation[1], reverse=True):
    if (value <= 10):
        break
    print("%20s: %6d" % (key, value))
```



Exercise Time

- How many distinct "Races" appear in the NCDemographics list?





Majorty non-white counties?

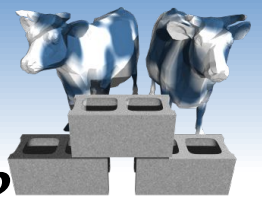
- In how many NC counties in the racial make-up non majoity 'white'?

```
RacePopulation = {}
for index, row in dataframe.iterrows():
    if row['Year'] != 2020 or row['Race'] == "Total":
        continue
    total = sum(row[8:])
    CountyStats = RacePopulation.get(row['County'], {})
    CountyStats[row['Race']] = CountyStats.get(row['Race'],0) + total
    RacePopulation[row['County']] = CountyStats
```

- Then find the ratio

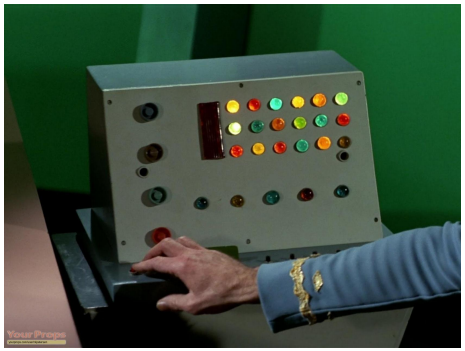
```
for county in sorted(RacePopulation):
    ratio = RacePopulation[county]['white']/sum(RacePopulation[county].values())
    if ratio < 0.5:
        print(county, "%4.2f" % (100.0*ratio))
```

```
Anson 47.75
Bertie 36.90
Edgecombe 39.97
Halifax 39.16
Hertford 38.49
Northampton 40.25
Robeson 31.07
Scotland 43.82
Vance 44.69
Warren 40.87
Washington 46.17
```

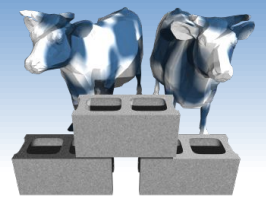


Every question requires new code

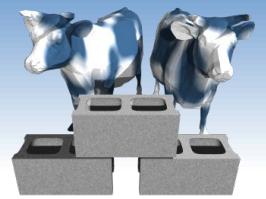
- Moreover, the various 'codes' fall into a common patterns
  - Scan through the file looking for instances that satisfy some test, and save the results in some other table/list/hash
  - As the file grows, so does the time required to answer our questions
- Rather than write 'code', can we devise a way 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

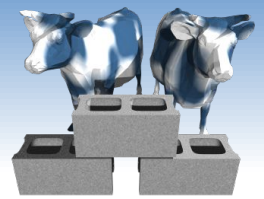


- Some questions are hard resolve in one pass
  - What county has the greatest rate of growth?
  - Which counties have the highest percentage of residents who are under 25?
  - What's the best predictor of COVID infection rate?
    - Population?
    - Population make up?
    - Age make up?
- If we reorganized the data could these anf other questions be answered faster?



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