CS102: Big Data

Tools and Techniques, Discoveries and Pitfalls

Spring 2017

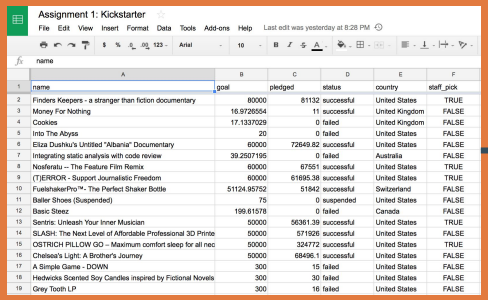
Ethan Chan, Lisa Wang

Lecture 5: Introduction to Python

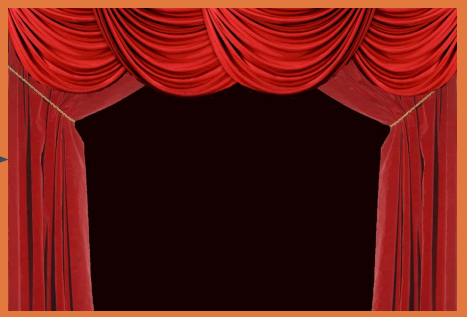
Announcements

- Lisa's OH this week will be on Sunday, 3-4.30pm in Huang Basement (Lathrop is closed on Sundays)

Your CS102 Journey so far:

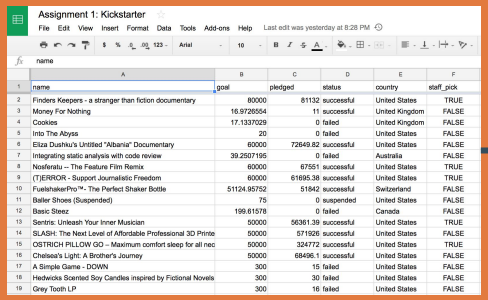



```
Select city
From CityTemps C1
Where Not Exists (Select *
                  from
                  CityTemps C2
                  where C1.lat > C2.lat)
```

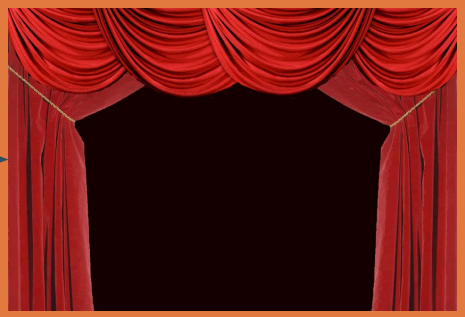


Flexibility

Your CS102 Journey so far:

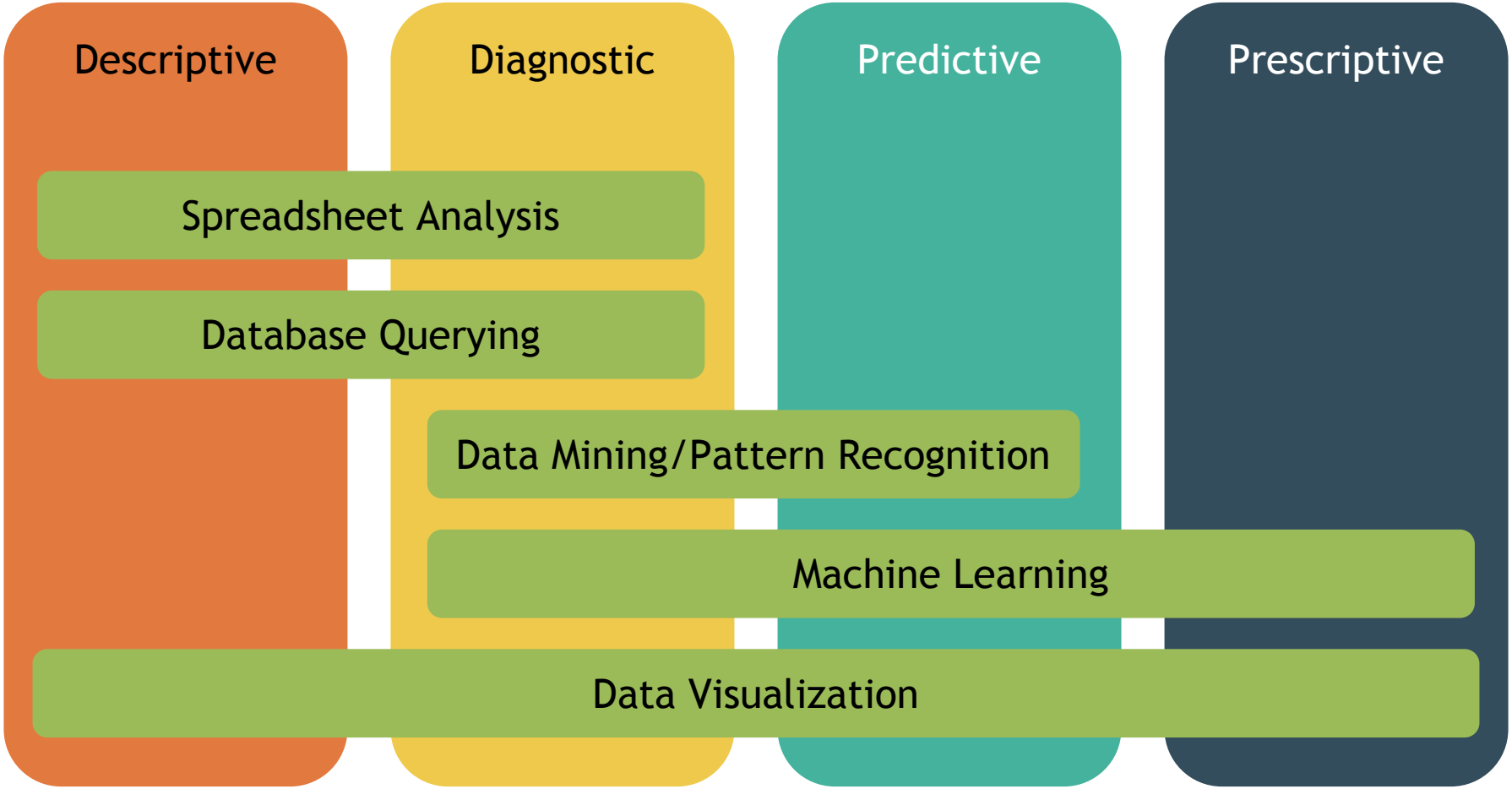



```
Select city
From CityTemps C1
Where Not Exists (Select *
                  from
                  CityTemps C2
                  where C1.lat > C2.lat)
```

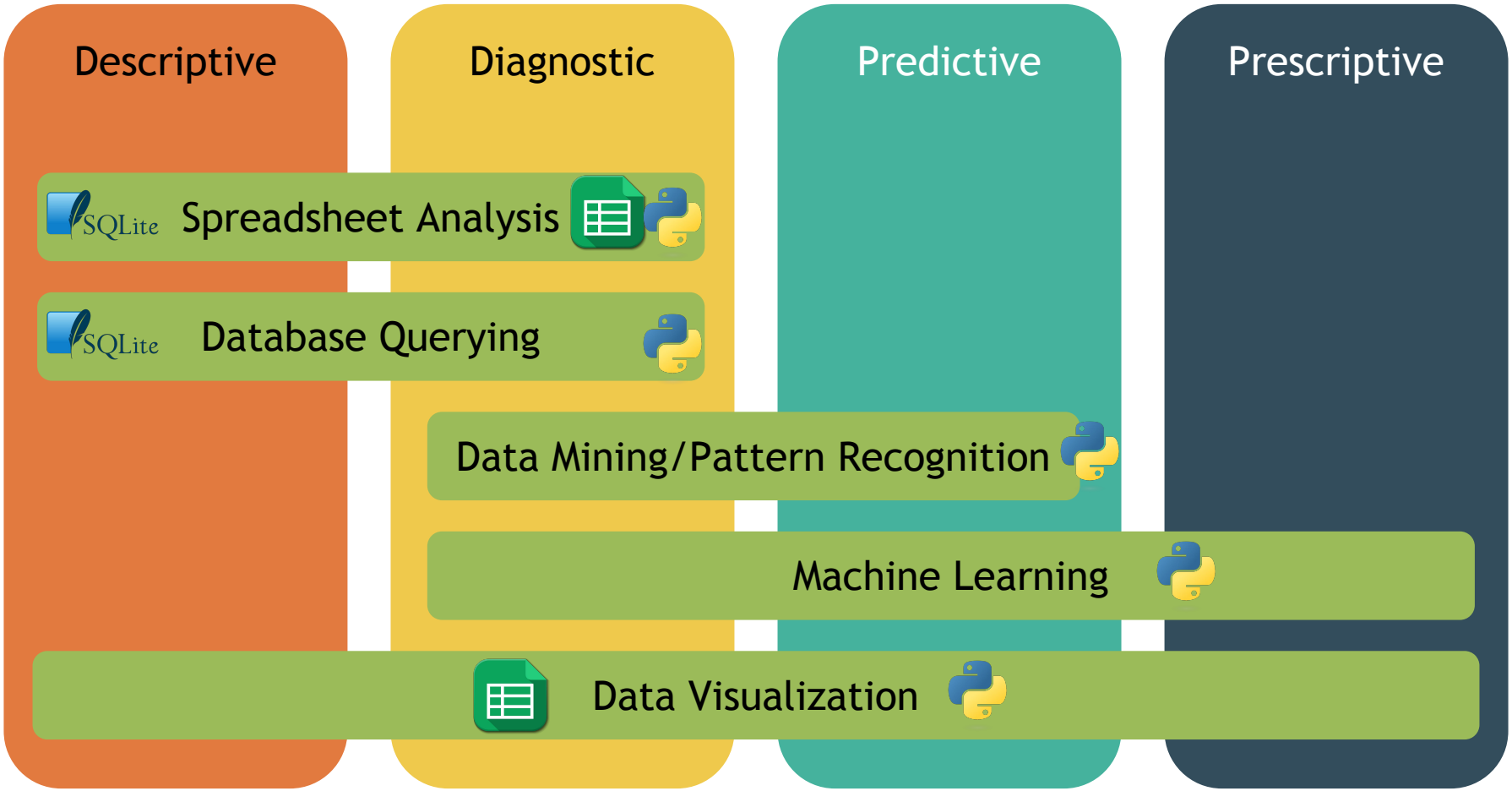


Badass Level ;)

Tools & Techniques



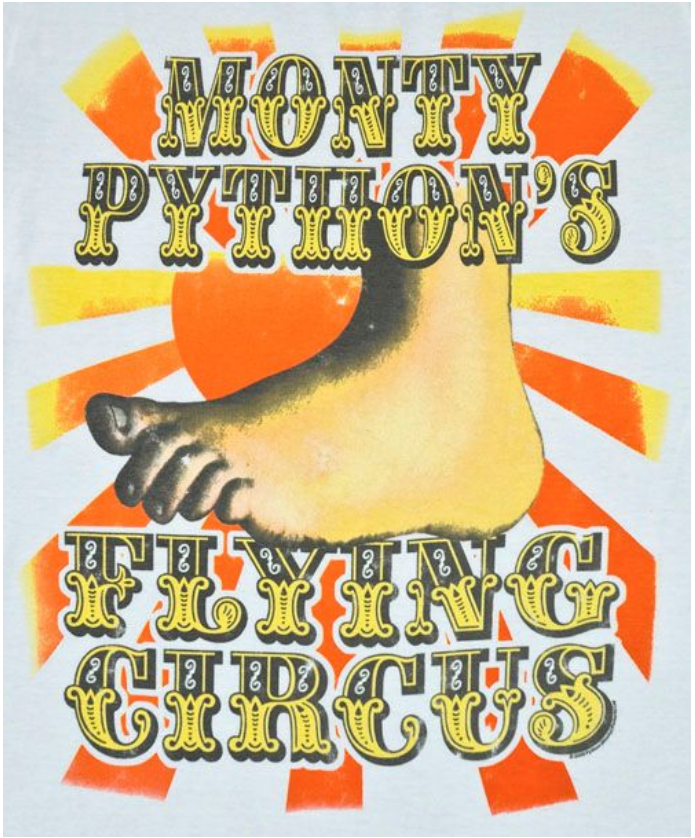
Tools & Techniques



Welcome to Python!

A Short and Sweet History

- Invented by Guido van Rossum in late 80s as a scripting language
- First public release in 1991
- Open-sourced from the beginning
- Named after “Monty Python”, a British comedy series

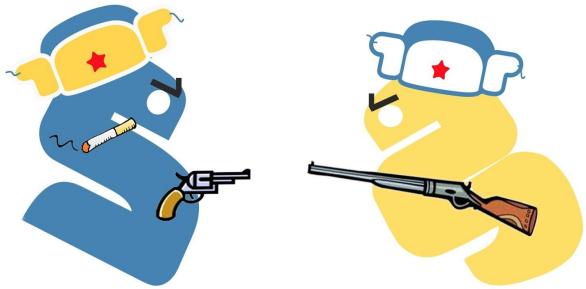


The true origin of Python: An excerpt from Monty Python's Flying Circus.

A Short and Sweet History

- 2000: Release of Python 2.0
  - 2.7 most popular
  - Legacy, but has more libraries
- 2008: Release of Python 3.0
  - backwards incompatible (so code written in 2.x won't work with 3.y)
  - "Future" of Python
  - Slowly getting adopted

➢ One of the most popular languages in use today



Who uses Python?

- Used heavily in both academia and industry
- E.g. companies using Python: Google, Facebook, IBM, Twitter, CERN, …
- Major applications written with significant Python component:
  - Dropbox file sharing system
  - Spotify application
  - Google search engine

Advantages of Python

- Easy to read (close to English!)
- Tons of libraries, e.g. for machine learning, visualization etc.
- Very flexible
- Since it is widely used, you can find most issues online (StackOverflow)
- Smooth integration with other tools like SQL
- It's free!

Drawbacks of Python

- Slower than compiled languages like Java or C++
  - Tradeoff between ease of coding and code performance
- Less typing can lead to confusion (you don't need to define that a variable is a string or an integer)

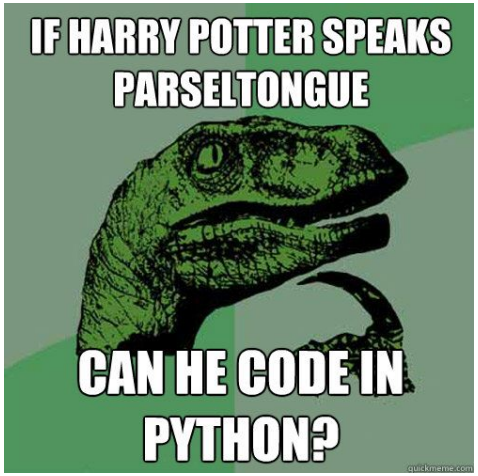
Today's Agenda:

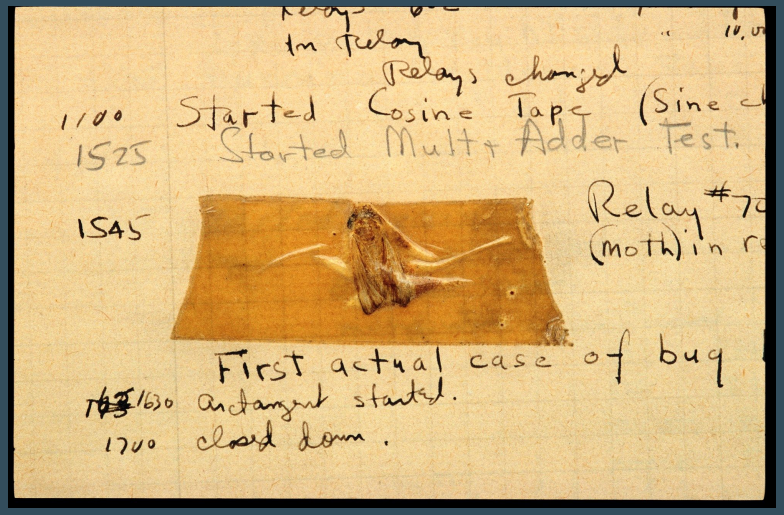
- Python Intro
  - Foundations: Variables, Data Structures, Functions
  - Data Manipulation
  - Simple Visualizations / Plotting
  - Object-Oriented Programming
  - Debugging tips
- Data in the Real World Discussion
  - Gender Balance in Computer Science

Python in action...

Speak Python with us!

- Click on “In Class Jupyter Notebooks and Data” to download the zip folder from course website
- Open 1_foundations_starter.ipynb
- You’re ready to go!





Debugging Tips

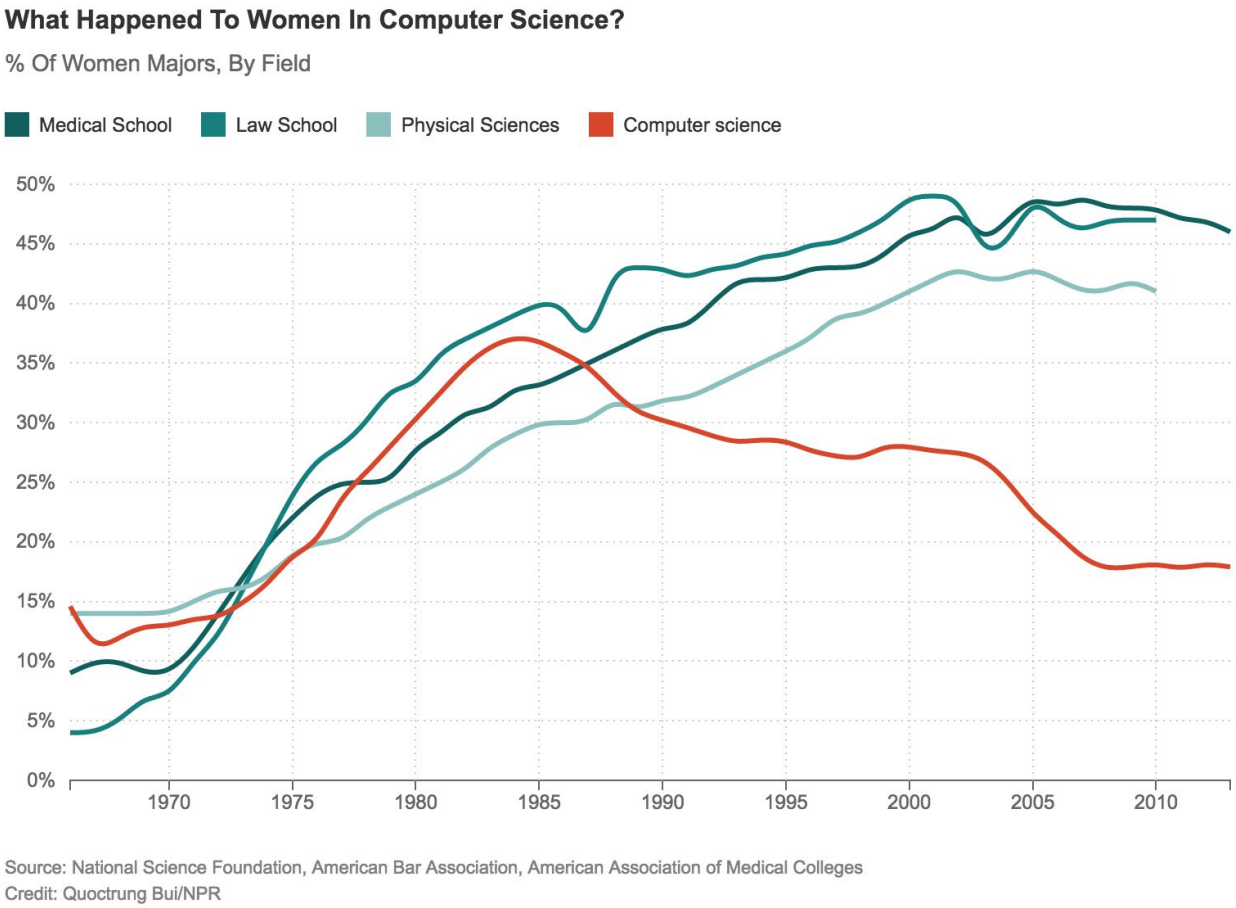
Debugging Tips

- Test individual functions separately
- Construct a variety of test cases
- Run and test your code frequently
- “Toy dataset”: Construct small test cases, e.g. on a smaller data set
- Print out variables to verify values
- Look up error messages / problems online
  - StackOverflow
- Everyone makes mistakes. Keep calm and debug on :)

Data in the real world:

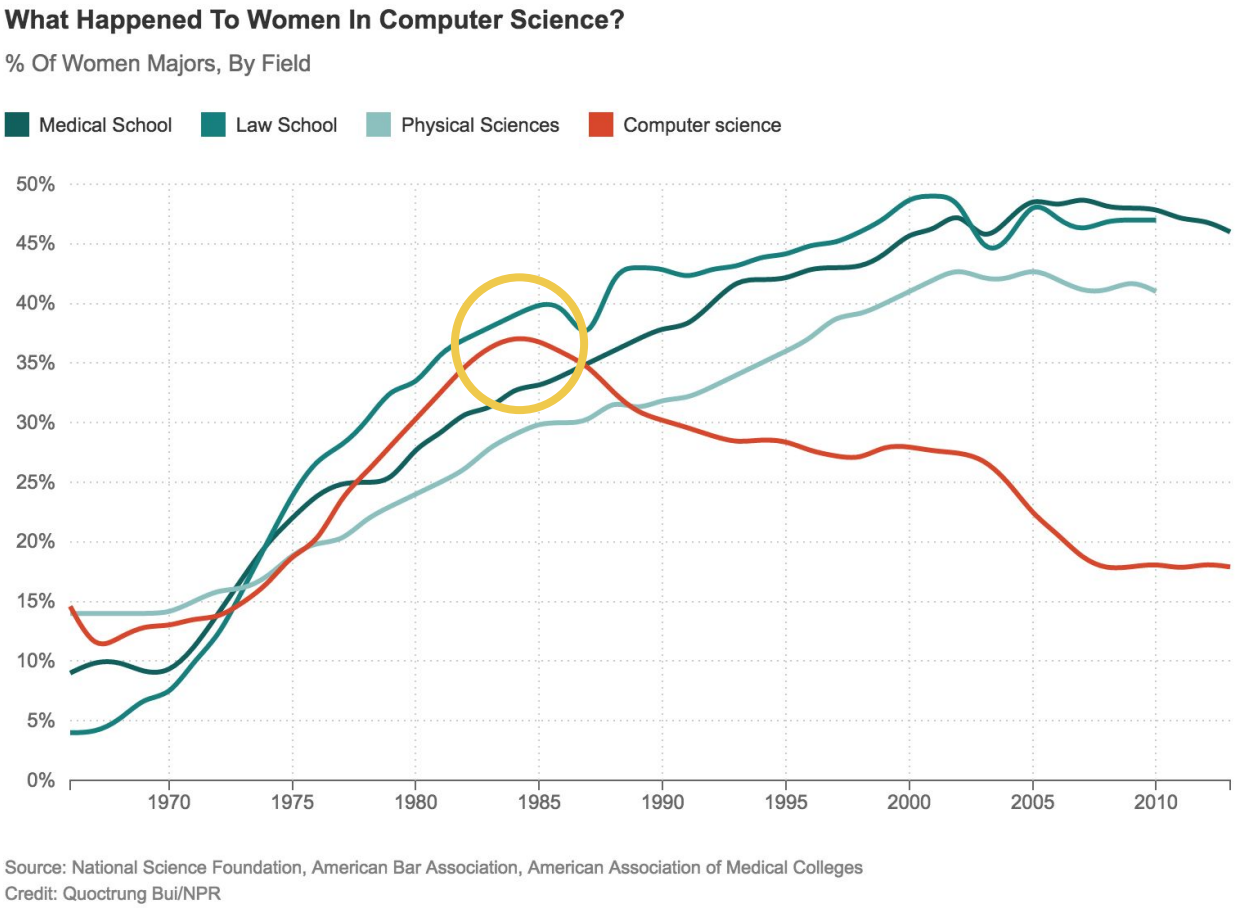
Gender Balance in Computer Science

“When Women Stopped Coding”



Source: http://www.npr.org/sections/money/2014/10/21/357629765/when-women-stopped-coding

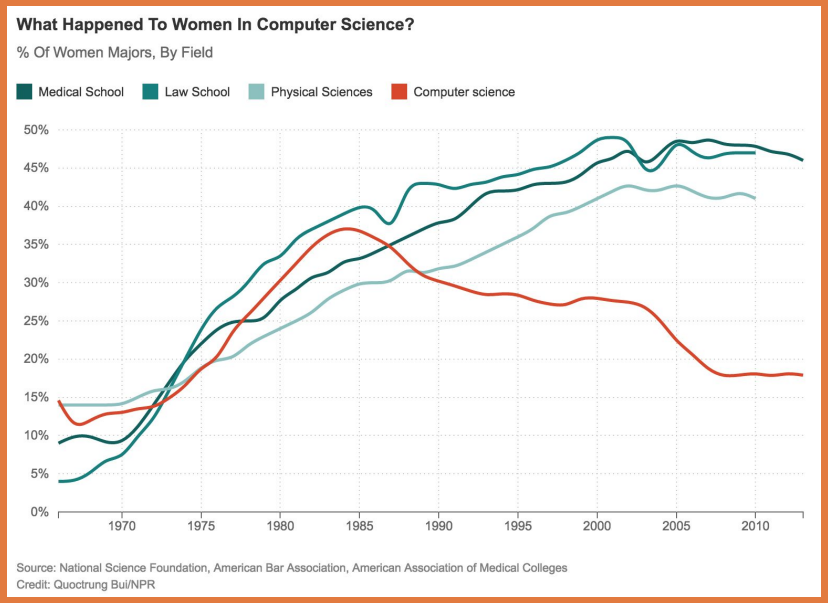
“When Women Stopped Coding”



Source: http://www.npr.org/sections/money/2014/10/21/357629765/when-women-stopped-coding

Imagine you are a member of an "Economic Future" council in the US Department of Education. Your goal is to determine why there was a sudden drop of women studying computer science.

Discuss in groups of 2-3 and take notes on the whiteboard

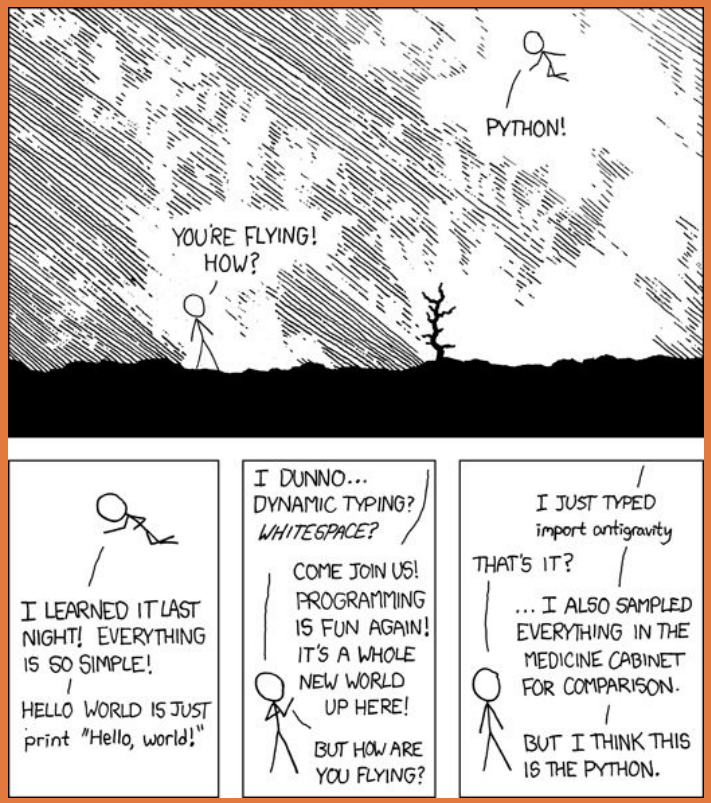


- What can you observe from these statistics?
- What is your hypothesis?
- What other data would you want to search for or gather to analyze your hypothesis?

- Try to find the data you need to evaluate your hypothesis.
- Discuss how you would analyze the data with the tools and techniques you have learned so far.
- What are your conclusions, if any?
- What recommendations would you make?

Source: NPR Planet Money. https://www.youtube.com/watch?v=vPuyDbQwfHs

Please take 2 minutes to give us feedback here:
http://bit.ly/cs102_feedback



Source: https://xkcd.com/353/

That's it for today!

The Zen of Python

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than right now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
—Tim Peters