Welcome to CS1!

• This is an exciting time to be a computer scientist! **Computational Thinking** is becoming part of all aspects of life:
  – Healthcare, Biology, Pharmacy, Biomedical Engineering
  – Economics, Finance
  – Sociology, Psychology
  – Physics, Astronomy
  – Music, Film
  – Humanities (e.g. Digital Humanities Initiative at UI)
Core Computer Science areas are full of exciting developments and activity

- Data mining and machine learning (many companies want expertise in machine learning to enable “mining” of customer and other data)
- Computer graphics, animation, virtual reality (e.g. Oculus Rift, Microsoft Hololens, Google Glass, ...)
- Human-computer interaction (HCI)
- Networks and distributed computing
- Natural language processing
- Computer vision
- Robotics
- Artificial Intelligence
- Algorithms
Search Engines

• Exemplify amazing computer science contributions
• Advances in algorithms, networking, distributed computing, machine learning all combine to support
  – millions of simultaneous searches of
  – billions of web pages, with
  – responses in milliseconds
Just a few other recent developments

- IBM’s Watson defeated the best human Jeopardy players in 2010 – substantial progress in natural language understanding

- In 1997, IBM’s Deep Blue became first computer to beat reigning world chess champion. In 2016, AlphaGo surprised many experts by becoming first computer program to defeat a top professional Go player (Lee Sedol, ranked #2 in international titles)

- Microsoft Kinect sensor – add-on for Xbox gaming console. Very impressive technology for $149. Distance and color sensors plus algorithms enabling detailed tracking of 20+ joints of 6 humans at once

- Next big thing: self-driving vehicles. Major advances in AI (deep learning), sensing (don’t hit people, animals, stuff!), etc.
This Course

• This course has one primary goal - to provide a core of computer science programming concepts and skills, and enough practice using them, that students will develop confidence in their ability to think computationally and to solve problems via programming.

• Beyond basic programming concepts, the course provides an introduction to object-oriented programming, algorithmic design and analysis, web programming, and graphical user interfaces.

• CS is much more than programming but this is a skills oriented programming course rather than a broad overview of CS. It teaches programming principles necessary for more advanced work in the Computer Science. CS1020 and CS1110 provide broader introductions to the overall field of Computer Science.
Me

• 24 years at Iowa. PhD from Cornell University, 1989. Lived many places before that - 13 in 17 years, all over US and also in Europe.

• Teaching: data structures, computer graphics, virtual reality, algorithms. Most recently have regularly taught iOS/iPhone/iPad App Development class (incl joint class with Vietnam), and Programming for Informatics (a lot like this course).

• Research: previously, the Hank VR research group. Now, the Computational Epidemiology group.

• Outside of work: when not hanging out at home, I’m somewhere in the world riding an ElliptiGO
ICON/the syllabus/course website

• ICON: used for three things
  – Submitting homeworks
  – Posting/reading questions about assignments/lectures (ICON Discussions)
  – Posting all scores/grades

• Syllabus and all course announcements, homework assignments, and lectures notes will be posted on external course website:
  – http://homepage.cs.uiowa.edu/~cremer/courses/cs1210/

The rest of today’s lecture discusses the information on the course website.
Many people find this course difficult. I believe that if you work at it steadily, not just last-minute, you can do well without suffering.

The course can be very rewarding – from (for some of you) no programming skill to real confidence and ability to make computer do cool stuff for you.

Come to class, do your own work, read and think (very valuable before sitting keyboard programming), work hard on the homeworks. If you do the homeworks well, the exams will take care of themselves.

Before next time:

- Read: Chapter 1 of the text. It is short and easy.
- Do: Install Python on your machine. Evaluate some Python code in the interpreter

Next time: beginning Python, some interesting problems
To think about: *examples of computational complexity*

- Euler vs Hamiltonian circuits and TSP. *Graphs* in computer science: different notion than the graphical plots taught in high school

- Can you find a path that traverses every edge (connection) exactly once?
- Can you find a path that visits every node (vertex) exactly once?
- If the edges are assigned “weights” (representing distance/cost to traverse), can you find path of minimum cost that visits every node exactly once?

*Are these similarly computationally hard or easy?*