Computer Science at Temple
Welcome!

• Exciting time to be studying computing
• Programming + *any scientific field* = good mix
How to do well in Computer Science courses
Show up

• Minimum
• Attendance normally isn’t graded, BUT
• Those who do well tend to be ones who attend regularly
• Try to put devices away (even for notetaking):
  • They’re distracting to you (and your neighbors)
  • We’re not as good at multitasking as we think
Practice

• More important than anything else
• Can’t just watch others or read others’ work. Think about learning:
  • a foreign language
  • riding a bike
  • to play the cello
• Not enough that you followed the lectures
  ( ... again, think about how that would work with German, the bike, or cello)
• Critical for your development
How to get practice

• Intro courses have weekly programming assignments
• Review, modify programs that you’ve done in class
• Example problems in your textbook
• Try to implement library functions on your own
• Websites like codingbat, codeacademy, leetcode
  • and maybe one associated with your textbook
Do your own work

- Easy to cheat (BTW, also easy to get caught)
- Purpose of programming assignments is to give you practice
- Programming classes require programming in exams
- You’re not getting the practice you need if you cheat
- Course content is by nature cumulative
  - Think of the Algebra courses you took in high school: could you possibly have passed Algebra II without understanding what was taught in Algebra I?
  - Same is true for your CS programming courses
- Get lots of practice, and if you’re having trouble, get help quickly
Working with Others

**Good sharing**
- Talking about ideas, algorithms, general approaches to solving problems
- Sketching outlines together on a whiteboard

**Bad sharing**
- "You do the answer to the first part. I’ll do the 2nd and we’ll swap”
- “Here’s my code. You can look at it to get some ideas”

- In short: unless your instructor tells you otherwise, the code-writing part of an assignment must be done solo
Where to get help

• Resist the urge to copy and paste code you’ve found online
• Remember the bigger goal is to learn how to program
• “... but my friend says copying and pasting online code is how it’s done”
  • Yes. By people who don’t know how to program
  • There are bad mechanics, electricians, and doctors too. Don’t be one of them
  • ... and Google isn’t there to bail you out on the exam
• You’re always welcome to see:
  • peer tutors
  • your lab instructor
  • the professor
Peer Tutors

• Tutoring for required CIS classes

• Who are the tutors?
  • Excellent upper-level undergraduate CIS students
  • Taken all intro courses and done exceedingly well

• When?
  • ≈ 9-5 Monday through Friday
  • Detailed schedule posted in the beginning of each semester

• Where? SERC 371
Keep up

• Don’t dig yourself into a hole
• Not easy to cram
• A little bit of daily consistent practice better than all-nighter
• The earlier you realize you need help, the easier it is to get help
Fear of Asking Questions

• In intro CS courses, we frequently hear students say that they’re afraid to ask questions, because they believe they have so much less background than other students

• Don’t be intimidated. When you’re unsure, please ask
  • You’re not here because you already know everything. You’re here to learn
  • CIS 1051, 1057 assume no prior programming background
  • Some students do some programming before college, but the playing field tends to level out between the students with some experience and those brand new to programming very quickly
More on: Fear of Asking Questions

*(i.e., More on: Fear of Looking Stupid)*

- We want to see you do well
- It can be scary to ask, but a good instructor will welcome your question and either:
  - answer it immediately OR
  - ask to arrange a time to address your question outside of class
Should I take CS, IS&T, Math-CS, Data Science?

• Which is which? Am I in the right program?
• Don’t stress
  • Broadly speaking, they’re similar
  • Graduates in all majors have excellent job placement
  • Lots of overlap
    • many shared courses
    • for most part can apply for same jobs at graduation
  • As you progress, you’ll figure out what you like, find courses that look interesting to you. In which program are these courses?
• Switching from one program to other shouldn’t delay graduation significantly, particularly if done within first few semesters
Some more specifics: CS vs IST vs Data Science

• These programs are much more similar than they are different
• Practitioners:
  • apply for many of the same jobs
  • work to solve many of the same problems
  • use similar tools to do their work
• Employers tend to care more about what you can do than the name of your degree. (Is there really a big difference between a CS major who has taken some data science electives and a data science major?)
• Some subtle differences
Some subtle differences: Computer Science

- Design, write software
- Develop new ways to use computers
- Devise tricks for solving challenging problems efficiently
Some subtle differences: IS&T

- Apply existing technology to meet need of organization
- Process and distribute information
- Communication with users of technology
Some subtle differences: Data Science

• Relatively new field
• Interdisciplinary
  • Data can come from wide range of sources
  • Helps to know something about the context, *e.g.*, data about thousands of patients taking a particular drug, helps to know something about medicine, chemistry.
• How do we take mountains of raw data and learn something from it?
Intro Course Sequence: 1051 or 1057?

- Don’t stress. Either is perfectly fine
- Not a big difference: more about learning to program than a learning particular language
- Suppose that you take 1057:
  - by the time you finish 2168, in a weekend you should be able to pick up as much Python as is taught in a semester of 1051
  - same is true for C if you took 1051
- CS majors program in C in 2107, 3207
  - so maybe tiny advantage for them to take 1057
CIS Department Student Organizations

- Consider joining, or at the very least visiting a meeting of:
  - ACM
  - ACM-w
  - TU-Dev
  - STARS

- Why?
  - Learn about technologies, job opportunities that might interest you
  - Interaction with other humans
  - Often free pizza
  - No need to make commitment
Hackathons

- **Competitions** in which hackers rush to create something in a very short period of time (usually a weekend)
- Amazing what is achieved in such short time
- Sometimes nice prizes
- Why try
  - Meet new people
  - Learn about lots of exciting technologies that you didn’t know about
  - Gain self confidence
  - Become inspired. (Sounds lame but true)
- A group or two from [TUDev](#) usually goes to local events like [PennApps](#)
- “... but I’m not good enough”
  - After a couple of semesters, you do have enough background
  - Be brave. You have absolutely nothing to lose. What were you going to do that weekend anyway?
Internships

• Apply
• Many (most?) CIS internships are paid.
• Potential for:
  • Good work experience prior to graduation
  • Looks good on resume
  • Learn about what you might like (and dislike) at a job
• Try to keep on track with course work and plan to intern:
  • summer between junior and senior year
  • and even possibly summer between sophomore and junior year
More on internships

• “... but I don’t know enough”
  • If you’ve worked hard and done well in your courses, by the time you’ve finished CIS 2168, you do know enough
  • Companies know that they’re hiring interns, not 10-year professionals. They cannot possibly expect the same level of knowledge
  • (Often, it’s the very best students who are most afraid of taking the plunge because they’re the most aware of what they don't yet know.)

• Don’t put off application prep:
  • be prepared for applications by early in the fall semester
  • pay attention to postings from ACM’s Claudia Pine-Simon
  • see Rose McGinnis, CST’s Professional Development Director
Experiment

• Only so much that can be covered in a few hours/week for 14 weeks
• Teach yourself to be proficient in different tools, *e.g.*:
  • IDE
  • Debugger
  • Version control
  • Command line
  • Testing frameworks
• Operating systems. Try something new. Use virtual machines
• Like many other things, you only get good with these things if you actually use them
• ... but don’t lose track of the fact that your priority is your coursework
More on experimentation

• You're going to be taught lots of things in class
• Maybe most important: learning how to learn
• Hot new technology 10 years from now? Who knows?
• When you’re off working you don’t want to be the person who needs to be sent to a paid training in order to learn a new thing
• The earlier you experiment independently, the better
Summary

Most critical
• You can do this
• Come to class. Be engaged.
• Get as much practice as you can. Do a little most days
• Don’t cram
• Don’t dig a hole: get help as soon as you realize you need it

Also keep in mind
• Participation (e.g., student groups, hackathons, internships)
• Independent experimentation