CSCI 0150
(also known as CS15)
A Gateway to Computer Science
CS15 Head TAs: We’re here for YOU!

Daniel

Lila

Harriet

UV

Will
Computer Science (1/2)

- CS15 is a start to understanding computer science
  - for your own intellectual interest
  - for its enrichment of other fields
  - for its combination of scientific, engineering, art and design concepts and practices, and as a “mode of thought” — “computational thinking”
Computer Science (2/2)

- IT, or information technology, including CS, is key to the “knowledge economy”

- Omnipresent in a breadth of various applications and fields
Stunning Special Effects

Disney’s “Soul”

Disney’s “The Lion King”
Immersive Virtual Reality

- Researchers can create fully immersive 3D environments via head-tracked stereo glasses, enabling realistic “field geology” on Mars!

- A state-of-the-art “Cave,” the YURT (Yurt Ultimate Reality Theater), at 180 George Street
  - much higher quality (e.g., 100Mpixels) and much more comfortable than headsets (Oculus Rift S, VALVE HTC Vive Pro, and other VR) but way more expensive!
Augmented Reality

- Creates virtual elements “on top of” the real world, blending a digital reality with an existing one!
- Smartphone apps, e.g., Pokémon Go
- Microsoft HoloLens 2: special glasses with built-in head tracker that create a mixed reality
The Internet and Social Networks

- Facebook, Inc. – July 28, 2021
  - 3.51 billion use Facebook, Instagram, WhatsApp monthly
  - 500 million use Instagram Stories daily as of Aug 2021
  - 10+ million active Facebook groups
  - Facebook removed 1.3 billion fake profiles in the first quarter of 2021
CS15 is Not Just About Learning to Program

- Introduces some fundamental concepts in CS
- Introduces some of the societal context and implications of our field
  - "Socially-Responsible Computing" (SRC) in the Department and in CS15: Anabelle Johnston and Lucas Gelfond as our STAs
Opportunities/Threats of the Digital Age (1/5)

● Machines continue to replace human labor and decision-making
  ○ machines have increased human productivity while reducing demand for routine, repetitive, and dangerous jobs (factory work, coal mining,…)
  ○ as middle-skilled, task-intensive jobs disappear, income gap (“income inequality”) widens
  ○ but new jobs are being created, old jobs “upskilled” to be more interesting
    ▪ impacting not just blue collar jobs such as factory work or driving: x-ray reading, tax advising, news reporting,…

● Education is key to economic survival

• Should there be a “robot tax”?
• Should there be a “guaranteed minimum income,” also called Universal Basic Income? (Andrew Yang ‘96)
• “What is the future of work?”
Opportunities/Threats of the Digital Age (2/5)

- Dangers of yielding too much control to algorithms, some too complex to be understood by most people
  - instability in the stock market due to trading algorithms
  - autonomous vehicles (autopilot on planes, driverless cars…)
  - nuclear power plants and other infrastructure
  - “bias” in algorithms (facial recognition, mortgage lending, job placement…)

Andries van Dam © 2021 9/9/21
Opportunities/Threats of the Digital Age (3/5)

- Cyberfraud, Cybercrime, Cyberwarfare
  - culpability of social media in spreading mis- and dis-information thereby causing polarization and distrust (e.g., 2020 election, COVID-19 vaccines, organizing the insurrection)

- we keep experiencing huge data breaches
  - Russia’s hack of SolarWinds, a major information technology firm, infiltrated over 18,000 clients’ systems
  - impacted clients included Fortune500 companies and many gov’t agencies (Pentagon, National Nuclear Security Administration, State Department, etc.)

- offense has the advantage over defense
- schools in Russia, China, North Korea (at least) teach hacking… we’re well beyond amateur hacking
- will the next war be fought by drones, and how can they be controlled?

- Brown is strong in cybersecurity technology and policy
Privacy and Security

- Mathematics
- Algorithms
- Systems
- Law
- Politics
Opportunities/Threats of the Digital Age (4/5)

- Big Data
  - “data mining,” “machine learning,” “deep learning,” “reinforcement learning”…
    - statistics-based algorithms for detecting patterns, anomalies, etc.
  - search engines
  - real-time language translation
  - facial recognition
    - can identify faces in crowd photos
  - gesture recognition for user interfaces
  - credit card fraud detection
  - crime and terrorism anticipation
  - but what about privacy in the age of the “surveillance state”?!?
Opportunities/Threats of the Digital Age (5/5)

- Big data & personal privacy
  - threat to privacy represented by increasing storage of personally identifiable information – is there any real “anonymous data”?!?
  - Google, Facebook, Apple… and their data collection and use of that data – our digital footprint is permanent, and we have no control over how it is used
    - Sun Microsystem’s Scott McNealy – “privacy is dead, get over it!”
    - When Apps are free, YOU are the product

- “Free speech” vs. (appropriate) censorship
  - hate speech, terrorism, violence
  - government-induced censorship (e.g., China, Saudi Arabia, Pakistan,…)
  - are social media content-neutral platforms or publishers, and what laws should apply to them?

- Need an educated government, citizenry
  - can we pass realistic laws to govern behavior?
CS: So Much More Than Programming!

- Computers are our only universal machine, through the magic of software…
  - if you can program it, a computer can execute it
- Programming is a means to an end, much like mathematics is… but they are both also fascinating topics in their own right!
- Big push to learn how to “code,” but there is no “royal road” to programming or CS – it requires serious, sustained effort
Computer Science at Brown Works on Hard Questions (1/3)

• How can robots understand their surroundings, the behavior of people, and language, to solve real-world problems?
Computer Science at Brown Works on Hard Questions (2/3)

• How can AI understand the intricacies of human language the way humans do?

Ellie Pavlick
Computer Science at Brown Works on Hard Questions (3/3)

• How can we use encryption to promote privacy?
• How can we analyze the efficiency of algorithms we use in encryption?
Other Areas of Research at Brown

• **Algorithms and Theory** (Lorenzo De Stefani, Pedro Felzenszwalb, Sorin Istrail, Philip Klein, Tim Nelson, Roberto Tamassia, Franco Preparata, Benjamin Raphael)
• **Artificial Intelligence** (Stephen Bach, Pedro Felzenszwalb, Amy Greenwald, George Konidaris, Michael Littman, Ellie Pavlick, Stefanie Tellex…)
• **Comp Bio** (Sorin Istrail, Sohini Ramachandran, Thomas Dean, David Laidlaw, Mark Nadel…)
• **Data Science** (Karianne Bergen, Ugur Cetintemel, David Laidlaw, Ellie Pavlick…)
• **Machine Learning** (Stephen Bach, Daniel Potter, George Konidaris, Michael Littman, Daniel Ritchie, Ritambhara Singh, Eugene Charniak, Eli Upfal…)
• **Security** (Seny Kamara, Vasileios Kemerlis, Shriram Krishnamurthi, Anna Lysyanskaya, Steve Reiss, John Savage, Tarik Moataz…)
• **Visual Computing** (Andy van Dam, James Hays, John Hughes, David Laidlaw, Barbara Meier, Daniel Ritchie, James Tompkin, Bruce Campbell…)

• And more… [http://cs.brown.edu/research/areas.html](http://cs.brown.edu/research/areas.html)
CS: So Much More Than Programming! (2/2)
Why Should You Study Computer Science?

- For fun and intellectual excitement
- Really exciting era is just beginning
  - CS still a young discipline, computers just starting to act intelligently
- Fundamental “mode of thought”
- Increasingly important component of all other fields
- Plenty of exciting and impactful jobs in established companies, start-ups, research labs, and academia
Welcome To CS15!
Welcome to CS15 in Salomon 101!

● We encourage you to download the PowerPoint slide deck before lecture and bring your laptop – lets you see clearly and annotate
  o http://cs.brown.edu/courses/cs015/

● We record and give you web access to every lecture
  o for review
  o for asynchronous students
  o in case you must miss a synchronous lecture
  o PowerPoint slides come with associated recording
Accessibility for Hybrid Learning

- CS15 can be taken asynchronously, only for students not physically on campus
  - register on C@B for asynchronous section (S02) – lecture will be watched asynchronously, section/lab and TA hours will still be synchronous
- Section, Lab, and TA Hours offered at times that will accommodate all time zones
- We are here for you!
  - reach out to TAs/Andy at any time – we want to help support you through this semester
Our Hardware

- The Sunlab: 80 PCs running Debian Linux
  - COVID-19 moved us away from Sunlab
- Now all work will be done locally on your own computer
  - using GitHub to acquire and store code
  - IntelliJ IDEA – integrated development environment (IDE) we will use to write, compile, and run code
- Working From Home setup materials and instructional video released after lecture
  - come to TA hours for help setting up software!
CS15 is All That

● **NOT** a course about video games or game design

● Uses games as a domain to teach Object-Oriented Programming (OOP)
  ○ most common current programming methodology
  ○ Brown was earliest to switch to Java for intro courses almost two decades ago
  ○ still a dominant web programming language (e.g., Google’s Android)

● Teaches fundamental *problem-solving skills* useful in all disciplines

● Provides introduction to computer science concepts

● Is **intense**, but **fun**, especially with interactive graphics
Who is CS15 For?

- Students with varying levels of programming experience, including **NONE!**
  - however, CS15 still requires a **serious** commitment

- Most students have **little or no** programming experience, including the TAs and HTAs when they took the course!
  - let’s visualize this!

- Prospective CS concentrators, who will go on to new CS200 course

- This is **not a weeding-out course**, but it is still time-consuming
  - don’t worry!! We expect lots of confusion in the beginning. All 44 TAs are here to help you through that initial confusion!
CS15 isn’t About Getting the Correct “Answer”

- It’s about the **process**, not just the final product!
  - **design**
    - planning efficient, effective designs for program structure
    - investing upfront (e.g., reviewing materials) saves time in long run
  - **implementation**
    - coding incrementally
  - **debugging**
    - diagnosing and fixing bugs/ errors in code effectively
Diversity & Inclusion in CS15

- CS15 welcomes all, helps you succeed, and aims to build community. These additional groups are also here for you:

- Mosaic+ mosaic.plus.brown@gmail.com
  - “created to foster Community, inspire Innovation, and provide opportunity to underrepresented minority students.”

- Women In Computer Science (WiCS) wics@lists.cs.brown.edu
  - “formed by female undergraduate students at Brown in the late 1980s, The goal of WiCS is to increase the participation of women in the field of Computer Science.”

- Women in Science & Engineering (WiSE) WiSE@brown.edu
  - “to encourage women who study in all science and engineering fields, by building a community of like-minded scholars that provides peer support on their journey to becoming successful scientists at Brown and beyond.”

- Our own CS15 mentorship program!
  - more on this later
Why Java

• Supports interactive OOP
• Syntax similar to C++ but simpler, cleaner, and more beginner-friendly
• Allows platform-independence: write once, run everywhere (in principle)
• One of the most prevalent languages in industry today, e.g., Android, web servers (others include C, C++, C#, Python, Ruby, etc.)
• Note: *not* the same as JavaScript, a less purely object-oriented language used commonly in web applications
• OOP is one of several programming paradigms – CS17 uses ReasonML and Racket for “functional programming”
Course Mechanics (1/4)

● **No** quizzes or exams!
  o no exam time pressure
  o no “grading on a curve”: you do the work, you get the grade you deserve! Thus A is by far the most common grade

● 9 Assignments
  o programming assignments, some of which have a design component
  o from brief homework to Tetris and beyond!
  o choose from a selection of final projects, or create your own “indy” project
  o all programs must meet a baseline level of functionality to receive credit, lots of room for “bells and whistles” for fun and extra credit
  o **all programs must be handed in with baseline functionality by end-of-semester!**
Course Mechanics (2/4)

- Assignments are graded on a hand-in schedule
  - most assignments have early, on-time, and late hand-in
  - early hand-in: 2% increase to your grade
  - late hand-in: 8% decrease from your grade
  - all assignments must be handed in before the end of the course

- TopHat questions during lecture
  - interactive questions to improve engagement and comprehension
  - graded on completion for synchronous students. Accounts for 5% of final grade
  - for asynchronous students, this 5% will be accordingly distributed among other portions of the total grade

- Weekly discussion/lab sections
  - graded on mini-assignments and participation
  - accounts for 12% of your final grade
Course Mechanics (3/4)

- Keys to success
  - increase in program size and complexity throughout the semester
  - you can’t procrastinate and then cram, unlike in some other courses
  - start early, start today, start yesterday!!!
  - other courses don’t teach you to tackle programs of this complexity

- TA Hours
  - 39 TAs and 5 Head TAs
  - 150+ TA hours of personalized help per week!!!
    - more than in any other course!
    - everyone struggles sooner or later, including the TAs – part of the learning process
    - we strongly encourage you all to go to hours and get to know the TAs - it is integral to the course (and NOT a sign of weakness!)
Course Mechanics (4/4)

• CS15 thrives on your feedback

• Questions *highly* encouraged during lecture! And we will add TopHat questions next week…

• We provide a lot of written material; YOU are responsible for digesting all of it
Major Changes This Year (1/2)

• Loosened Collaboration Policy
  o as requested by students through previous years’ feedback forms
  o stay tuned for more explanation later in this lecture…
Major Changes This Year (2/2)

• Slower pace for first month of course
  o emphasize foundations of Object-Oriented Programming used throughout semester
  o two *new* smaller projects to replace previous project

Pong

Tic Tac Toe
Alternatives to CS15 (1/2)

For Concentrators & Non-concentrators:

- CS17 (fall semester) – John Hughes
  - also, no prior experience required
  - multiple programming paradigms
  - multiple programming languages
    - Racket, ReasonML
  - mastery, not mystery → no magic
  - focus on problem-solving skills/strategies
    - emphasis on abstraction and scale
  - integrate programming with analysis of algorithms
  - multiple application areas (AI, databases, etc.)
  - pair programming for labs and projects
  - for more information on other CS courses: [http://cs.brown.edu/degrees/undergrad/whatcourse/](http://cs.brown.edu/degrees/undergrad/whatcourse/)
Summary of CS15/17 Choice

- Both will adequately prepare you for upper-level courses
- No prior experience needed for either, similar work loads
- Different material covered
  - CS15 – Object-Oriented Programming, CS17 – Functional Programming
  - CS15 is more practice-oriented, CS17 is more foundations-oriented
  - CS15 celebrates magic, while CS17 emphasizes no magic
    - CS15 has little reliance on TA support code, but uses JavaFX extensively
- Less pair programming and collaboration on projects than CS17
  - but no tests
- CS15: games and skits
- Pick based on your taste and what appeals to you – you can’t go wrong!
- Both CS15 and CS17 feed into CS 200 in spring semester, new course with debut this year
  - CS15 will get quick 2-week intro to functional programming, CS17 to OOP and Java
Alternatives to CS15 (2/2)

For Concentrators & Non-concentrators:

- CS0111 (Fall + Spring) – Kathi Fisler
  - no prior experience required
  - the first in a sequence that spreads the foundational concepts over three courses rather than two
    - “allow more time to combine CS with other studies & mastering the fundamentals”
  - functional programming and imperative programming
    - learn Pyret and Python
  - integrates programming with data science and discussion of use of digital information
  - less intensive workload
  - option to do extra work at the end for students who want to feed directly into CS200
Alternatives to CS15 and CS17/19

For Non-concentrators:

- **CS20: The Digital World (Fall) - Donald Stanford**
  - introduction to computing; little emphasis on programming
  - discusses computing topics such as artificial intelligence, IT security, and digital media
  - a small introduction to HTML, Photoshop, Access, and Python

- **CS100: Data Fluency for All (Fall) - Amy Greenwald**
  - introduce data literacy, basics of statistics, machine learning, data communication
  - hands on experience using statistical tools such as 'R' to analyze real world data sets, and 'ggplot' to visualize them.
Collaboration (1/6)

- Brown’s Academic Code
  - “Academic achievement is evaluated on the basis of work that a student produces independently. A student who obtains credit for work, words, or ideas that are not the products of his or her own effort is dishonest and in violation of Brown’s Academic Code. Such dishonesty undermines the integrity of academic standards of the University. Infringement of the Academic Code entails penalties ranging from reprimand to suspension, dismissal, or expulsion from the University.”
Collaboration (2/6)

- CS15 Collaboration Guidelines
  - Lectures
    - *always* allowed to review and discuss with your peers!
  - Mini-assignments
    - collaboration and discussion are *allowed and indeed encouraged*
  - Labs / Sections
    - collaboration and discussion are *allowed and again encouraged*
Collaboration (3/6)

- Collaboration on Programming Assignments
  - first half of semester, no collaboration on assignment programming or debugging (same as last year)
  - second half of semester, more flexible rules around debugging with peers
  - two programming assignments (DoodleJump and Tetris) planned and implemented with a partner
  - as trade-off for partner coding, individual check-ins after submitting partner assignments
    - 5-minute conversation with two TAs about the assignment concepts
    - not intended to be scary, nor an oral exam
    - if you understand course concepts, you will get an A on the check-in
  - will read detailed CS15 Collaboration Policy during first assignment
Collaboration (4/6)

- **MOSS (Measure of Software Similarity)**
  - Stanford-hosted AI software used to detect plagiarism – it signals undue similarity and we hand-check the code
  - used across industries in multi-million dollar lawsuits to protect intellectual property
  - every year, MOSS finds multiple collaboration violations (we check multiple years!)
  - last year, half a dozen cases, all found guilty by UH’s official committee
  - punishments typically directed NC, parental notification
  - MOSS is *very good* at what it does – don’t even think of trying to outwit it! (which is more work than doing the assignment!)
  - we also check the web

If ever in doubt about what is allowed, ask a TA!
Better to NC an assignment or even the course than being accused (and likely convicted)!

Note: we have a Regret Policy
Collaboration (5/6)

The issue of collaboration in intro CS courses has been in the news in past years:

Possible cheating uncovered in popular Harvard computer class
By Travis Andersen and Brian MacQuarrie Globe Staff  May 05, 2017

As Computer Coding Classes Swell, So Does Cheating
By JESS BIDGOOD and  • MAY 29, 2017

Competitive environment drives culture of cheating in computer science classes
BY KATE HUANGPU | FEBRUARY 22, 2018, 3:43 AM
Collaboration (6/6)

- Illegal collaboration is **not** worth the risk
  - start early and get help when you need it! Lots of resources available to help you succeed in this course
What is Programming?

Aspects of Programming, Computer Languages, Objects and Object-Oriented Programming
Many Aspects of Programming

- Programming is **controlling**
  - computer does exactly what you tell it to do – literal minded idiot savant

- Programming is **problem solving**
  - always trying to make the computer do something useful
  - e.g., finding an optimal travel route
  - methodology is applicable to other fields

- Programming is **creative**
  - must find the best solution out of many possibilities

- Programming is **modeling**
  - describe *salient* (relevant) properties and behaviors of a system of components (objects)

- Programming is **abstraction**
  - identify important features without getting lost in detail

- Programming is **concrete**
  - must provide detailed instructions to complete task

- Programming is a **craft**
  - a bit like architecture, engineering – disciplined and creative craft for building artifacts
What’s a Program? (1/3)

- Model of complex system
  - model: simplified representation of important features of something, either tangible or abstract
  - system: collection of collaborating components
What’s a Program? (2/3)

- Sequences of instructions expressed in specific programming language
  - syntax: grammatical rules for writing instructions
  - semantics: meaning/interpretation of instruction
What’s a Program? (3/3)

● Instructions written (programmed/coded) by programmer
  o coded in a specific programming language
  o *programming languages* allow you to express yourself precisely unlike *natural (human) language* that thrives on “shading”, nuance, ambiguity, implicit context…
  o algorithms are 100% literal, cannot have ambiguities

● Real world examples
  o Banner, email, video game, smartphone and apps, ATM, embedded computers in appliances and vehicles…

● Executed by computer by carrying out individual instructions
Java Programs

- CS15 uses Java
  - Java was developed by Sun Microsystems (absorbed by Oracle)
    - the Sunlab was named for the desktop computers that it held for over a decade
  - it is meant to run on many “platforms” without change, from desktop to cell phones
  - platform independence
  - but Java isn’t sufficient by itself: many layers of software in a modern computer
The Computer Onion

- Layers of Software
  - cover hardware like an onion covers its core
  - make it easier to use computers
  - organized into libraries and programs

In CS15, we only deal with the outermost layer
Two Views of a Program

user interface

software layers hidden by user interface

user’s view

programmer’s view
Programming Languages (1/2)

- Machine language – computer’s native language
  - sequence of zeroes and ones (binary)
  - different computers understand different sequences
  - too hard for humans to understand (01010001…)
- Assembly language
  - symbolic but still one-to-one with machine language
  - still hard for humans to understand:
    - ADD.L d0, d2
  - assembly language taught in CS33
Programming Languages (2/2)

- High-level languages
  - FORTRAN, C, C++, Java, C#, Python, JavaScript, Scheme, Racket, Pyret, ML, OCaml, etc.
  - high level: each instruction is composed of many low-level instructions
  - closer to English and high school algebra
    \[ \text{hypotenuse} = \text{Math.sqrt}(\text{leg1} \times \text{leg1} + \text{leg2} \times \text{leg2}); \]
  - much easier to read and understand than Assembly language
  - unlike machine and assembly language, it is machine-independent
Running Compiled Programs (1/2)

- In CS15, code in a high-level language, Java

- But each type of computer only “understands” its own machine language (zeroes and ones)

- Thus must translate from Java to machine language
  - a team of experts programmed a translator, called a “compiler”, which translates the entirety of a Java program to an executable file in the computer’s native machine language
Running Compiled Programs (2/2)

- Two-step process to translate from Java to machine language:
  - compilation: your program → executable
  - execution: run executable
  - machine executes your program by “running” each machine language instruction in the executable file
  - not quite this simple “underneath the covers” — “Java bytecode” is an intermediate language, a kind of abstract machine code
Object-Oriented Programming (1/2)

- OOP: the dominant way to program, yet it is over 40 years old! (Simula ‘67 and Smalltalk ‘72 were the first OOPLs)
  - Dr. Alan Kay received ACM’s Turing Award, the “Nobel Prize of Computing,” in 2003 for Smalltalk, the first complete dynamic OOPL

- OOP was slow to catch on, but since mid-90’s it’s been the dominant programming paradigm
  - but it isn’t the only useful programming paradigm…

- CS17 and 19 teach functional programming in
  - Racket
  - ReasonML

- CS200 will also teach some functional programming – you won’t miss out
Object-Oriented Programming (2/2)

- OOP emphasizes objects, which often reflect real-life objects
  - have both properties and capabilities
  - i.e., they can perform tasks: “they know how to…”
- Look around you… name that object!
OOP as Modeling (1/3)

● In OOP, model program as collection of cooperating objects
  o program behavior determined by group interactions
  o group interactions determined by individual objects

● In OOP, objects are considered *anthropomorphic*
  o each is “smart” in its specialty
  o e.g., bed can make itself, door can open itself, menu can let selections be picked
  o but each must be told when to perform actions by another object – so objects must cooperate to accomplish task
OOP as Modeling (2/3)

- Each object represents an *abstraction*
  - a “black box”: hides details you do not care about
  - allows you as the programmer to control programs’ complexity – only think about primary features
OOP as Modeling (3/3)

- So, write programs by modeling the problem as a system of **collaborating components**
  - you determine what the building blocks are
  - put them together so they cooperate properly
  - like building with smart Legos, some of which are pre-defined, some of which you design!
  - containment diagrams, like the one shown here, are a great way to help model your program!
Example: Tetris (1/3)

- What are the game’s objects?
- What properties do they have?
- What do those objects know how to do?
Example: Tetris (2/3)

- What are the game’s objects?
  - piece, board

- Properties: What attributes and components do they have?
  - piece
    - orientation
    - position
    - shape
    - color
    - tiles
  - board
    - size
    - rows
Example: Tetris (3/3)

● **Capabilities**: What do those objects know how to do?

○ **piece**
  ■ be created
  ■ fall
  ■ rotate
  ■ stop at collision

○ **board**
  ■ be created
  ■ remove rows
  ■ check for end of game
Announcements (1/2)

- If you are even considering taking the course, register or add it to your primary cart on C@B by this weekend
  - you will not get course emails unless you do this!

- Lab signups released today (9/9) – instructions sent via email
  - we will email you your lab meeting time by Monday
  - first lab occurs Tuesday 9/14 and Wednesday 9/15

- We will send an email with instructions to set up for CS15
  - set up IntelliJ code editor **before your lab time**
  - set up Top Hat lecture quiz software **before Tuesday’s lecture**
  - join Ed online Q&A forum as soon as you can
Announcements (2/2)

- RISD and other non-Brown undergrads please speak to an HTA or Andy after class

- Head TA (HTA) Hours this weekend – come if you have any questions about CS15 or just want to say hello!
  - Friday 3-5pm in Sayles 104
  - Sunday 7-9pm in Friedman 101

- Follow us on TikTok @cs15.brown

- Check the course website at http://www.cs.brown.edu/courses/cs015 and your email daily

- If you are undecided about which CS intro course to take, this document is a good reference:
  - https://cs.brown.edu/degrees/undergrad/whatcourse/
Hope you’re excited for a great semester!
Software Development: A 5-Step Process (1/2)
Software Development: A 5-Step Process (2/2)

1. Analysis
   a. English description of what the system models to meet user requirement/specification

2. Designing the system
   a. “Divide et impera” – divide and conquer: system is composed of smaller subsystems which in turn may be composed of even smaller subsystems (diagrams often helpful)

3. Implementing the design (in Java for CS15)
   a. if design is good, most of the hard work should be done

4. Testing and Debugging
   a. testing: submitting input data or sample user interactions and seeing if program reacts properly
   b. debugging: process of removing program bugs (errors)

5. Maintenance
   a. in a successful piece of software, keeping a program working and current is often said to be 80% of the effort