Week 01

Loading & Documentation

INFO 3402: Information Exposition

Brian Keegan, Ph.D.
Assistant Professor, Information Science
brian.keegan@colorado.edu

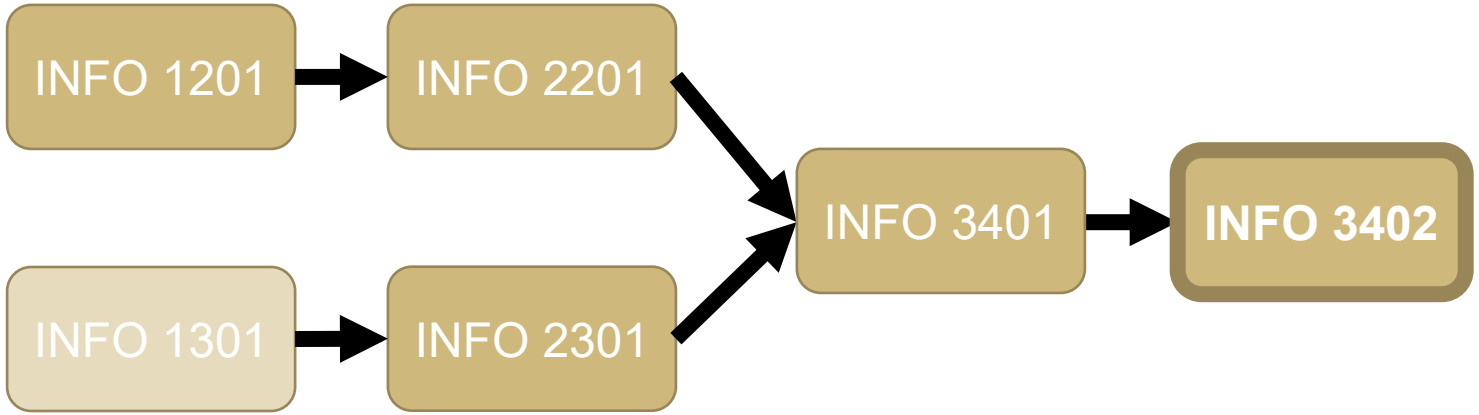
University of Colorado Boulder

Agenda

- 00:00 – 00:15 → Course Overview
- 00:15 – 00:30 → Data Science Mindset
- 00:30 – 00:40 → Documentation and Professionalization
- 00:40 – 00:75 → Loading Data, Weekly Assignment, Next Class

Course Overview

Why Are You Here?



- Analyze and interpret relationships in quantitative data
- Create effective data visualizations through iterative design
- Communicate findings for general and professional audiences
- Think critically about and critique data narratives and visualizations

Course Design

- Six modules about data analysis, visualization, and communication
  - Shaping, Distribution, Comparison, Trend, Relationship, Spatial
- Lectures are Tuesdays and Thursdays, 11:00 am – 12:15 pm
  - On Zoom first two weeks, then Eaton Humanities 1B80
  - **Tuesdays**: Lecture with slides, notebook, exercises
  - **Thursdays**: Review exercises, visualization critique, weekly assignment prep, weekly quiz
- Canvas is King: announcements on Canvas override syllabus

Evaluation

- **Weekly Assignments**: 15 weeks x 2%/week = 30% total
  - Reinforcing and applying technical skills from the week. Due on Fridays by midnight on Canvas.
- **Weekly Quizzes**: 15 weeks x 1%/week = 15% total
  - Multiple choice quizzes to evaluate readings, lectures, notebooks. Thursdays at end of class.
- **Module Assignments**: 6 modules x 5%/module = 30% total
  - 500–1,000-word blog posts on Medium with figures. Due Tuesdays after end of module.
- **Final Project**: 25% total
  - In-depth and polished write-up with goal of submitting for an op-ed, guest post, *etc*.
- No midterm or final exam

Please submit all assignments on Canvas, emailed assignments will be ignored.

Course Overview

Module	Week	Dates	Computational skill	Communication skill
Shaping	1	Jan 11, Jan 13	Loading	Documentation
	2	Jan 18, Jan 20	Aggregation	Summarization
	3	Jan 25, Jan 27	Joining	Validation
	4	Feb 1, Feb 3	Tidying	Tables
Distribution	5	Feb 8, Feb 10	Histograms	Perception
	6	Feb 15, Feb 17	Box plots	Audience
Comparison	7	Feb 22, Feb 24	Cat plots	Objectives
	8	Mar 1, Mar 3	Faceted plots	Simplicity
Trend	9	Mar 8, Mar 10	Line plots	Trust
	10	Mar 15, Mar 17	Stacked plots	Annotation
	11	Mar 22, Mar 24	Spring Break	
Relationship	12	Mar 29, Mar 31	Scatter plots	Fallacies
	13	Apr 5, Apr 7	Heatmaps	Persuasion
Spatial	14	Apr 12, Apr 14	Choropleths	Conventions
	15	Apr 19, Apr 21	Point plots	Design
Projects	16	Apr 26, Apr 28	Projects	

COVID-19 Contingencies

- Masks required in class and campus COVID-19 policies will be enforced
- If you require sequestration, treatment, convalescence:
  - We will try to accommodate through extensions and incompletes
- If a roommate, partner, or family member's diagnosis will affect you:
  - We will also try to accommodate through extensions and incompletes
- We are bound by and will enforce campus COVID-19 policy

Do not ghost us! Notify us *as soon as possible* of diagnoses or events that will impact your ability to participate in class so we can accommodate without end-of-term scrambling.

Computing Requirements

- Code will be delivered with Jupyter Notebooks of Python 3.9
  - Download Anaconda Individual Edition if you haven't already
  - Update your Anaconda installation. From a Terminal (MacOS) or Anaconda Prompt (Windows)

```
conda update conda
conda install anaconda=2021.11
```

- Readings, code, data will be posted to Canvas: **https://canvas.colorado.edu/courses/79135**
- We will be using pandas, matplotlib, and seaborn but you can explore others if you like
  - Plotly, Altair, Bokeh, etc.
- Students **are not** permitted to use spreadsheet or business intelligence software
  - Excel, Numbers, Tableau, PowerBI

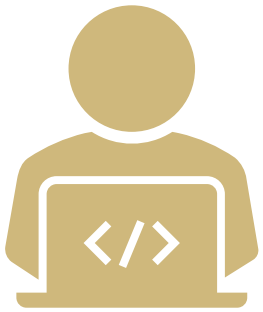
If you cannot reliably access Anaconda and Canvas or don't have a laptop: please contact the instructors immediate to work out an accommodation

Data Science Mindset

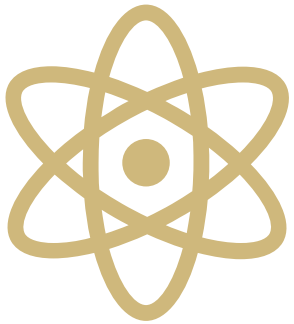
A Modest Proposal

Growth Mindset

Computational Thinking



Hacker Ethic

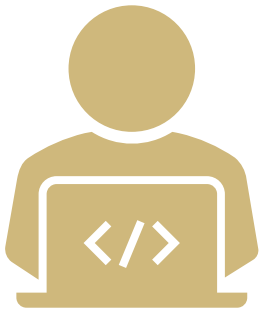


Scientific Norms

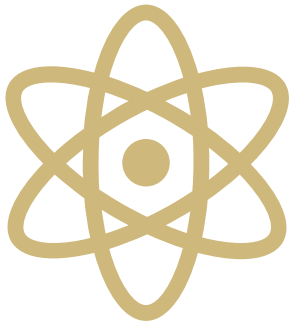
A Modest Proposal

Growth Mindset

Computational Thinking



Hacker Ethic



Scientific Norms

Growth Mindset

- Fixed mindset: fears lack of ability, need to prove ability, challenge-avoiding, stereotype belief/threat, mind as a mold
- Growth mindset: fears lack of effort, need to improve ability, challenge-seeking, resilience, mind as a muscle
- We make these judgments about ourselves and others

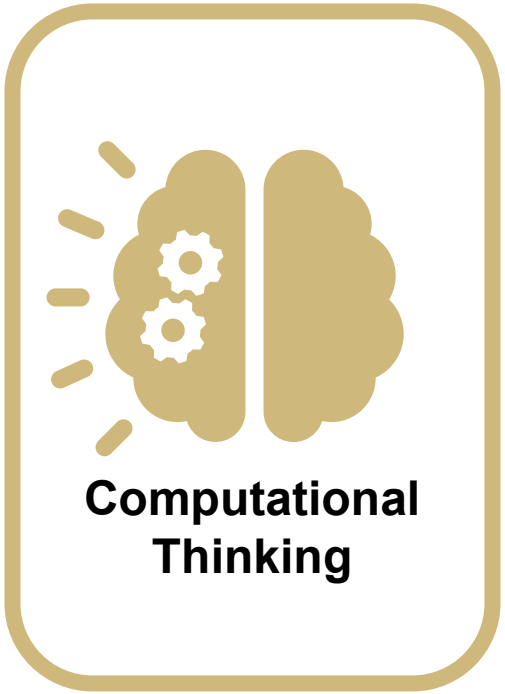
Dweck, C. S., & Yeager, D. S. (2019). Mindsets: A View From Two Eras. *Perspectives on Psychological Science*; Buchanan, A. (2017). The nature of mindsets.

Growth Mindset Questions

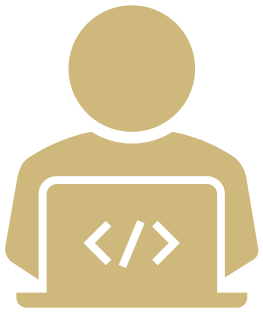
- How will you challenge yourself today?
- What can you learn from a mistake?
- Who can you ask for help?
- How could you make this more interesting?
- What can you do to focus better?
- What's the next challenge to tackle?
- What support resources can you find?
- What else do you want to learn?

A Modest Proposal

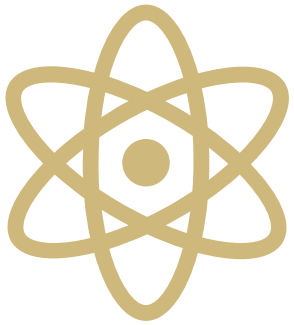
Growth Mindset



Computational Thinking



Hacker Ethic

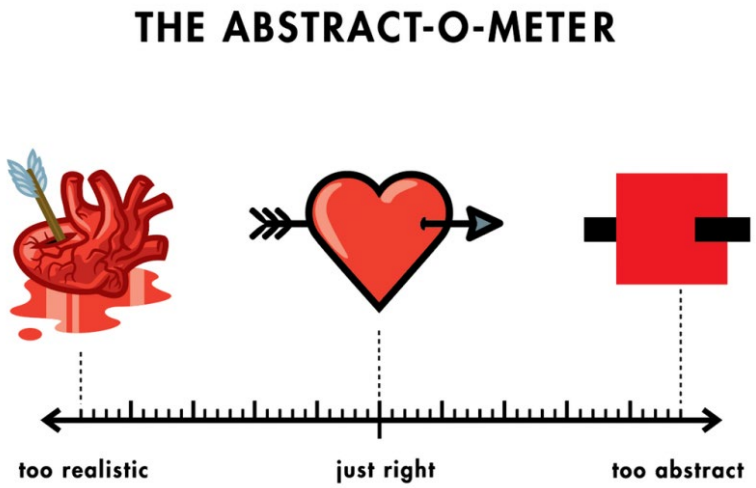


Scientific Norms

Computational Thinking (CT)

- Formulating problems and their solutions that can be executed computationally

- Reformulation, recursion, decomposition, abstraction, testing **[Wing 2006, Grover & Pea 2013]**
- Creativity, algorithmic thinking, cooperativity, critical thinking, problem solving **[Korkmaz, et al. 2017]**
- Data practices, simulation, computational problem solving, systems thinking **[Weintrop, et al. 2016]**
- Decomposition, abstraction, algorithms, debugging, iteration, generalization **[Shute, et al. 2017]**



https://computersciencewiki.org/index.php/File:Abstract_heart.png

Wing, J.M. (2006). Computational thinking. *CACM*; Grover, S., & Pea, R. (2013). Computational Thinking in K–12. *Educational Researcher*; Korkmaz, Ö., et al. (2017). A validity and reliability study of the computational thinking scales. *Computers in Human Behavior*; Shute, V., et al. (2017). Demystifying computational thinking. *Educational Research Review*; Weintrop, D., et al. (2016). Defining Computational Thinking for Mathematics and Science Classrooms. *Journal of Science Education and Technology*.

CT Isn't Just Technical Skills

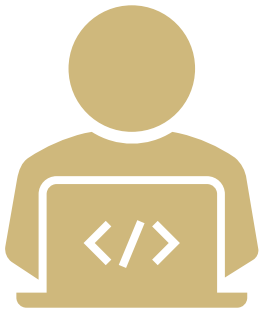
- Concepts: sequences, loops, parallelism, events, conditionals, operators, data
  - This is what INFO 1201 and 2201 are designed for and INFO 3401 and 3402 will continue to reinforce
- Practices
  - Experimenting and iterating: developing, experimenting, and developing some more
  - Testing and debugging: making sure things work and solving problems when they arise
  - Reusing and remixing: building on existing projects or ideas and sharing your own work
  - Abstracting and modularizing: building something complex by putting together smaller parts
- Perspectives
  - Expressing: computation as a medium for creative and critical expression
  - Connecting: computation as a tool for of creating for and interacting with others
  - Questioning: computation as a tool for investigating how the world works

Brennan, K. & Resnick, M. (2012). *New Frameworks for Studying and Assessing the Development of Computational Thinking*.

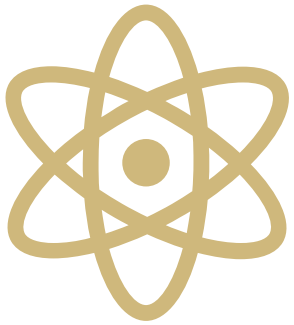
A Modest Proposal

Growth Mindset

Computational Thinking



Hacker Ethic



Scientific Norms

Hacker Ethic

- "Hacking" as a culture of subverting technical systems not cyber-crime
- Common values within hacker ethic **[Levy 1984]**
  - Sharing & openness – access to computers/software/data/information should be free and open
  - Autonomy & meritocracy – hackers should be free to pursue individual goals and judged by their work
  - Creativity & playfulness – novelty and efficiency; irreverence and references to geek culture
  - Decentralization & anti-authoritarianism – build technologies to resist bureaucracy and support freedom
- Exploring then exploiting opportunities by gaining then using know-how
- Moving from curiosity and observation to execution and efficiency
- These values emerged from a culture that remains overwhelmingly white and male **[Coleman 2012]**

Levy, S. (1984). *Hackers: Heroes of the computer revolution*; Coleman, E.G. (2012). *Coding freedom: The ethics and aesthetics of hacking*.

The Hacker Attitude

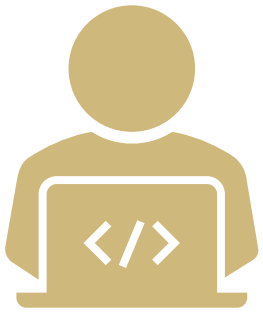
1. **The world is full of fascinating problems waiting to be solved.**
   - “to be a hacker you have to get a basic thrill from solving problems, sharpening your skills, and exercising your intelligence.”
2. **No problem should ever have to be solved twice.**
   - “it's almost a moral duty for you to share information, solve problems and then give the solutions away just so other hackers can solve *new* problems instead of having to perpetually re-address old ones.”
3. **Boredom and drudgery are evil.**
   - “Hackers should never be bored or have to drudge at stupid repetitive work, because when this happens it means they aren't doing what only they can do — solve new problems.”
4. **Freedom is good.**
   - “[Authoritarians] distrust voluntary cooperation and information-sharing. [Hackers] have to develop an instinctive hostility to censorship, secrecy, and the use of force or deception to compel responsible adults.“
5. **Attitude is no substitute for competence.**
   - “Hackers won't let posers waste their time, but they worship competence... at demanding skills that few can master and skills that involve mental acuteness, craft, and concentration is best.”

Raymond, E.S. (2001). How To Become A Hacker.

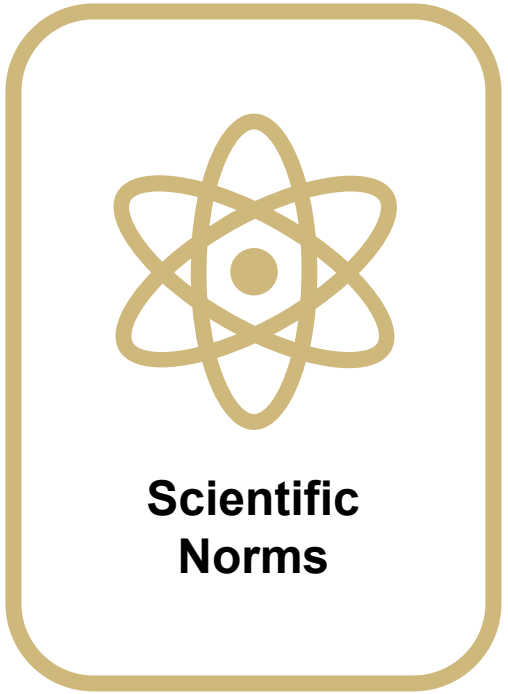
A Modest Proposal

Growth Mindset

Computational Thinking



Hacker Ethic



Scientific Norms

- Merton (1942) proposed four norms that guided scientific culture (CUDOS)
  - Communalism: common ownership of scientific results and methods, duty to share
  - Universalism: scientific work should be evaluated objectively and impersonally
  - Disinterestedness: scientific work should be altruistic and avoid self-enrichment and promotion
  - Organized skepticism: transparency of results, consideration of all evidence, scrutiny by peer review

- Widely-criticized and many counter-examples exist, but they remain influential with new norms
  - Governance: scientists are responsible for direction, control, and self-regulation of science
  - Quality: scientists judge contributions on the basic of quality not quantity
  - Calling: science is a higher purpose worthy of sacrificing material benefits
  - Breadth: scientists fulfill responsibilities like teaching and service in addition to research

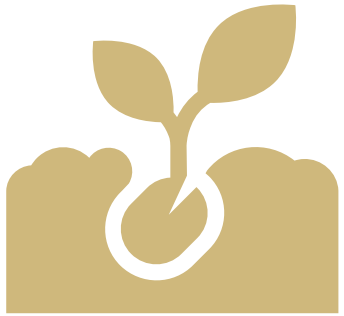
Merton, R. K. (1942). A note on science and democracy. *J. Legal & Pol. Soc.;* Anderson, M. S., et al. (2010). Extending the Mertonian Norms. *Journal of Higher Education*.

Science as a Profession

- Understanding issues of ethics, social responsibility, and awareness of regulations and policies
  - Privacy, property, accessibility, security, human rights
- Communicating results of analyses back to both expert and general audiences
  - Visualization, evaluation, explanation, trust, persuasion, storytelling
- Collaborating to manage complexity and accomplish larger goals
  - Teamwork, brokerage, delegation, translation, infrastructuring
- Management of people and projects
  - Planning, documentation, leadership, conflict resolution, mentorship, consciousness

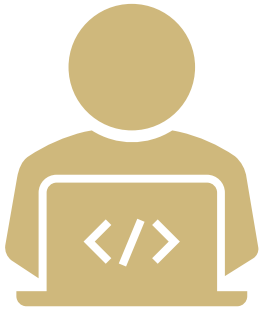
Hall, K.L., et al. (2018). The science of team science. *American Psychologist*; National Academies. (2017). Communicating Science Effectively.

Data Science Mindset Components

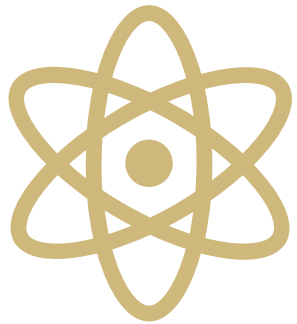


Growth Mindset

Computational Thinking



Hacker Ethic



Scientific Norms

- Growth Mindset: effort not ability, continual improvement, challenge-seeking, resilience
- Computational Thinking: concepts, practices, perspectives of applying computing technologies
- Hacker Ethic: sharing, openness, creativity, autonomy, curiosity, bias towards action
- Scientific Norms: communalism, skepticism, responsibility, communication, collaboration

Documentation and Professionalization

Using Documentation

- Your previous classes may have discouraged using online resources → training wheels are off now!
- Finding, reading, interpreting, and writing documentation are essential skills
  - "Documentation is for 'real' developers, not newbies like me" → **WRONG!**
  - Bookmark documentation for numpy, scipy, pandas, matplotlib, and seaborn
- "It's not working" is not an acceptable request for help
  - What have you tried? What can you get to work? What does the documentation say?

Escalating Issues

- **Examples, tutorials, user guides** → *super* helpful general resources, accessible writing style
  - Common use cases and combining multiple functions, check these first to get oriented
- **Reference or API reference** → details about specific functions & methods, more technical
  - If the tutorials or user guides don't address your issue, look to the specifics for your function, method, *etc*.
- **Question-Answer website** → ask (and answer!) questions on StackOverflow
  - Copy-and-paste error messages, search for a generalized version, *etc*.
- **Developers or Development** → details about contributing code, reporting bugs, early releases
  - Maybe you've encountered a genuine bug or boundary case → see if others have had the same problem
  - Most libraries are developed on GitHub, go to the library's repo, and search under "Issues"

Credit External Resources

- If you use an external resource (documentation, Q&A, blog post, repository/gist) to implement a feature or solve a bug beyond what we've covered in class, just include a link in your code!

```
f,ax = plt.subplots(1,1)

# https://matplotlib.org/stable/tutorials/text/text_intro.html
ax.text(2,6,r'an equation: $E=mc^2$', fontsize=15)
```

- Using advanced functionality that we haven't covered in class and you're not citing is a reliable signal that you're using external resources without credit → just include a link in your code!

- We reserve the right to request a code review for any submitted assignment. If you're unable or unwilling to explain how something was implemented, you could lose all credit on the assignment
  - Repeated violations will be escalated to the Honor Code office
  - Just include a link in your code!

Documenting Your Own Code

- The class notebooks make extensive use of Markdown cells to organize and narrate the analysis

- You should develop a similar practice and style of making clear and well-organized notebooks
  - Importing all libraries in one place, loading data in one place, cleaning data after loading, *etc*.
  - Sectioning (and sub-sectioning!) so different steps are easy-to-find
  - Markdown with narrative of what's happening, hyperlinks to resources/documentation
  - Use hanging indentation to help with legibility instead of cramming into single lines
  - Check out style guides like the Space Telescope Science Institute

Loading Data

Notebook Time!

- Download the “Week 01 – Lecture.ipynb” and “ACS2019_DP05_state.csv” files
  - Please create a dedicated folder for class instead of keeping everything in Downloads, Desktop, *etc*.
  - Put both these files in the same folder
- Open the “Week 01 – Lecture.ipynb” notebook file
  - From Anaconda Prompt (Windows) or Terminal (Mac), navigate to class folder
  - Launch Jupyter Notebook: `jupyter notebook`
- Make sure the first few cells work
- Work on Exercises 1-5, practice consulting documentation and other resources
- No grading on exercises, we’ll cover solutions on Thursday

Weekly Assignment & Next Class

Weekly Assignment 01

- **Skills**: Reviewing how to access, sort, and filter pandas DataFrames
- **Data**: Comparing weather data for Boulder and Broomfield

- Due Friday before midnight on Canvas
  - Save an HTML version of your notebook with all output present
    - File > Download as > HTML (.html)
  - Upload the HTML file to Canvas
  - Cannot upload Notebook files (security and usability)

Next Class

- Review concepts and exercises from last class
- Critique a data narrative or visualization
- Time to work on Weekly Assignment
  - Weekly Assignment due on Friday by submitting to Canvas before midnight
- Weekly quiz at the end of class