



# COMS 2132: ~~Advanced~~ *Intermediate* Computing in Python

Columbia University  
Spring 2026

Dr. Daniel Bauer

# Teaching Team

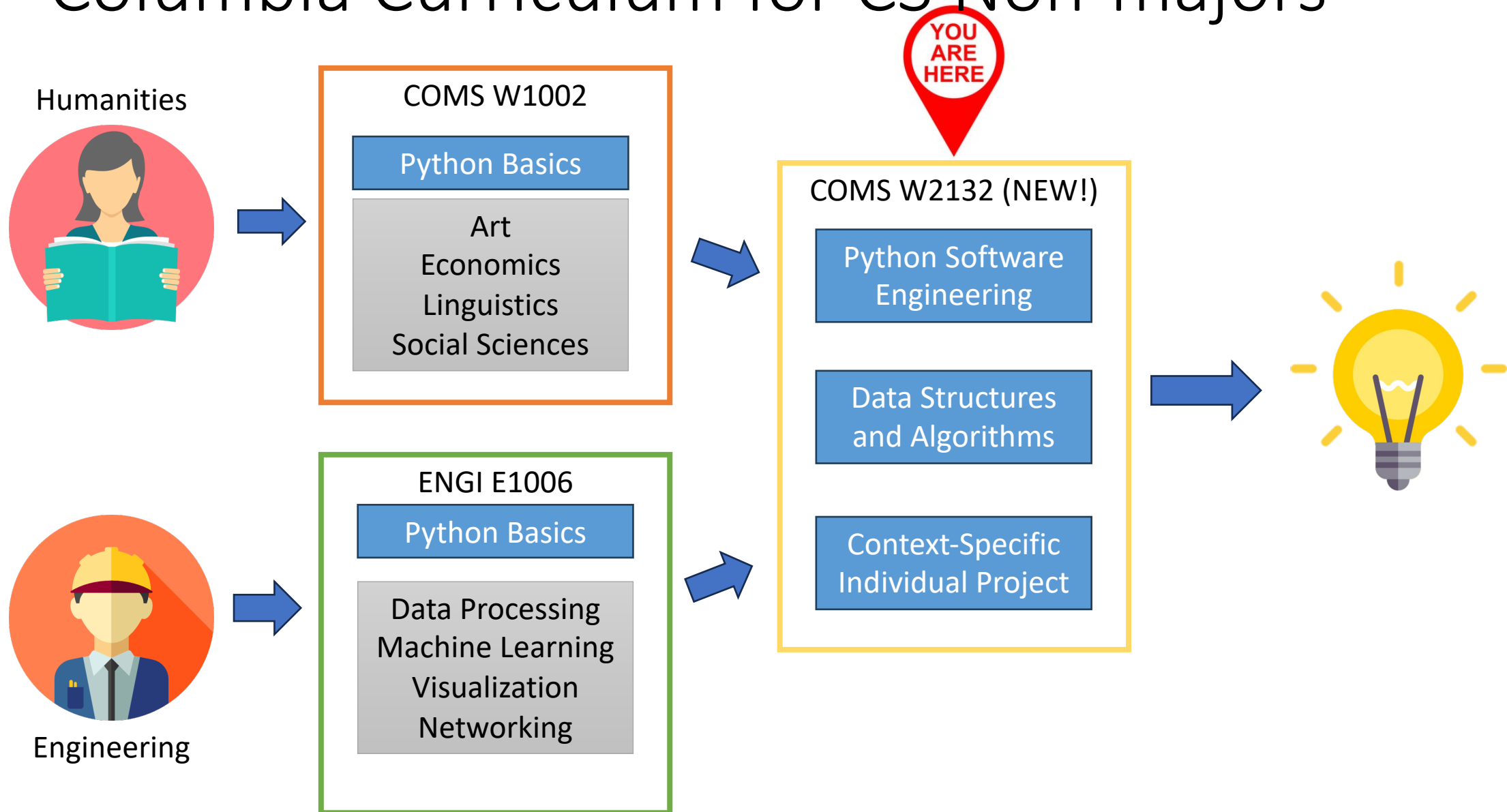
---

- Dr. Daniel Bauer <bauer@cs.columbia.edu>  
(Senior Lecturer in Computer Science)  
<https://www.cs.columbia.edu/~bauer>  
Office: 704 CEPSR, Office Hours Monday 1:15-3:00pm

- Assistants:

Aditi Chowdhuri	anc2207@columbia.edu
Irene Nam	yn2334@columbia.edu
Olivia Huang	oh2251@columbia.edu
Kristy Martinez	kpm2155@columbia.edu
Sophie Pan	sxp2000@columbia.edu

# Columbia Curriculum for CS Non-majors



# What is this Course About?

- “Intermediate course in computing for CS non-majors”
- A follow-up course to COMS W1002 and ENGI E1006
  - You asked for it, we listened...
  - Fill a gap in Columbia’s Python curriculum
- Introduction to Python software engineering
- Essential data structures and algorithms in Python
  - Practical alternative to CS Java data structure classes (COMS 3134 / 3137)
- 4-6-week individual project to deepen your Python skills
  - Learn to design, develop, and publish a Python project

# This Course is NOT About

- Python language basics.
- Rigorous theory of data structures and algorithms.
- Passive learning (you will need to write programs and design / work on your own project)

# Learning Objectives

- By the end of the course, you should be able to:
  - understand essential data structures (linked lists, stacks, queues, trees, graphs) and fundamental algorithm analysis.
  - implement the data structures and corresponding algorithms in Python.
  - use functional and conceptual abstraction to design and incrementally implement a non-trivial software projects.
  - use version control (Git), debugging (PDB, debugpy) and documentation tools (Pydoc, Sphinx).
  - use generative AI productively to aid incremental software development.

# Disclaimer

- This is a relatively new course.
  - Based on requests from COMS W1002 and ENGI E1006 students
- Please expect adjustments, changes, and course corrections
- Feedback is greatly appreciated

Appeal to CS non-majors: Please help us find a model for W2132 that is useful to you!

# Programming / Python Prerequisites

Concepts you should know from COMS W1002 or ENGI E1006

- 1) Names (variables), values, data types (including lists, tuples, dictionaries)
- 2) Statements vs. expressions
- 3) Control Structures (**if**, **else**, **elif** statements, **for** and **while** loops)
- 4) Functions (parameters, return values, function calls)
- 5) Files
- 6) Exceptions
- 7) Classes and instances / objects (basic object-oriented programming)

Review in Lab next week. See topic checklist on Courseworks:

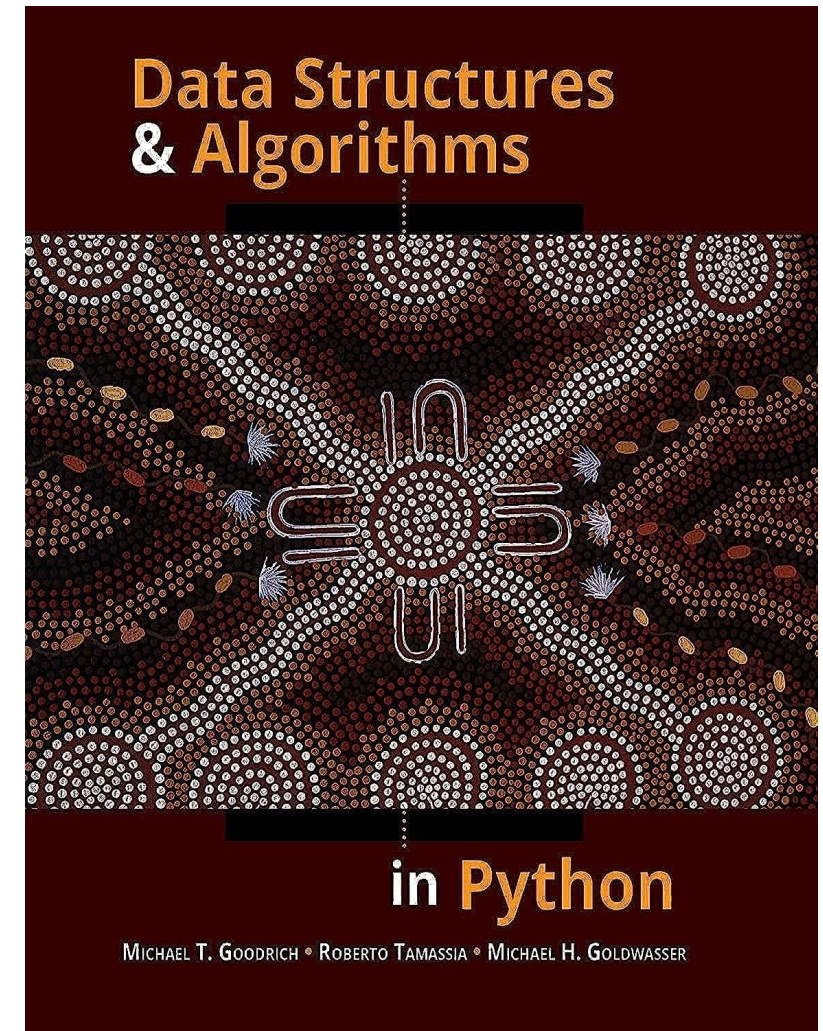
<https://courseworks2.columbia.edu/courses/237696/pages/python-basics-checklist>



# Recommended Textbook

- Excellent textbook for CS non-majors !
- The right level of rigor
- Lectures will mostly follow the textbook
- Recommended but not required
- Columbia Library ebook:

<https://clio.columbia.edu/catalog/ebs2467537e>



Goodrich, Tamassia, Goldwasser, *"Data Structures & Algorithms in Python"*, 1<sup>st</sup> edition, Wiley, 2013

# CourseWorks

- Announcements
- Syllabus
- Important Times & Dates
- Contact Information
- Course Materials
- Homework
- Gradebook

# Getting Help

Computer science and programming can be difficult for people at all levels.

Getting stuck is normal!

**Ask for help early and often!**

- Ed Discussions (online forum)
- TA and Instructor office hours
- Labs Sessions!

# Lectures and Labs

- Lectures: Mon/Wed 10:10am-11:25am, 451 CSB
- Labs (Mandatory, pick one – let me know if neither session works for you):
  - Fri 4:10-5:25pm, 451 CSB
  - Tuesday 6:00pm-7:15pm, 480 CSB (smaller room)
- Both in-person. Attendance mandatory and graded. If you need to miss a session, please let the instructor know.

# Grading

3 individual homework assignments, plus ungraded HW0	24% (8% each)
Midterm Exam (in class, 65 minutes)	15%
Final Exam (120 minutes)	15%
Attendance & Participation	6%
<b>Project</b>	<b>40%</b>

# Individual Homework Assignments

- Mostly first half of the semester
- Will cover data structures and algorithms, programming and theory
- Must be completed individually
- Two-week windows to complete
- Submit through GitHub Classrooms (to be discussed)

# Academic Honesty Policy

- Part 1, collaboration policy:
  - Individual homework assignments and project.
  - You are encouraged to **discuss** classroom material, problems, and homework with other students, the TAs, and the instructor.
  - You can work out the theory together on the whiteboard or paper.
  - **However, you must do all final writing and programming by yourself without further interaction.**
  - Do not share your solutions with others.
  - Do not copy anyone else's work without attribution.

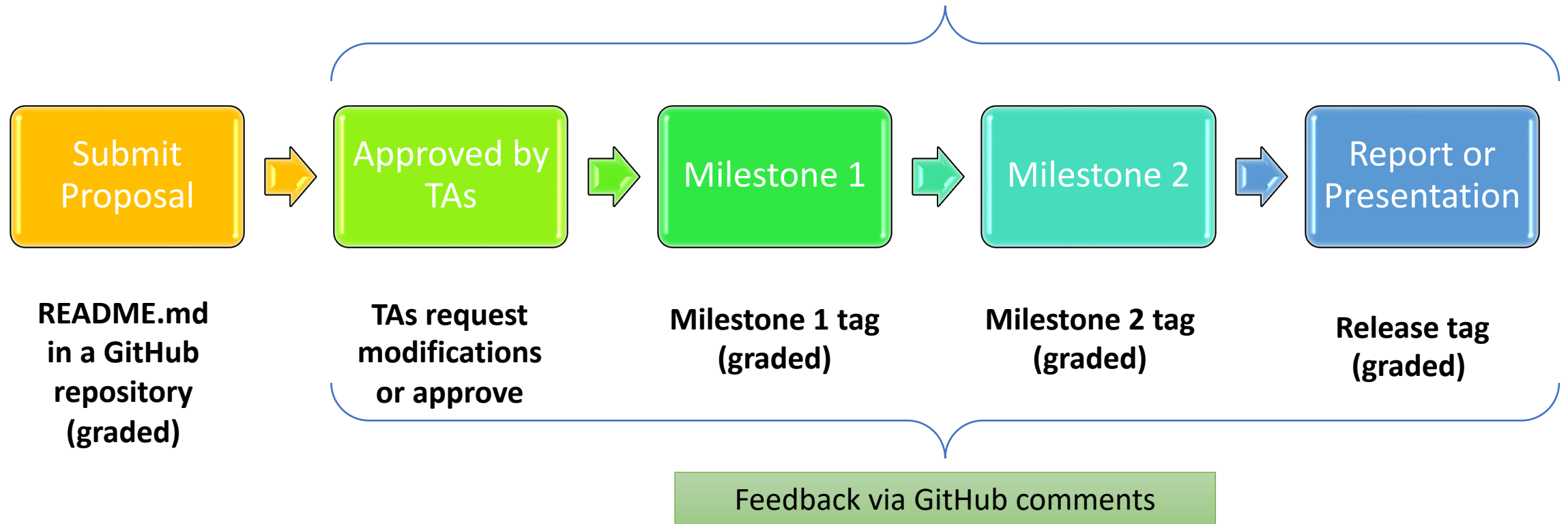
# Academic Honesty Policy Cont'd

- Part 2, AI and online resources policy:
  - LLMs can aid in program development; their use is permitted and encouraged responsibly.
  - Homework: Avoid problems directly using AI unless specifically requested as part of the assignment.
    - We need you to learn the fundamentals. Homework as exam prep.
    - AI can be used for validation/testing.
  - Projects: Responsible AI use encouraged (do more with less!)
  - You must attribute **any** code taken from an online source (full URL, AI prompt, ...).

See syllabus for details!



# Project: Workflow on GitHub



Similar to typical open-source software development workflow on GitHub

# Project Proposal

- Short README.md addressing the following:
  - 1) What are you making?
  - 2) Why is it important, relevant or interesting?
  - 3) Data structures, tools, libraries, or services you intend to use
  - 4) Proposed milestones
  - 5) Optional: How do you want to publish your work?

*We will provide an example when the time comes*

# How to Pick a Project?

- Try to think of a problem that is fun or important to you
- Perhaps there is a tool or a script that you have always wanted?
- Maybe you have an idea for a web service or website?
- Maybe you have a chore that could be automated?
- If you have a hobby, can you use Python to get better at it?
- There are no bad ideas (only undeveloped). If it is something you want, others probably want it too.

# Pick a Project Area (examples)

- Computational linguistics / NLP
- Computational biology
- Social sciences
- AI and machine learning
- Systems, networking, Internet
- Data analysis and databases
- Computing in the arts

# Pick a Project Area (continued)

- **Computational linguistics / NLP**
  - understand or respond to text, summarization, information extraction, dialog systems, ...
- Computational biology
- Social sciences
- AI and machine learning
- Systems, networking, Internet
- Data analysis and databases
- Computing in the arts

# Pick a Project Area (continued)

- Computational linguistics / NLP
- **Computational biology**
  - Simulations and modeling of biological systems
- Social sciences
- AI and machine learning
- Systems, networking, Internet
- Data analysis and databases
- Computing in the arts

# Pick a Project Area (continued)

- Computational linguistics / NLP
- Computational biology
- **Social sciences**
  - Economics, political science (analyze datasets), education, environmental studies, law
- AI and machine learning
- Systems, networking, Internet
- Data analysis and databases
- Computing in the arts

# Pick a Project Area (continued)

- Computational linguistics / NLP
- Computational biology
- Social sciences
- **AI and machine learning**
  - Build AI enabled applications. Experiment with some new dataset. Fine-tune existing models for new applications (e.g. computer vision), ...
- Systems, networking, Internet
- Data analysis and databases
- Computing in the arts



# Pick a Project Area (continued)

- Computational linguistics / NLP
- Computational biology
- Social sciences
- AI and machine learning
- **Systems, networking, Internet**
  - Internet services, websites, APIs, network analysis, robotics, IoT (Raspberry Pi?).
- Data analysis and databases
- Computing in the arts

# Pick a Project Area (continued)

- Computational linguistics / NLP
- Computational biology
- Social sciences
- AI and machine learning
- Systems, networking, Internet
- **Data analysis and databases**
  - Analyze or visualize public datasets, Google BigQuery, and Datalab
- Computing in the arts

# Pick a Project Area (continued)

- Computational linguistics / NLP
- Computational biology
- Social sciences
- AI and machine learning
- Systems, networking, Internet
- Data analysis and databases
- **Computing in the arts**
  - Video / Audio demo, algorithmic art, games

# Projects by Application Type

- 1) Command line tool
- 2) GUI program (TkIntern, PyGame, PyQt, Web UI)
- 3) Internet service (API) or website
- 4) Jupyter Notebook
- 5) Embedded program (Raspberry Pi)

# Some Pointers for Inspiration

- Peter Norvig's Jupyter Notebooks: <https://github.com/norvig/pytudes>
- Data science: <https://www.dataquest.io/path/data-analyst/>
- Machine learning: <https://scikit-learn.org/stable/index.html>
- Building websites with Django: <https://www.tangowithdjango.com/>
- Making games with Pygame: <https://www.pygame.org/wiki/tutorials>
- Learning robotics using Python (book): <https://www.amazon.com/dp/B00YEVZ6UK>
- Programming the Raspberry Pi (book): <https://www.adafruit.com/product/1089>

*We will also offer a library of more concrete ideas later this semester*