Introduction to:
Introduction to:
Computer Science
with a Focus
on Algorithms
and Writing Code
Adam Meyers
New York University
Outline

• What is computer science?
• What is an algorithm?
• How do we use Python?
• By 2025, why should all newly educated people know how to write computer programs?
  – Alternatively, do you believe that learning to program is unnecessary for the general population? I.e., should programming be left to specialists?
• Homework Assignment
Some Quick Answers

• A (modern) **computer** is:
  – A mechanical device that makes calculations and solves problems, consisting of physical components, aka **hardware**. Examples: main frame, work station, (cluster/farm), desktop, laptop, tablet computer, cell phone, smart watch, gps, game console, arduino, etc.

• A **computer program** (software) is:
  – Sets of instructions which a computer can “interpret” to solve problems, make calculations, perform tasks, etc.

• A **programming language** (like Python) is:
  – A formal “language” that humans use to write programs.

• **Computer Science** is a field of study concerning:
  – The design of computer hardware and software
  – Modelling problems and using models to find solutions
What is an Algorithm?

• A precise “recipe” for solving a type of problem

• A crude formalization of the PBJS Algorithm
  – Transfer **slice** of bread from loaf to plate
  – Repeat until enough peanut butter
    • Put knife into peanut butter jar and get peanut butter
    • Transfer peanut butter from knife to slice of bread
  – Transfer **other slice** of bread from loaf to plate
  – Repeat until enough jelly
    • Put knife into jelly jar and get jelly
    • Transfer jelly from knife to **other slice** of bread
  – Put **slice** of bread (pb side down) on **other slice** of bread
Program = Implementation of Algorithm

- The PBJS instructions in ACME sandwich maker 2000 implements the PBJS algorithm
- The instructions refer to specific levers, switches and gears that are part of the ACME sandwich maker 2000
- The instructions refer to specific types of knives, plates, breads, and jellies
- Other PBJS instructions use different, but mathematically equivalent methods
  - The PJ-maker Deluxe squeezes peanut butter out of a squeezy bottle and does not use a knife, but otherwise follows the same algorithm.
Sandwich Making Algorithms Online

- https://randazza.files.wordpress.com/2009/04/flowchart_1.jpg

ETC

- Most sandwich making algorithms online are the same
  - They involve getting ingredients and building a sandwich in essentially the same order
  - The McDonald's one (the 2nd one) deals with multiple kinds of sandwiches
  - There are also some presentations of sandwich making algorithms designed to argue for the benefits of certain CS principles (object oriented, etc.)
Writing Algorithms

• Design algorithms independently of programs
  – Focus on “best” way to solve a problem
  – Ignore punctuation, syntax, etc. of programming language

• Use *Pseudo Code* (previous PBJ algorithm slide)

• A *Flow Chart* (next slide)

• One algorithm can be implemented as a computer program in many different computer languages
Flow Chart for PBJ algorithm

Start

Get slice of bread from loaf and put it on the plate.

Enough Peanut Butter?

No

Get some peanut butter and spread it on slice of bread

Yes

Get 2nd slice of bread and put it on the plate.

Enough Jelly?

Yes

Put slice of bread (pb-side down) on 2nd slice

No

Get some jelly and spread it on slice of bread

End
Some Flowchart Conventions

- Circles/Ovals are for Start and End
- Rectangles represent steps in processing
- Diamonds represent decision points
  - Yes or No; Multiple Choice; ...
- Arrows show the sequence of steps
  - Looped arrows represent steps that can repeat, typically until some condition is met

More details: https://en.wikipedia.org/wiki/Flowchart
Program Structure

• There are choice points (or *if statements*)
  – Checks for true/false or yes/no
    • If Yes, one thing happens
    • If No, something else happens

• There are loops
  – There are events that keep occurring over and over
  – Usually, there is a choice point in the loop
    • One condition means that we exit the loop
    • The other condition means that we repeat the loop one more time
Algorithms Predate Computer Science

• The word *algorithm*: 12th century based on 9th century Persian mathematician Muhammad ibn Mūsā al-Khwārizmī, whose name is also the basis of the word *algebra*

• Formal processes in science/math have been described this way for centuries

• Music algorithms: 10th Century to Present (Guido of Arezzo, J. Bach, J. Cage, M. Feldman, Brian Eno, Nine-Inch Nails)

• Art algorithms: 8th century Islamic tile patterns, M. Escher, Sol Lewitt and modern computer artists

• Algorithms implemented on computers are not subject to some human limitations (e.g., speed of computation, memory size, emotional bias), but are more limited in other ways (e.g., imagination, intuition, real world knowledge)
More about Algorithms?

• Kahn Academy has interesting material:
  – https://www.khanacademy.org/computing/computer-science/algorithms
  – First video is pretty good
  – The rest is interesting, but quickly gets more complex than we will get in this class
Working with Models on Computers

• A model is a representation of something
  – A plastic model may represent an actual car
  – The letters *cat* may be a model of the sound |kæt|
  – The word “cat” (sound, letters, ...) can represent a cat in some model (e.g., taxonomy of mammals)
  – A set of formulas may be a model of gravity

• People use computer programs to work with models
  – To model scientific theories, processes, events, finance, audio-visual, games, etc.
  – Input/Output can be symbols, pictures, sound, etc.
  – Input/Output can be instructions from/to other machines
Computers are General Problem Solving Devices

- Scientists can test theories faster than any time in history
- Artists can implement ideas that were inconceivable fifty years ago
- In particular, tedious parts of all problems can be easily formulated as computer programs
- Many hard problems can be broken down into smaller simpler problems
- Programming is a game-changing innovation
Anyone Who Has Ever Played Civilization Knows...

• Literacy and Writing were game changers
  – They were necessary precursors to abstract thinking
• Like literacy, computer programming provides new ways to model parts of our world
• Currently, most newly trained scientists spend about ½ of their time writing computer programs
• Computer programming is also extremely important in law, medicine, art and other areas
• Prediction: In 10–20 years, being properly educated will include knowing how to program
Why Should Everyone Know How to Program?

• Knowing how to program and how programs work gives you:
  – Understanding of what current software can do
  – Understanding of what programs are possible to write
  – Understanding of how to solve new types of problems
  – Important for writing code, commissioning others to write code and/or purchasing software.

• Why not just use a computer like an appliance?
  – Can only use existing software, e.g., basic entertainment, office work, basic art, etc.
  – Can only solve problems within narrow parameters compatible with previously designed programs
What you should learn in this class

• How to write algorithms to solve simple problems
• How to write simple programs in Python
• What programming is and why it is important
• And possibly…
  – You will start thinking about how lots of problems could be solved by writing computer programs
    • problems in a hobby, area you are expert in, your job, a major or minor, etc.
  – IMHO, the best way to learn to program well is:
    • Find an interesting problem that you care about
    • Write a program to solve that problem or part of that problem
Homework due on day of 3rd Lecture

- Go to Class Website in HW Section:
  - http://cs.nyu.edu/courses/fall18/CSCI-UA.0002-007/#Homework%20Schedule
  - Modules and Readings due at the Beginning of the Day
    - Quiz Deadline Strictly Enforced (Small amount of leeway for 1st 2 quizzes due to add/drop)
      - Purpose is to Prepare for Lecture
      - Each quiz separately has almost no actual effect on grade
        » Each test is worth .1 * 5% = .5% of grade
        » Difference between 2 letter grades (e.g., B+ and A-) is .3/4 = 7.5%
        » 1 quiz = 1/15 of the difference between a B+ and an A-
      - All the quizzes together could make a difference (5% is nearly 7.5%)
  - Programming Assignment due at the end of the Day
    - No formal late penalty
    - Natural Consequences as discussed
  - In other words, no reason to ask permission to do HW 1 hour late
    - For quiz, it doesn't matter – the module is more important than the quiz and missing a few quizzes is unlikely to effect your final grade
    - For HW, it doesn't matter
    - Do it on time to gain better understanding, don't worry about the grading