WHAT HAVE WE ACCOMPLISHED IN 6.00.1X?

WRAPPING IT ALL UP

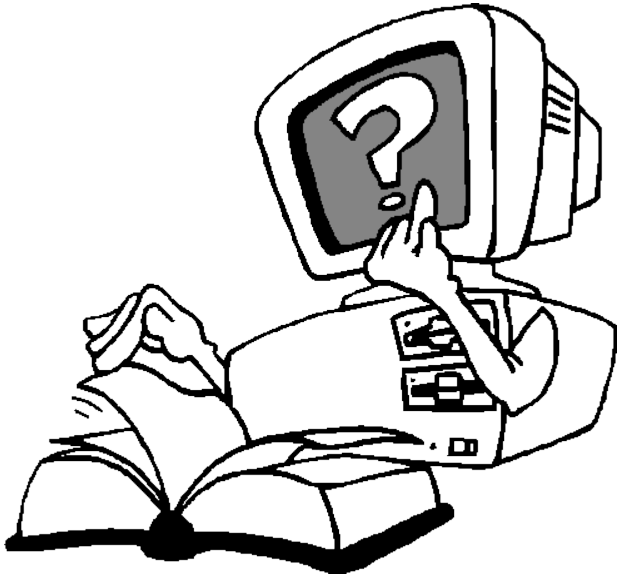
- where have you been?
  - what are the key topics learned in this course?
  - what are the key lessons to take from this course?
- where are you headed?
  - how might you use the knowledge you have gained?
  - what are next steps in enhancing your knowledge of computation?

TOPICS (from Lecture 1)

- ✓ represent knowledge with **data structures**
- ✓ **iteration and recursion** as computational metaphors
- ✓ **abstraction** of procedures and data types
- ✓ **organize and modularize** systems using object classes and methods
- ✓ different classes of **algorithms**, searching and sorting
- ✓ **complexity** of algorithms

OVERVIEW OF COURSE (from Lecture 1)

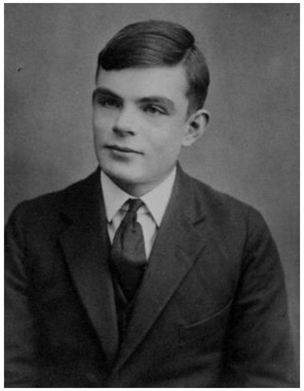
- ✓ learn computational modes of thinking
- ✓ master the art of computational problem solving
- ✓ make computers do what you want them to do



Hope we have started you down the path to being able to think and act like a computer scientist

WHAT DO COMPUTER SCIENTISTS DO?

- they think computationally
  - abstractions, algorithms, automated execution
- just like the three r's: reading, 'riting, and 'rithmetic – computational thinking is becoming a fundamental skill that every well-educated person will need


Alan Turing

Ada Lovelace

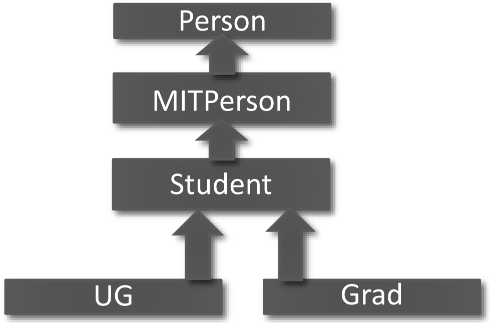
I ♥ 6.00.1x

COMPUTATIONAL THINKING: THE PROCESS

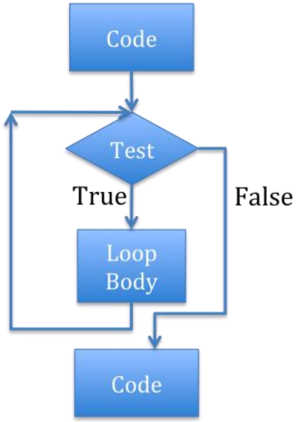
- identify or invent useful abstractions
  - suppressing details, formulating interfaces
- formulate solution to a problem as a computational experiment using abstractions
- design and construct a sufficiently efficient implementation of experiment
- validate experimental setup (i.e., debug it)
- run experiment
- evaluate results of experiment
- repeat as needed

THE THREE A’S OF COMPUTATIONAL THINKING

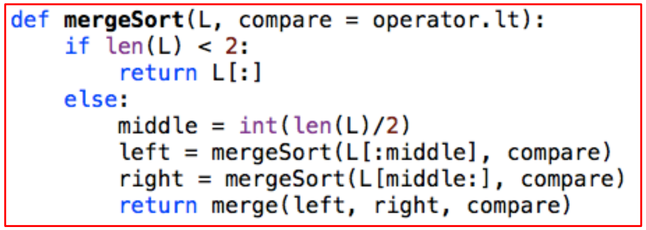
- abstraction
  - choosing the right abstractions
  - operating in multiple layers of abstraction simultaneously
  - defining the relationships between the abstraction layers



- automation
  - think in terms of mechanizing our abstractions
  - mechanization is possible – because we have precise and exacting notations and models; and because there is some “machine” that can interpret our notations



- algorithms
  - language for describing automated processes
  - also allows abstraction of details
  - language for communicating ideas & processes

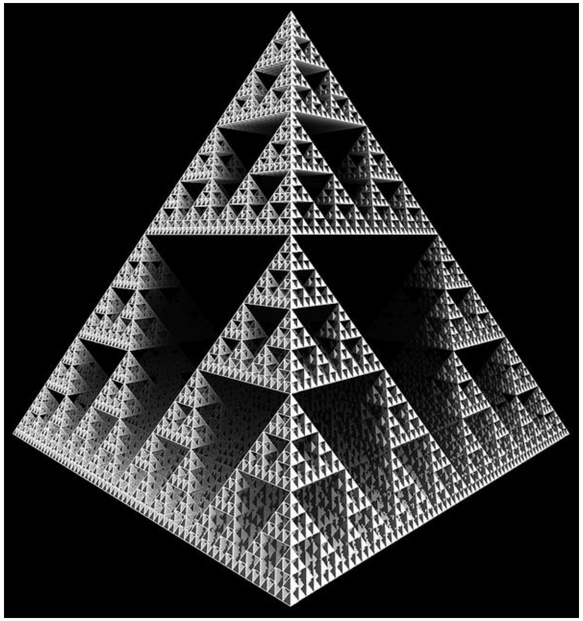


```
def mergeSort(L, compare = operator.lt):
    if len(L) < 2:
        return L[:]
    else:
        middle = int(len(L)/2)
        left = mergeSort(L[:middle], compare)
        right = mergeSort(L[middle:], compare)
        return merge(left, right, compare)
```

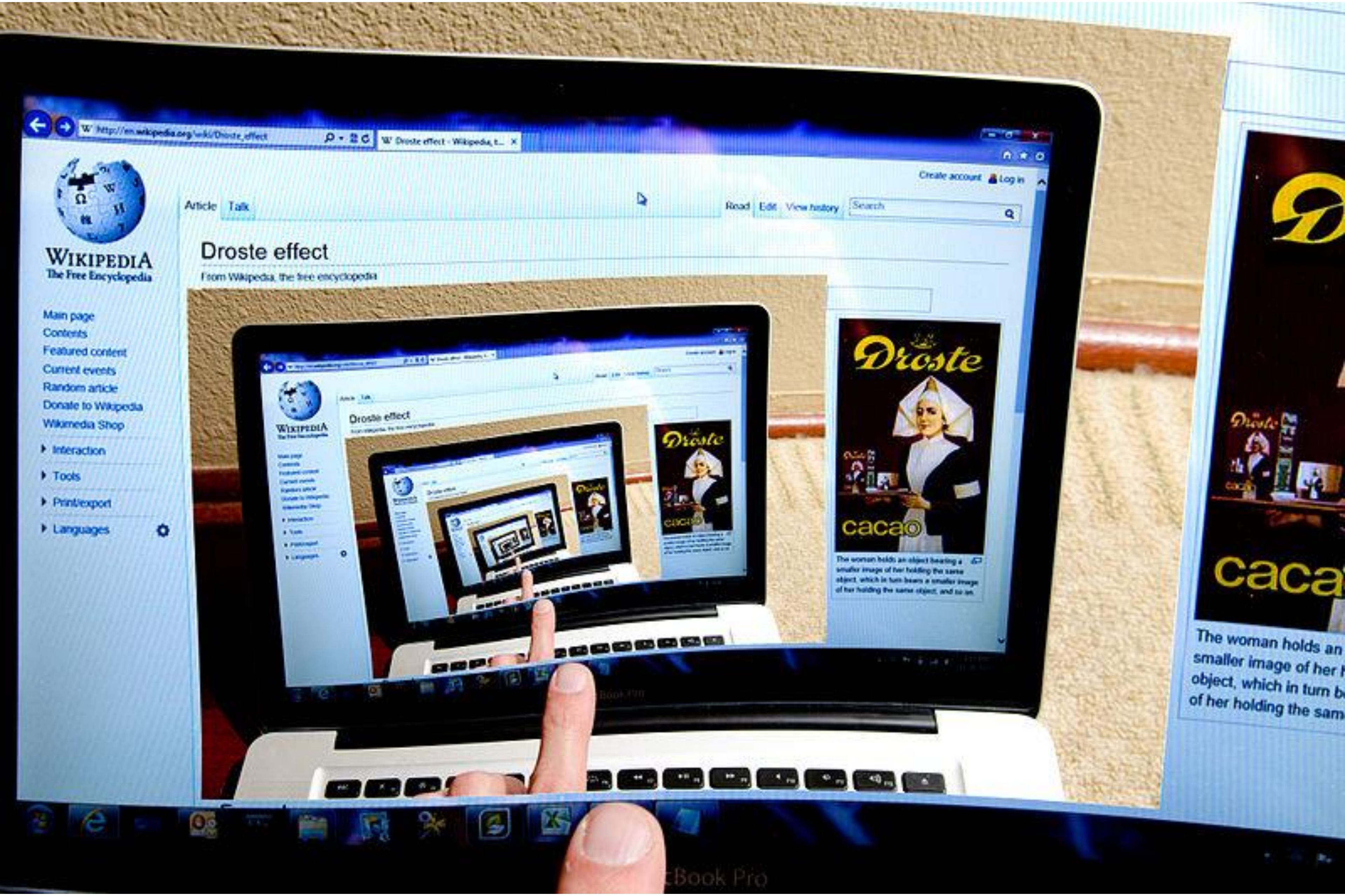
ASPECTS OF COMPUTATIONAL THINKING

- how difficult is this problem and how best can I solve it?
  - theoretical computer science gives precise meaning to these and related questions and their answers
- thinking recursively
  - reformulating a seemingly difficult problem into one which we know how to solve
  - reduction, embedding, transformation, simulation

$O(\log n)$; $O(n)$;
$O(n \log n)$;
$O(n^2)$; $O(c^n)$



CC-BY Image Courtesy of Robson#



Droste effect

From Wikipedia, the free encyclopedia

The woman holds an object bearing a smaller image of her holding the same object, which in turn bears a smaller image of her holding the same object, and so on.

WRAPPING IT ALL UP

- where have you been?
  - what are the key topics learned in this course?
  - what are the key lessons to take from this course?

- where are you headed?
  - how might you use the knowledge you have gained?
  - what are next steps in enhancing your knowledge of computation?

NEXT STEPS

- look for ways to apply what you have learned:
  - can you use algorithmic approaches in your professional life?
    - how might abstraction, or computational experiments, be used to improve what you do for your job?
    - if you are a student, how can these ideas help you pursue your choice of discipline more effectively?
  - can you use algorithmic approaches in your personal or family life?
    - organizing your personal finance records, your family historical records

NEXT STEPS

- consider taking another course in computation
  - 6.00.2x – Introduction to Computational Thinking and Data Science
  - a course in algorithm design
  - a course in software engineering
  - a course in machine learning
  - a course in data analytics and data storage
  - a course in …

GOOD LUCK!

- however you choose to use computational thinking, we hope that it becomes a useful tool for you:
  - as a way of approaching professional problems
    - e.g., running computational experiment to simulate physical or biological or financial or other problems
  - as a basis for understanding the impact of computation in everyday life
    - e.g., what is the power of machine learning methods in solving complex problems
  - as a language for communicating ideas
    - e.g., explaining ideas as concise steps in an algorithmic process, independent of whether one actually implements it

By OsamaK (Own work) [Public domain], via Wikimedia Commons