Designing Algorithms and Programs

- Designing algorithms to be correct and efficient
  - The most important of these is ____________
  - When do we need to worry about efficiency?
  - Example: finding a number between 1 and 1000
    - High, Low, Correct: how many guesses?
    - Same algorithm can find an element in a sorted list

- Python searching in dictionary, set, list
  - How can we find an element?
  - How long does it take?

From Algorithms to Programs

- Coping with simulation of one million molecules
  - Or thousands of points, or nucleotides or ...
  - Suppose each has (x,y) and velocity which is (vx,vy)
    - Might have other attributes too

- We could use four lists
  - X[i], Y[i] represent ____________
  - Vx[i], and Vy[i] represent ____________
  - Color[i] and ...
  - Processing each molecule is very cumbersome

Objects (and classes) to the rescue

- Encapsulate the state of a molecule together
  - One entity that *encapsulates* the state together

- Class is the structure combining state/behavior
  - Class is an object factory, we use it to create objects
  - Each object is an *instance* of the class, e.g., molecule
  - Functions in an object manipulate the state
    - In a class/object these functions are often called *methods*

- Python allows programmers to use objects
  - Java requires programmers to use objects

Fran Allen

- IBM Fellow, Turing Award
  - Optimizing compilers

- Taught high school for two years, then Master’s degree and IBM
  - Teachers excited me to learn

I’ve always felt that theory without practice is maybe nice and maybe pretty, but it’s not going to influence computing as much as the practice side. But the practice has to be backed up with the ability to talk about it, reason about it, and formulate it so that it can be reproduced.
Classes and Objects in Python

- Each object knows about itself
  - Uses `self` to refer to itself, to pass itself, to access data
  - Why `self.value` similar to `global value`, but better?

- Method names (function names) mean something
  - `__init__`, `__str__`, `__mul__`, `__imul__`
  - See `BallSimulate.py`

- How does ball move? How does simulation work?
  - How is program launched?

Forensic Analysis of `Ballsimulation.py`

- Where is collision of objects detected?
  - What happens when worlds collide?

- How is simulation run, what is the time step?
  - Where do you look for this? How do you find it?

- Where are objects created? How are they created?
  - Where do you look for this, how is it identified?

- How can we make the ‘container’ work?
  - Experiment with more or fewer objects?