6.009 Lecture 2 –
Decomposing a Program into Functions

Lecturer: Adam Chlipala
September 12, 2017
MIT

Debugging: The Secret Essence of Programming

“By June 1949 people had begun to realize that it was not so easy to get programs right as at one time appeared.

[...] the realization came over me with full force that a good part of the remainder of my life was going to be spent in finding errors in my own programs.”

Maurice Wilkes, *Memoirs of a Computer Pioneer*, MIT Press, 1985, p. 145.

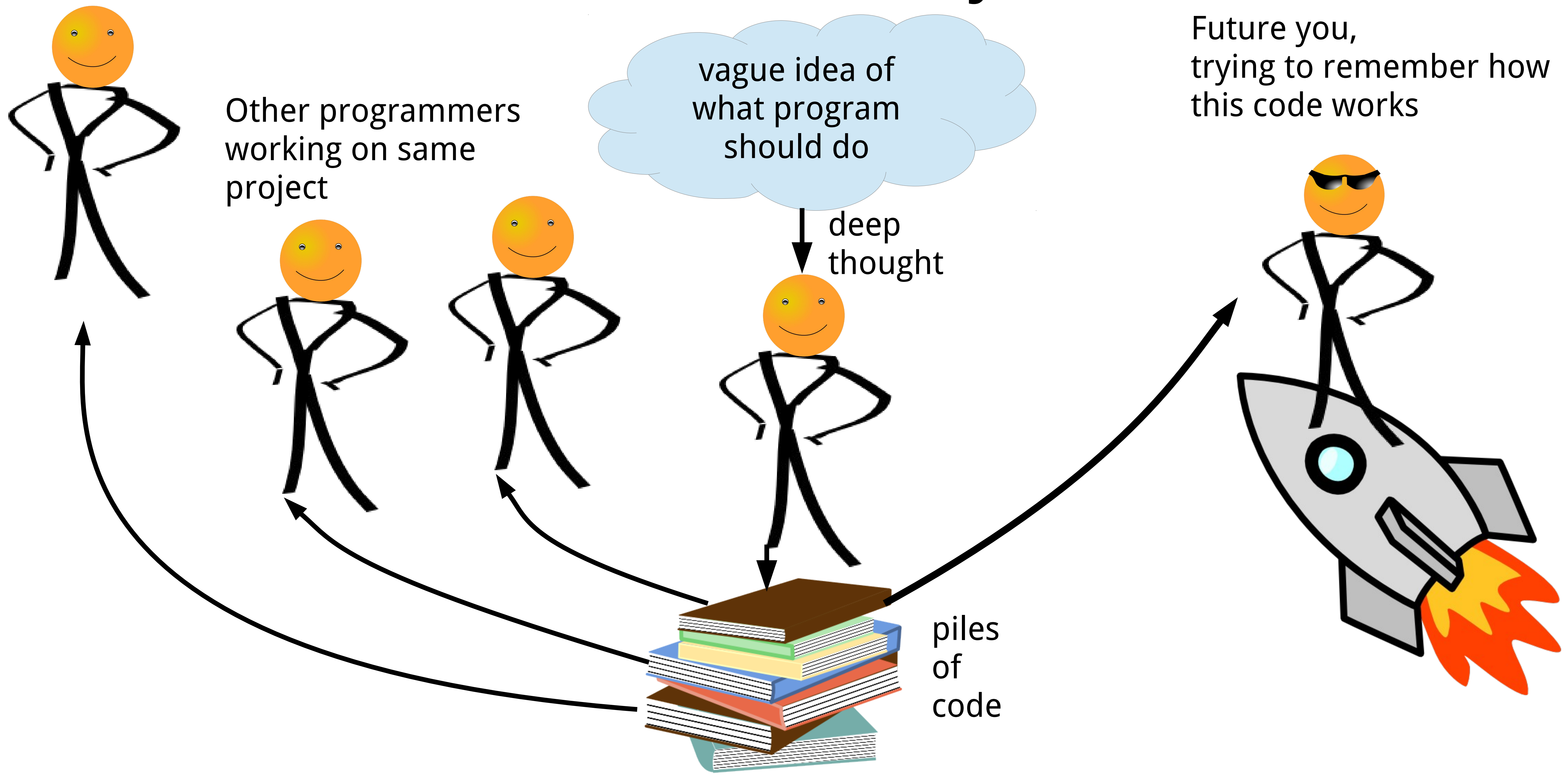
"EDSAC (9)". Licensed under CC BY 2.0 via Commons - https://commons.wikimedia.org/wiki/File:EDSAC_(9).jpg#/media/File:EDSAC_(9).jpg

The Software Crisis

“There is a **widening gap between ambitions and achievements in software engineering**. This gap appears in several dimensions: between promises to users and performance achieved by software, between what seems to be ultimately possible and what is achievable now and between estimates of software costs and expenditures. The gap is **arising at a time when the consequences of software failure in all its aspects are becoming increasingly serious.** Particularly alarming is the seemingly **unavoidable fallibility of large software**, since a malfunction in an advanced hardware-software system can be a matter of life and death, not only for individuals, but also for vehicles carrying hundreds of people and ultimately for nations as well.”

- David and Fraser, quoted in report of <u>**1968**</u> Software Engineering conference

Software Lifecycle

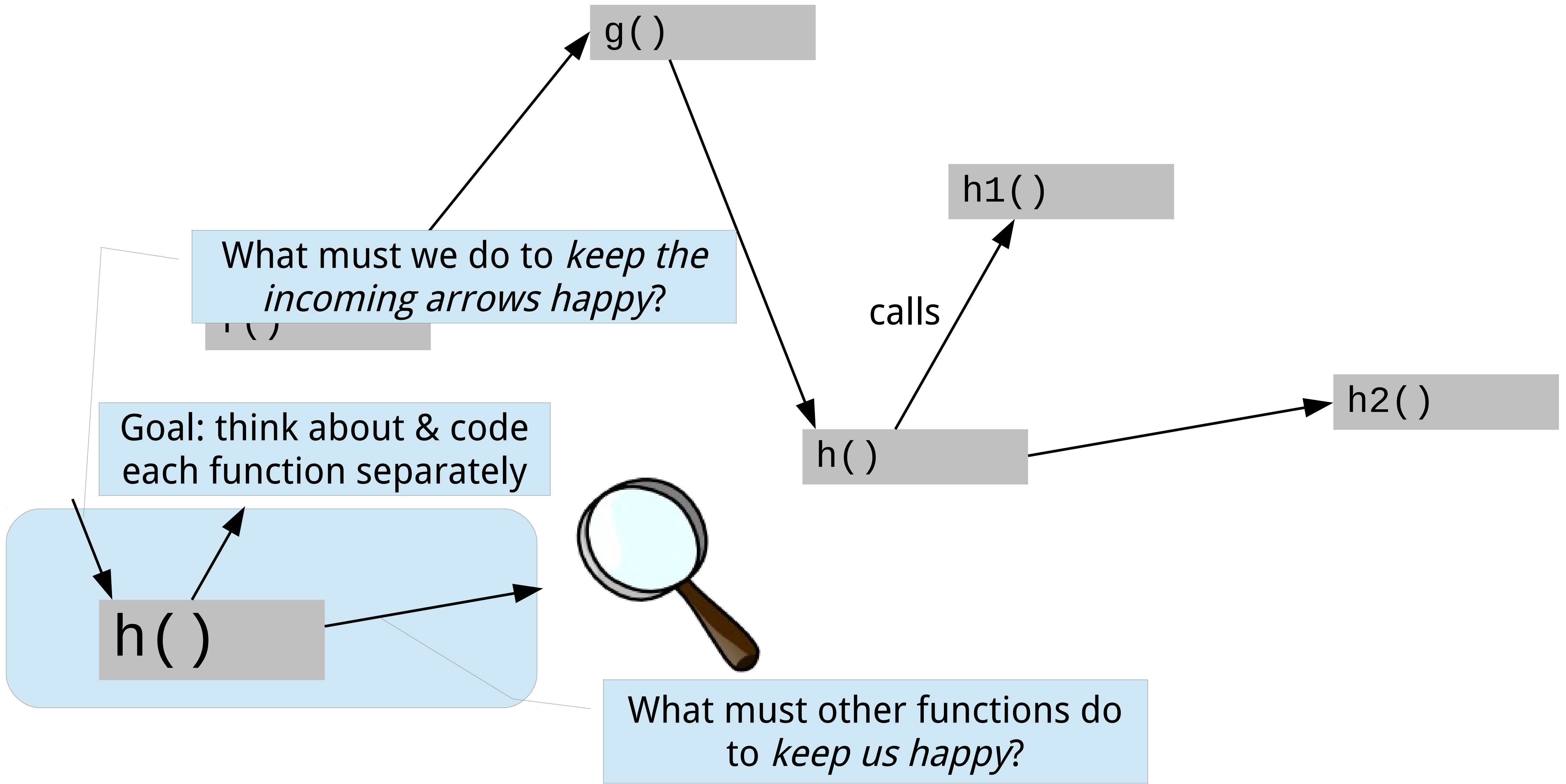

Other programmers working on same project
vague idea of what program should do
deep thought
Future you, trying to remember how this code works
piles of code

Code is Written to be *Read*

In the long term, what matters most about code is how easy it is for someone to pick it up and use it in a surprising way

or modify it.

Functional Decomposition



What must we do to *keep the incoming arrows happy*?

Goal: think about & code each function separately

What must other functions do to *keep us happy*?

Function Interfaces Matter

In a well-designed program, each function has an easy-to-understand description of what it *expects from* and *provides to* the rest of the code.

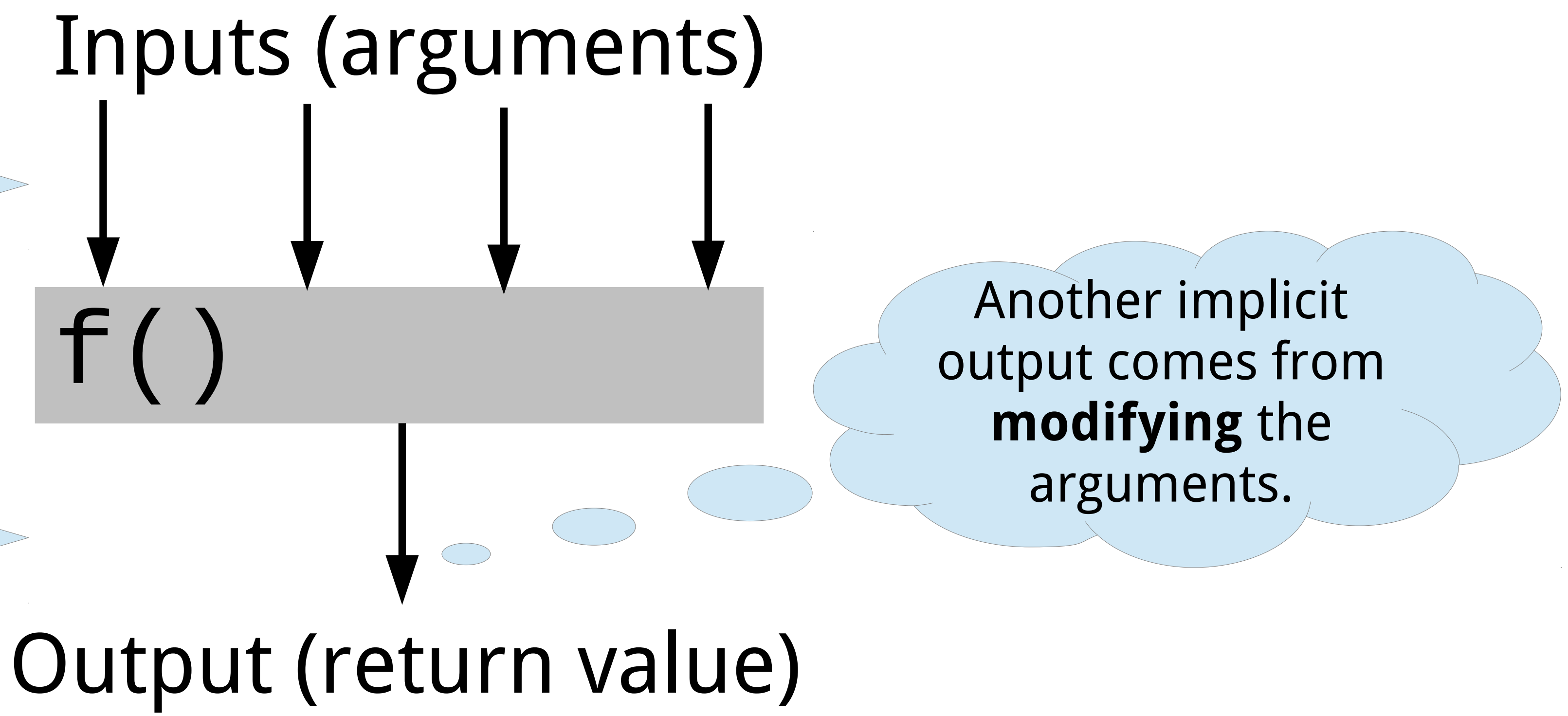
The Classic Approach: Preconditions & Postconditions

Precondition

A **requirement** on *inputs*.
Not allowed to call otherwise!

Postcondition

A **promise** about *outputs*.
Author of `f()` is responsible for maintaining it.

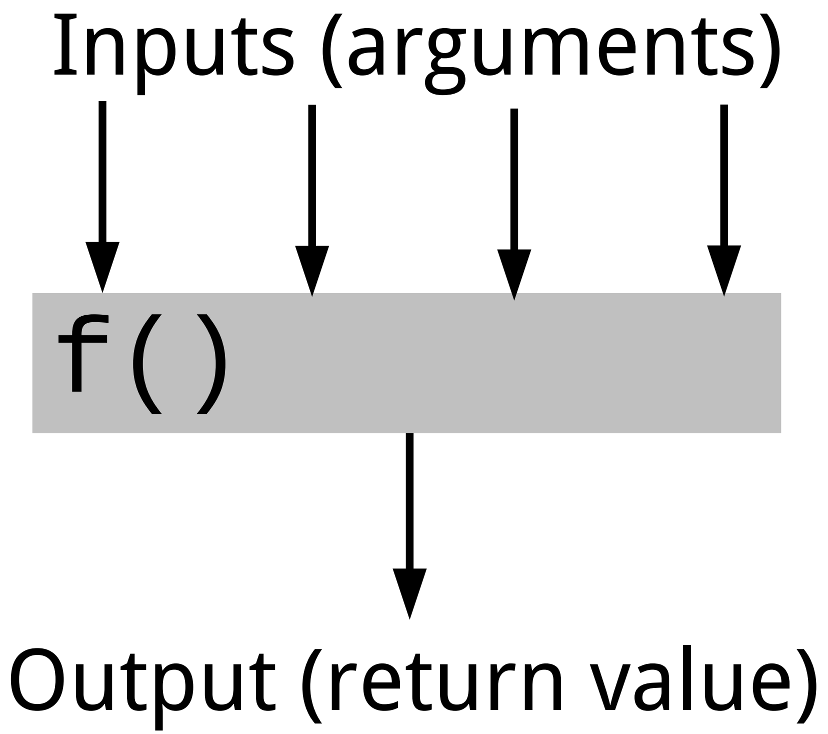


Another implicit output comes from **modifying** the arguments.

Most programs (in Python and other languages) have preconditions & postconditions expressed in *comments* or other documentation for people.

The More Newfangled Approach: Systematic Testing

List of test cases:
f(1, 2, 3) == 4
f(2, 1, 3) == 7
f(8, 8, 8) == 8
…



Function-at-a-time tests are called **unit tests**.

Test cases are present in the code and are actually run, to find bugs.

Let's Write Some Code....

I'll be using **Emacs**, my preferred development environment.

It's generally viewed as having a *very steep learning curve*,
and we are definitely not suggesting it by default for this class.

More information: `https://www.gnu.org/software/emacs/`

Scalable Vector Graphics (SVG)

A standard graphics format, supported by all the major web browsers.

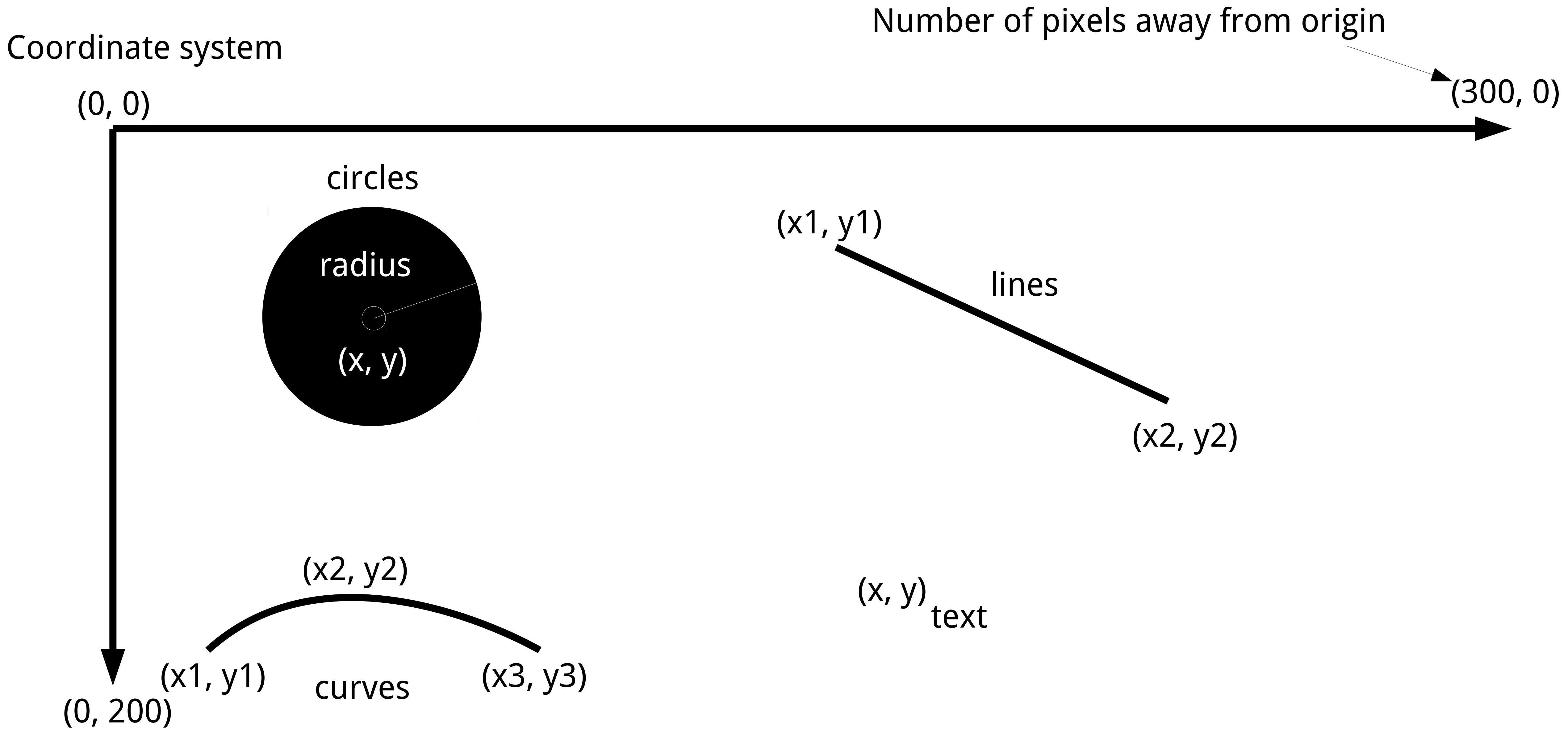
We'll use it for simple graphical output that should work well across operating systems, for the same reason we do the lab visualizations in web browsers.

The details don't matter, as we start from some support code written by the course staff.

“Scalable” and “vector” mean that we can zoom in or out of pictures as far as we want without losing information.

More information: `https://en.wikipedia.org/wiki/Scalable_Vector_Graphics`

Primitive SVG Ingredients (for our purposes)



Reminder on Course Collaboration Policy

Novice programmers often don't appreciate what counts as “copying” when it comes to programs.

Part of our job in 6.009 is to teach you that!
We want you to practice *all* the key skills of software development, *not* outsource some to other students, StackOverflow, etc.

Please read our collaboration policy:

https://6009.csail.mit.edu/fall17/collaboration