WRITING SIMPLE PROGRAMS

CSSE 120—Rose Hulman Institute of Technology

Show Off Some Animations

- Who would like me to show off their work?
- Otherwise I'll pick some programs at random

Setting Up IDLE Shortcut

- Verify IDLE shortcut:
  - Launch IDLE
    (Start → All Programs → Python 2.5 → IDLE)
  - In IDLE, choose File → Open…
  - What directory does the "Open Dialog" start in?
    - CSSE 120? Excellent, help a neighbor in need
    - Python25? Follow the link in the Schedule details
      - http://www.rose-hulman.edu/class/csse/resources/Python/installation.htm
      - Follow step 4

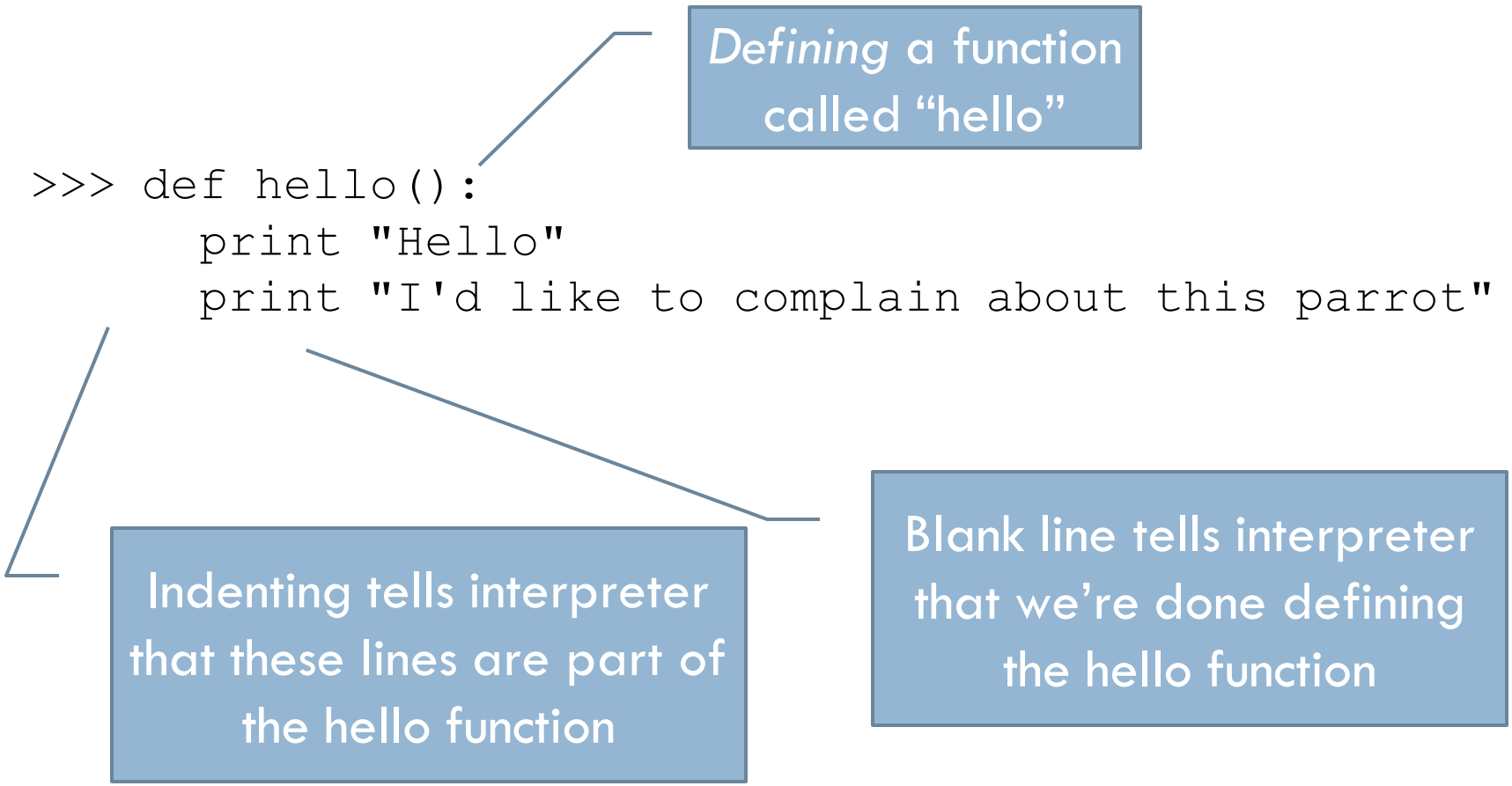
The Python Interpreter

- What it does:
  - Takes in Python commands
  - Converts them to 0s and 1s for the "CPU"
  - Gets answer back from "CPU"
- How we'll use it:
  - IDLE's Python *shell*—lets us "talk with" the interpreter
  - >>> is a Python *prompt*

Sequences of Statements

- *Functions*
  - Named sequences of statements
  - Can *invoke* them—make them run
  - Can take *parameters*—changeable parts

Parts of a Function Definition



Defining vs. Invoking

- Defining a function says what the function should do
- Invoking a function makes that happen
  - Parentheses tell interpreter to invoke the function

```
>>> hello()
Hello
I'd like to complain about this parrot
```

  - Later we'll define functions with parameters

Saving Programs

- Annoying to keep retyping
- Can save definitions in separate files
  - Call *modules* or *scripts*
  - In IDLE, use File → New Window
- Can edit with plain old text editor (like Notepad)
- Can use a *programming environment*
  - Recognizes what you type
  - Tries to help
  - Examples: IDLE, Eclipse

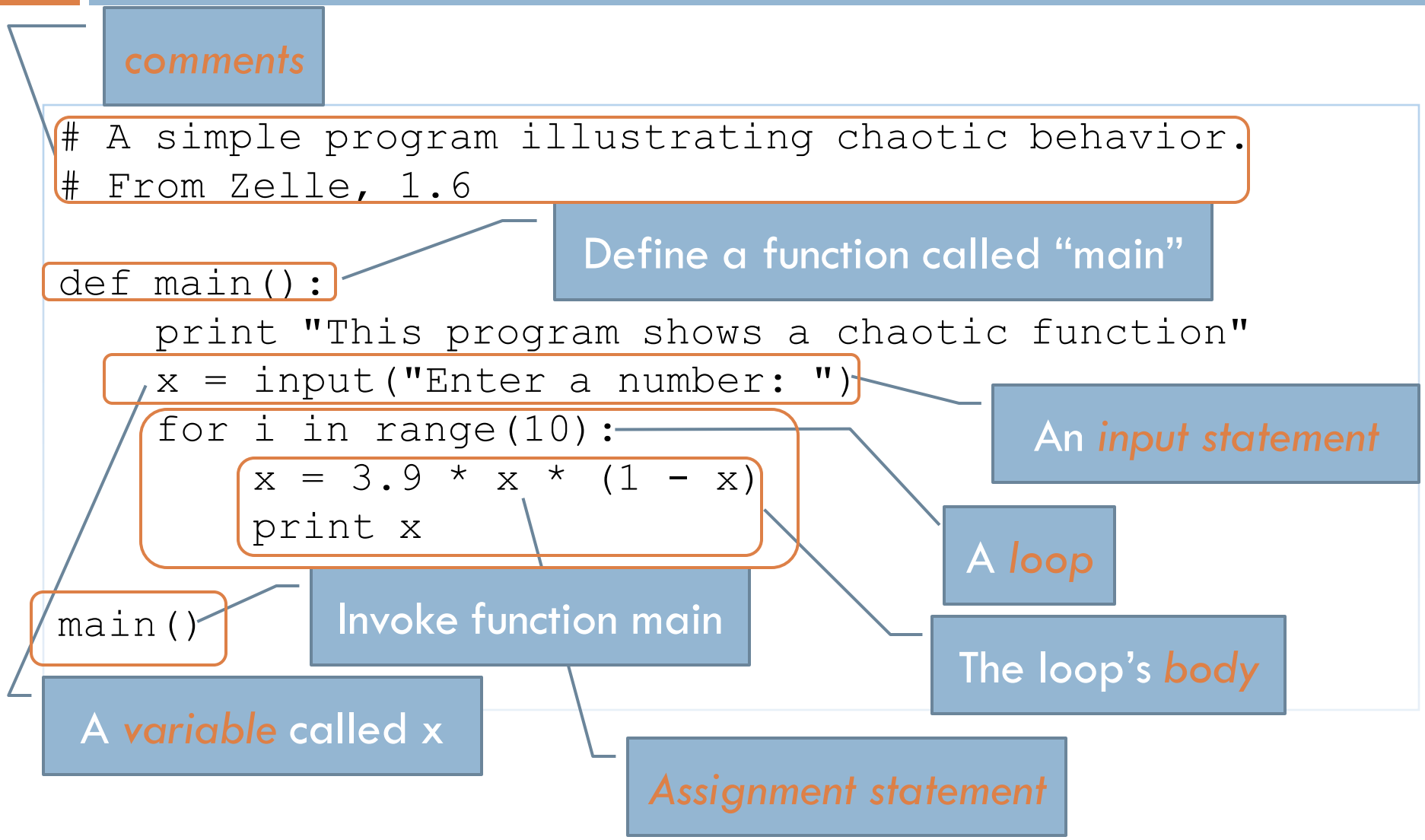
Running Programs

- Like typing in all the lines, but easier
- One way: Open file in IDLE and run it
  - File → Open…
  - Select the file
  - Run → Run Module
- Another way: type `import <module>` at prompt
  - Replace <module> with name of module
  - Don't type the ".py"
  - Example: `import chaos`

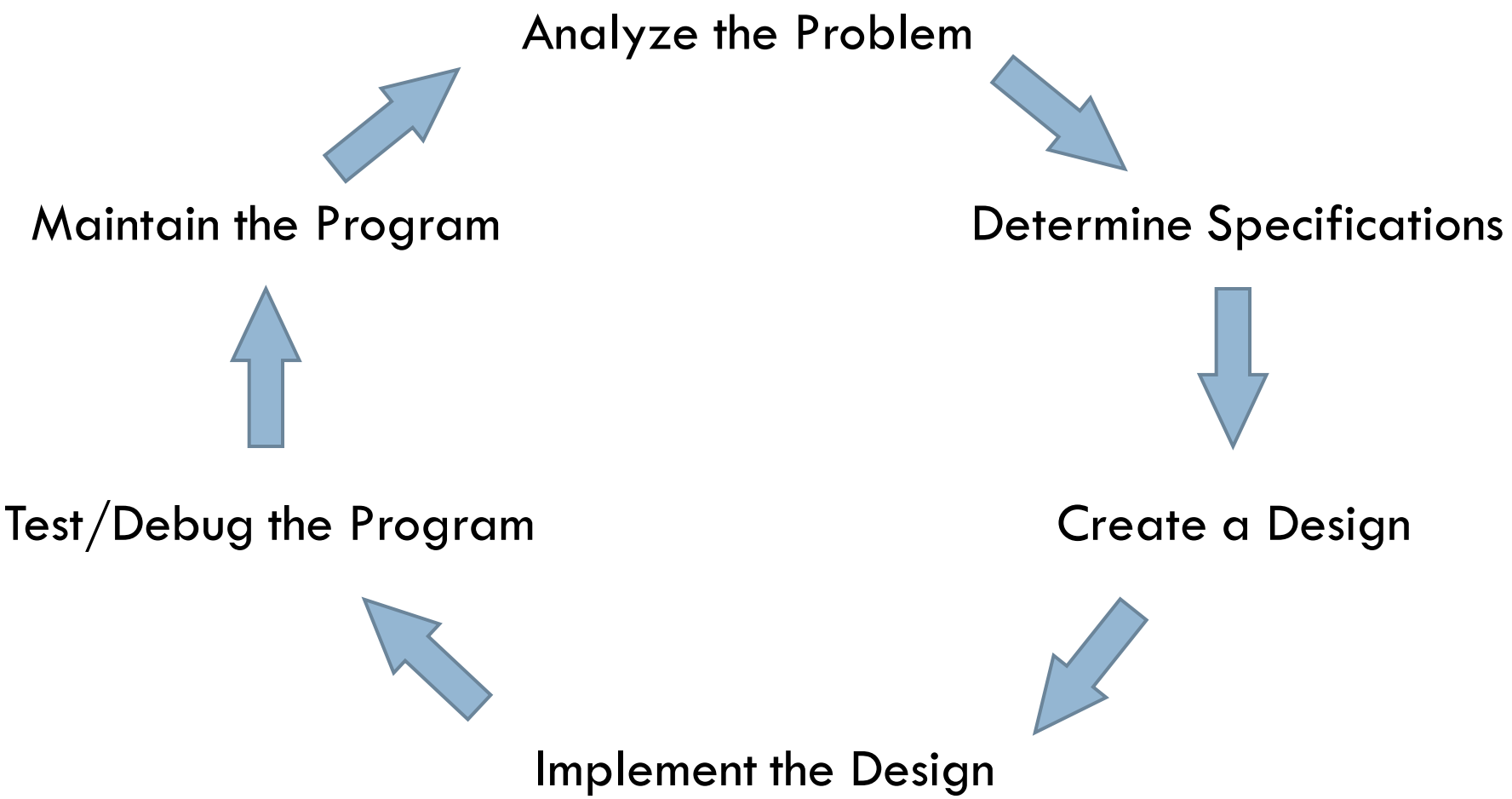
Your .pyc is in my directory!

- A partially translated version of your file
- Called *byte code*
- Interpreter saves this to make loading faster
- Try double-clicking it!

Breaking Down Chaos

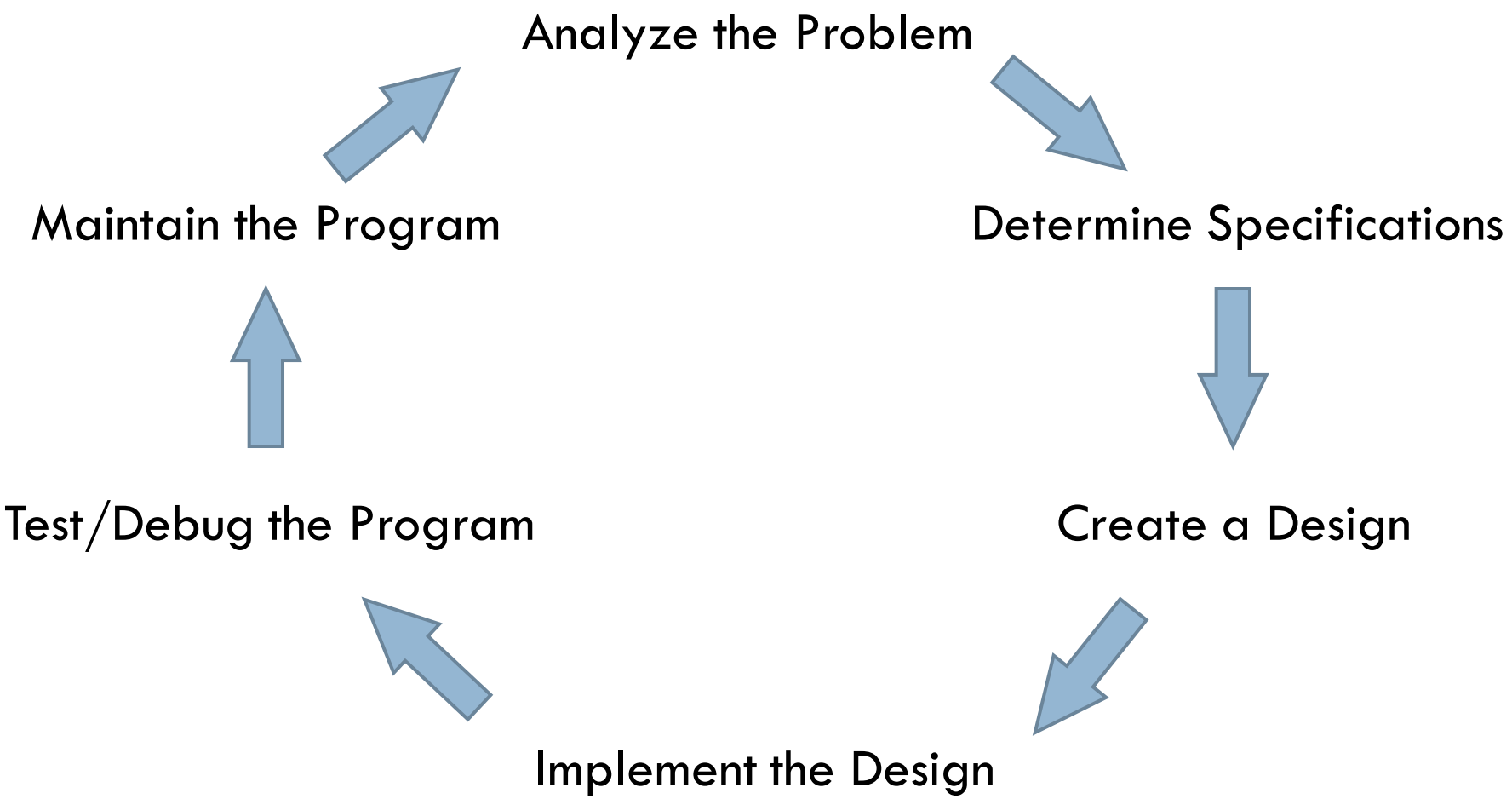


The Software Development Process



Susan's European Adventure

The Software Development Process



Identifiers: Names in Programs

- Uses of *identifiers* so far…
  - Modules
  - Functions
  - Variables
- Rules for identifiers in Python
  - Start with a letter or _ (the "underscore character")
  - Followed by any sequence of letters, numbers, or _
- Case matters! spam ≠ Spam ≠ sPam ≠ SPAM
- Choose descriptive names!

Reserved Words

- Built-in names
- Can't use as regular identifiers
- Python reserved words:

and	de	for	is	raise
assert	elif	from	lambda	return
break	else	global	not	try
class	except	if	or	while
continue	exec	import	pass	with
def	finally	in	print	yield

Expressions

- Fragments of code that produce or calculate new data values
- Examples
  - *Literals*: indicate a specific value
  - *Identifiers*: evaluate to their assigned value
  - *Compound* expressions using *operators*: +, -, *, /, **
- Can use parentheses to group

Programming Languages

- Have precise rules for:
  - Syntax (form)
  - Semantics (meaning)
- Computer scientists use *meta-languages* to describe these rules
- Example…

Output Statements

- Syntax:
  - print
  - print <expr>
  - print <expr>, <expr>, …, <expr>
  - print <expr>, <expr>, …, <expr>,
- Semantics?
- Is this allowed?
  - print “The answer is:”, 7 * 3 * 2

A “slot” to be filled with any expression

Repeat indefinitely

Note: trailing comma

Homework

- Hand in Quiz
- On Angel
  - Lesson → Homework → Homework 2 → Homework 2 Instructions
  - (or just follow link from Schedule page, that's what I do)