

# WRITING SIMPLE PROGRAMS

CSSE 120—Rose-Hulman Institute of Technology

# Announcements

- Homework assigned on Monday and Tuesday each week:
  - ▣ Reading/Quiz part of first assignment due at start of next class
  - ▣ Programming part due 24 hours later

Assigned on	Reading and Quiz Due	Programs due
Monday	Tuesday	Wednesday
Tuesday	Thursday	Friday
Thursday	Monday	Monday

# Show Off Some Animations



- Who would like me to show off their work?
- Otherwise I'll pick some programs at random
  
- What other kinds of programs would you like to write?

# Setting Up IDLE Shortcut

- Verify IDLE shortcut:
  - ▣ Launch IDLE  
(Start → All Programs → Python 2.6 → IDLE)
  - ▣ In IDLE, choose File → Open...
  - ▣ What directory does the “Open Dialog” start in?
    - CSSE 120? Excellent, help a neighbor in need
    - Python26? 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 the Python *prompt*

# Reviewing some concepts you read



## □ *Functions*

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

# Parts of a Function Definition

```
>>> def hello():  
    print "Hello"  
    print "I'd like to complain about this parrot"
```

*Defining a function  
called "hello"*

Indenting tells interpreter  
that these lines are part of  
the hello function

Blank line tells interpreter  
that we're done defining  
the hello function

# 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



# A simple program that defines and invokes a function called main()

*comments*

```
# A simple program illustrating chaotic behavior.  
# From Zelle, 1.6
```

Define a function called "main"

```
def main():
```

```
    print "This program shows a chaotic function"
```

```
    x = input("Enter a number: ")
```

An *input assignment*

```
    for i in range(10):
```

```
        x = 3.9 * x * (1 - x)
```

```
        print x
```

A *loop*

```
main()
```

Invoke function main

The loop's *body*

A *variable* called x

*Assignment statement*

# 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 any old text editor (like Notepad)
- Can use an *integrated development environment(IDE)*
  - ▣ Recognizes what you type
  - ▣ Tries to help
  - ▣ Examples: IDLE, Eclipse

# Running Programs

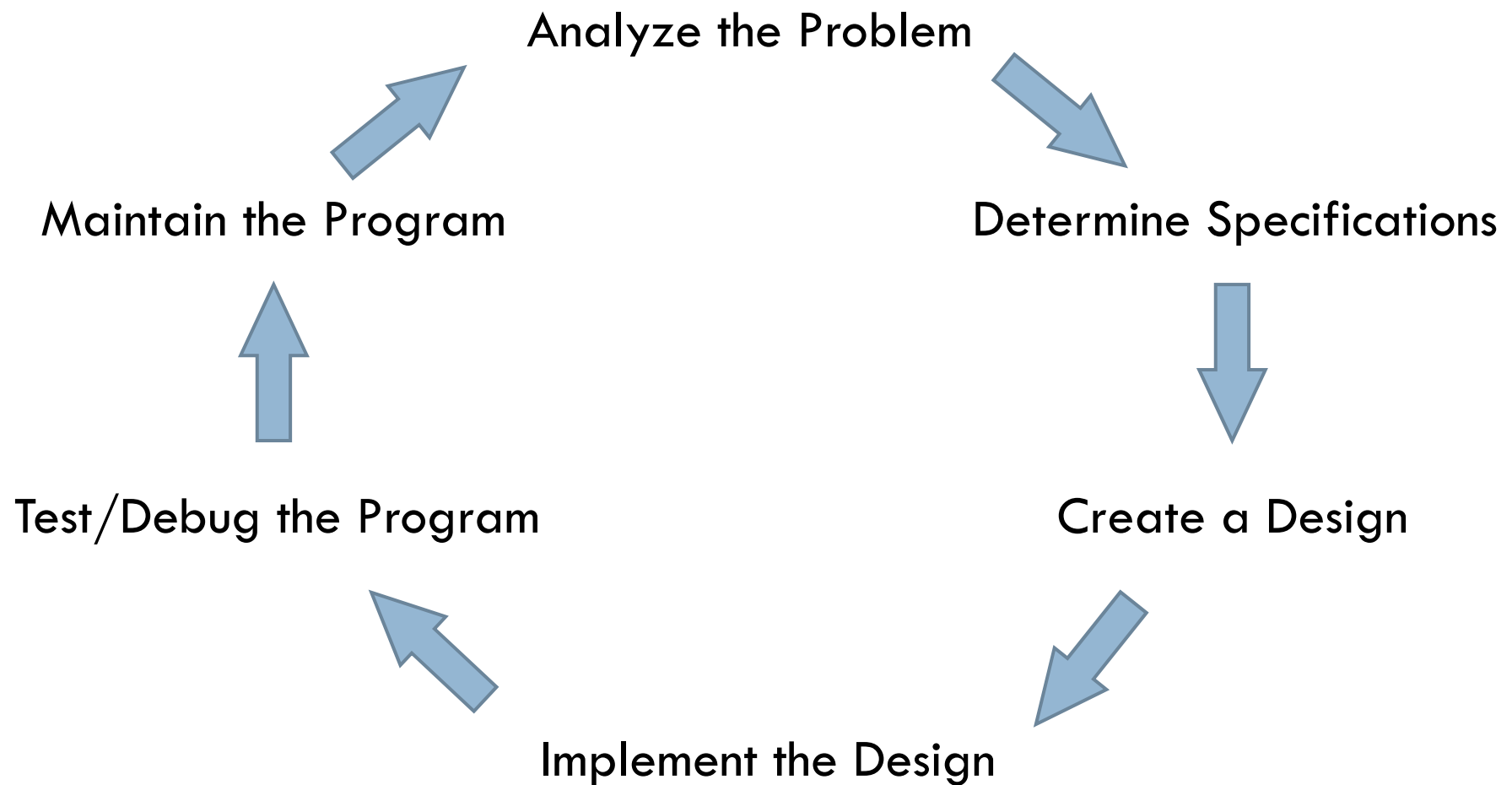
- Like typing in all the lines, but easier
- One way: Open file in IDLE and run it
  - ▣ File → Open...
  - ▣ Select the file
  - ▣ Run → Run Module
- Another way: type `import <module>` at prompt
  - ▣ Replace `<module>` with name of module
  - ▣ Don't type the `".py"`
  - ▣ Example: `import chaos`

# Your .pyc is in my directory!



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

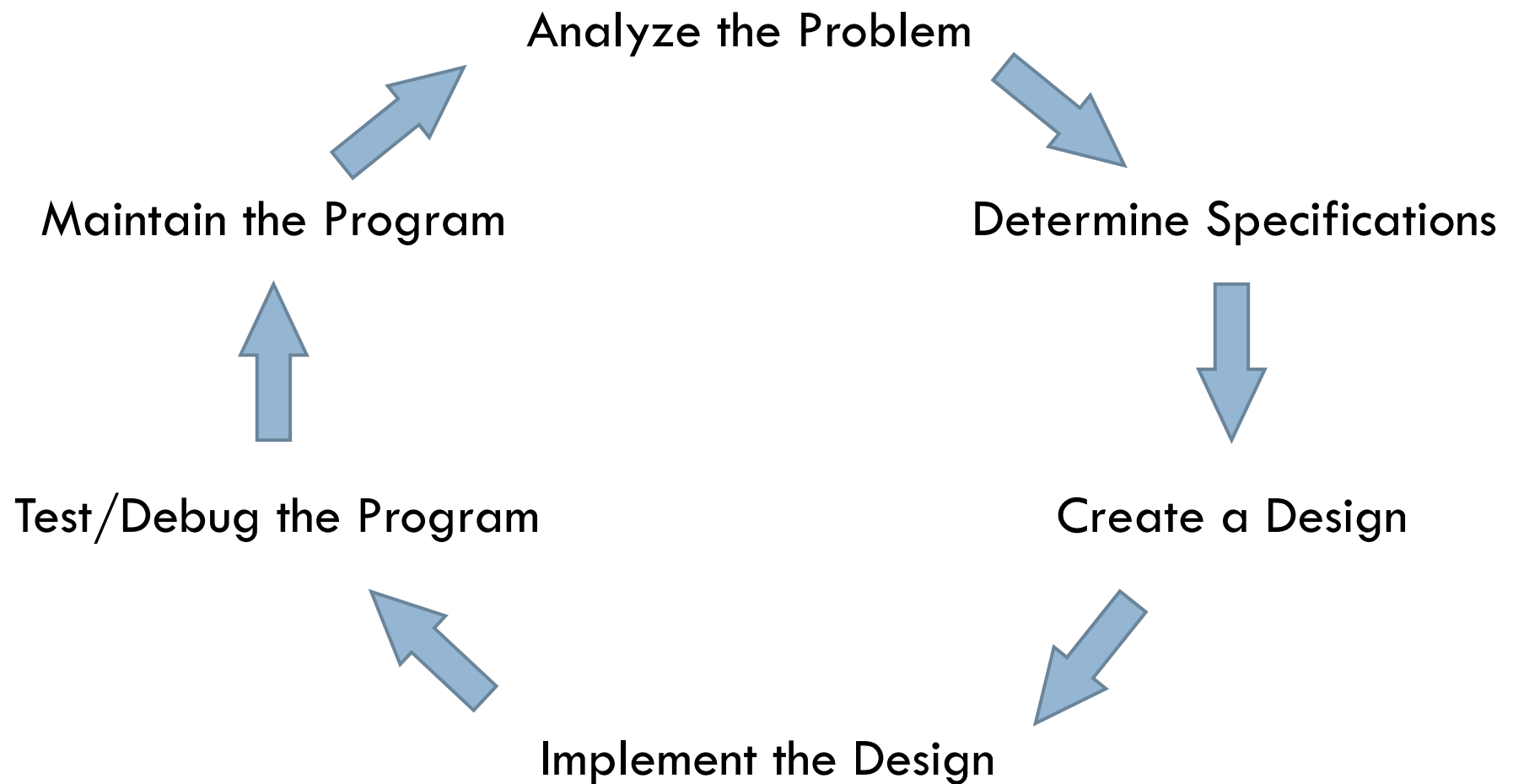
# The Software Development Process



# Road Trip!



# The Software Development Process



# Hidden slide

- This slide is just here because the notes for the previous slide were too long to print on the notes page.



# 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	del	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

# Be careful not to redefine function names accidentally

## □ Examples:

- len – used to find the number of items in a sequence
- max
- min
- float – used to convert a number to a floating point number

# 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>,

A “slot” to be filled with any expression

Repeat indefinitely

## □ Semantics?

## □ Is this allowed?

- print "The answer is:", 7 \* 3 \* 2

Note: trailing comma

# Homework



- Hand in Quiz
- On Angel
  - ▣ Lessons → Homework → Homework 2 → Homework 2 Instructions
  - ▣ (or just follow link from Schedule page, that's what I do)
  - ▣ Reading and Angel quiz due Tuesday.  
Programming part due Wednesday.