

# 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

# 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

```
>>> 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

# 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

*comments*

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

```
def main():
```

Define a function called "main"

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

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

An *input statement*

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

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

A *loop*

```
        print x
```

```
main()
```

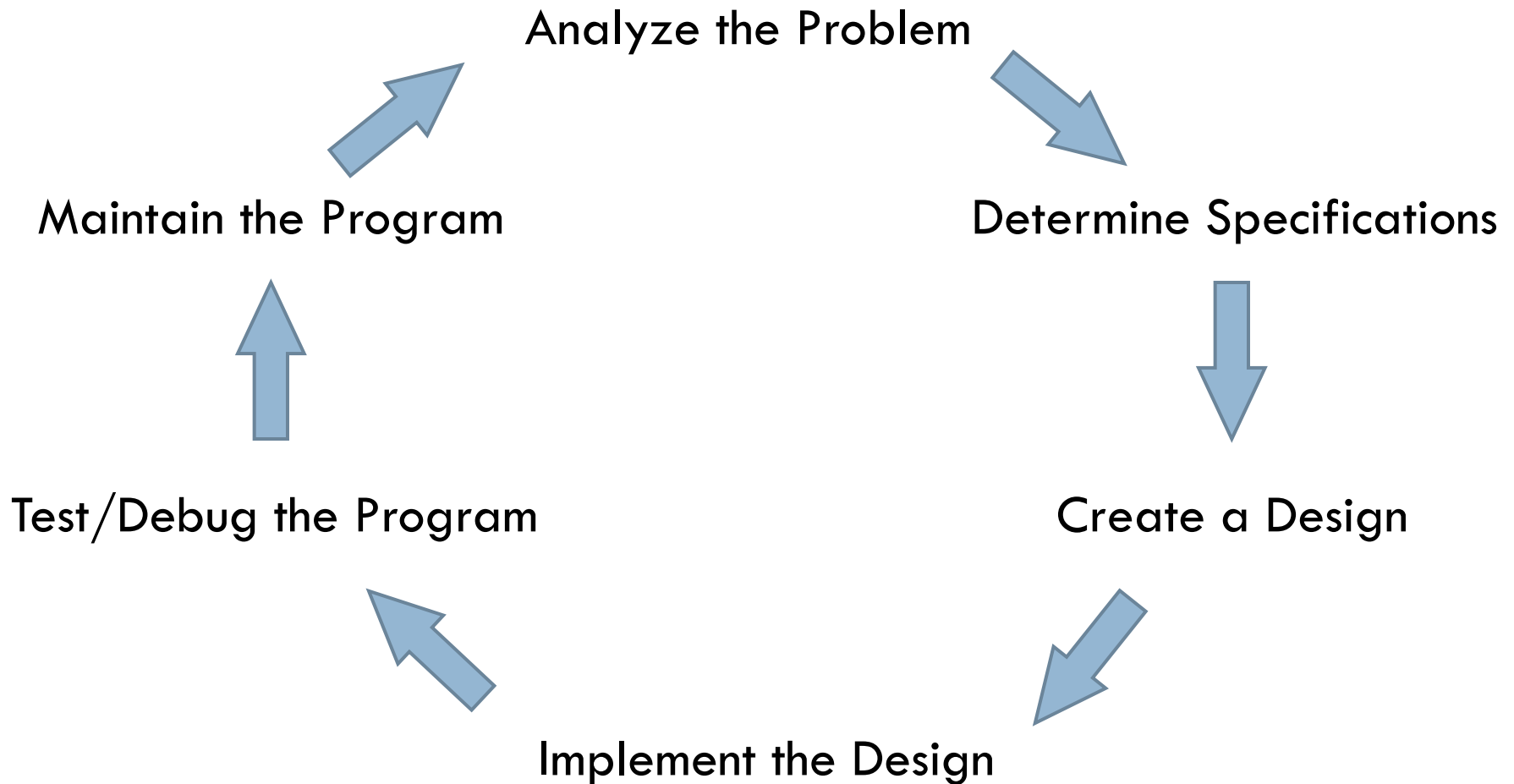
Invoke function main

The loop's *body*

A *variable* called x

*Assignment statement*

# The Software Development Process

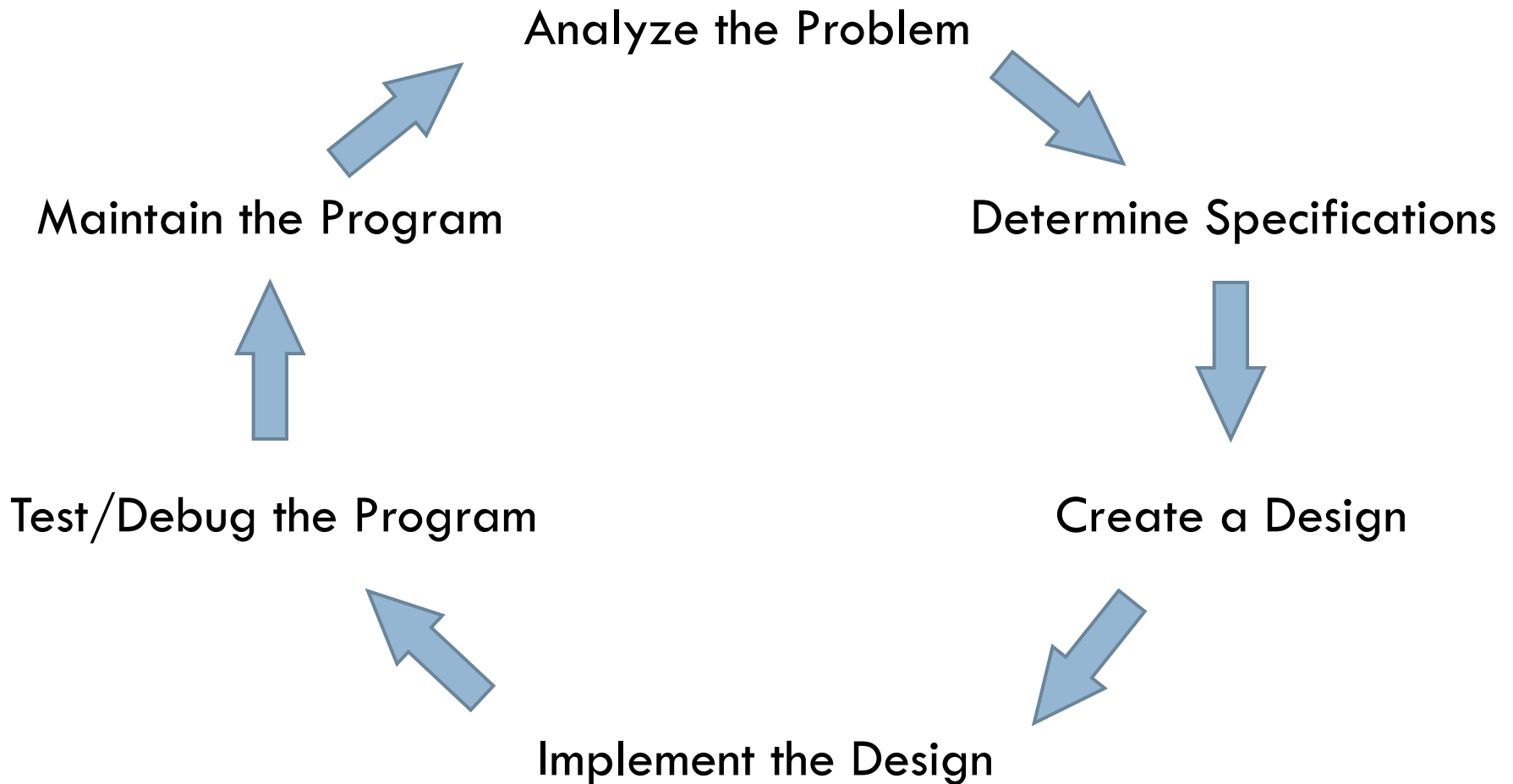


# Susan's European Adventure



# 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`  $\neq$  `Spam`  $\neq$  `sPam`  $\neq$  `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

# 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?

Note: trailing comma

- Is this allowed?

- print “The answer is:”, 7 \* 3 \* 2