

Basic Python Programs, Software Engineering Tools

Rose-Hulman Institute of Technology
Computer Science and Software Engineering

Announcements

- Homework timing:

Day assigned	Reading quizzes due (next class day – 8AM)	Other parts of assignments due 8 AM
Monday	Tuesday	Wednesday
Tuesday	Thursday	Thursday
Thursday	Monday	Monday

Q1

Who wants to share?

Sample Scenes from HW 1

Outline

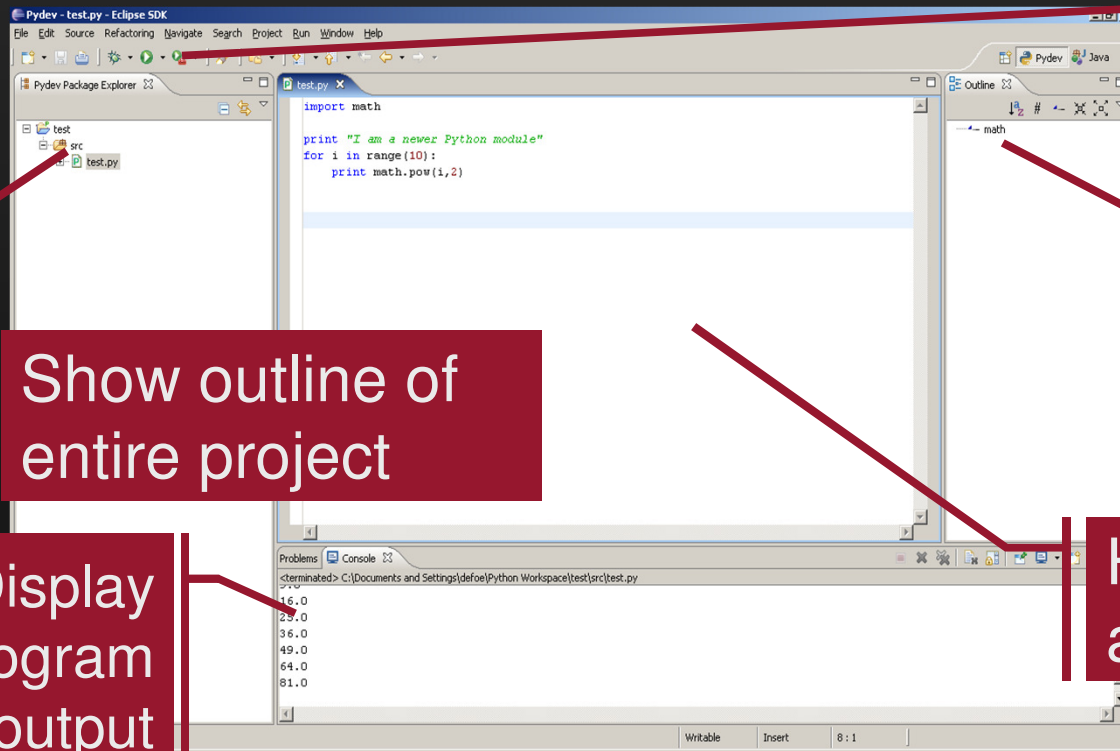
- Goals for today:
 - Get started with Eclipse and Subversion
 - Learn and practice writing small programs
 - Functions
 - The main function
 - The input-compute-output pattern
 - Examples: chaos, temperature, kph

Integrated Development Environments

- What do they do?
- Why use one?
- Our IDE – Eclipse

Details on next slides

IDEs – What do they do?



Let us compile, run, and debug

Show an outline of our module (file)

Show outline of entire project

Display program output

Help us enter and edit code

IDEs – Why use one?

- Harness the power of the computer to make us more productive!

IDEs – Why Eclipse?

- Powerful
- Easy to use
- Free and open-source
- An IDE for any language, not just Python

Basic concepts in Eclipse

- *Workspace* – where your projects are stored on your computer
- *Project* – a collection of files, organized in folders, that includes:
 - Source code
 - Compiled code
 - Design documents
 - Documentation
 - Tests
 - And more...

Fixing Eclipse First

- Link from home page

<http://www.rose-hulman.edu/class/csse/resources/InstallationInstructions/2011-2012/EclipseWorkaround-2011-2012.html>

Setting up Eclipse

- Just need to do this the **first time!**
- Follow along with me
 - We will:
 - Open Eclipse
 - Set the workspace
 - Switch to the **PyDev120** perspective
 - If your setup is different, see
 - <http://www.rose-hulman.edu/class/csse/resources/Eclipse/installation.htm>

Follow Along...

1. *File* ➤ *New* ➤ *Pydev Project*
2. *File* ➤ *New* ➤ *Pydev Module*
3. Type `print("hello world")`
4. Run the program
5. Type a few more *print* statements, including one that is wrong.
 - See where the error message appears. Clicking on it brings you to the offending line.

Help us enter
and edit code

Views, Editors, PyDev Perspectives

Tabbed views of source code

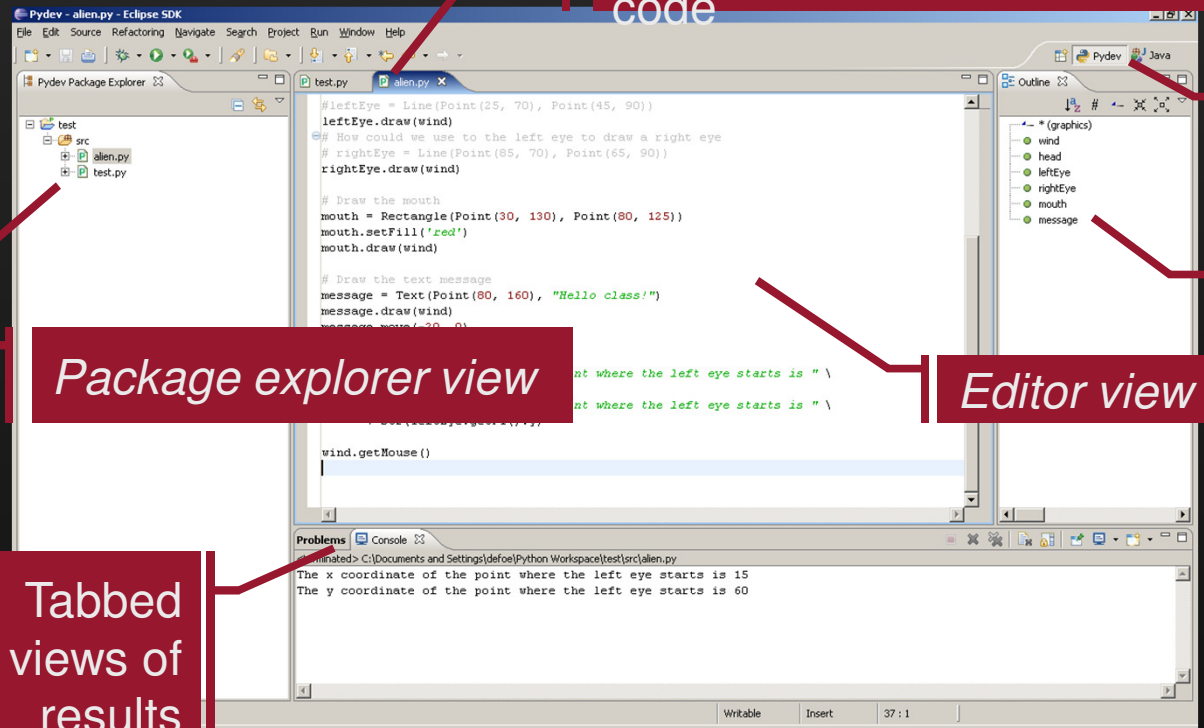
Perspective switcher

Outline view

Editor view

Package explorer view

Tabbed views of results



Eclipse in a Nutshell

- *Workspace* – where your projects are stored on your computer
- *Project* – a collection of files, organized in folders
- *Perspective* – a set of views and editors
- *View* – an area of a window showing focused information (code, outline, results, etc.)

Software Engineering Tools

- IDEs, like **Eclipse** and **IDLE**
- Version Control Systems, like **Subversion**
- Testing frameworks, like **JUnit**
- Diagramming applications, like **UMLet**, **Violet** and **Visio**
- Modeling languages, like **Alloy**, **Z**, and **JML**
- Task management trackers like **TRAC**

Version Control Systems

Store “snapshots” of all the changes to a project over time

Version Control Systems

- Benefits for individual work:
 - Logging and Backups
 - Act as a “global undo” to whatever version you want to go back to
 - Maintain a log of the changes made
 - Can simplify debugging

Version Control Systems

- Benefits for group work:
 - Multiple users can share work on a project
 - Record who made what changes to a project
 - Provide help in resolving conflicts between what the multiple users do
 - Maintain multiple different versions of a project simultaneously

Version Control Systems

- Benefits for CSSE 120 programming work:
 - Three-click turn in for programming projects
 - Get back comments from the grader right in the code
 - No need for ANGEL drop boxes

Our Version Control System

- Subversion, sometimes called SVN
- A free, open-source application
- Lots of tool support available
 - Works on all major computing platforms
 - *TortoiseSVN* for version control in Windows Explorer
 - *Subclipse* for version control inside Eclipse

Q2a

Version Control Terms

Repository: the copy of your data on the server, includes **all** past versions

Subversion Server

Alice's Repository

Bob's Repository ...

Working copy: the **current** version of the files, on your computer

Q2b

Alice's Computer

Working Copy

Bob's Computer

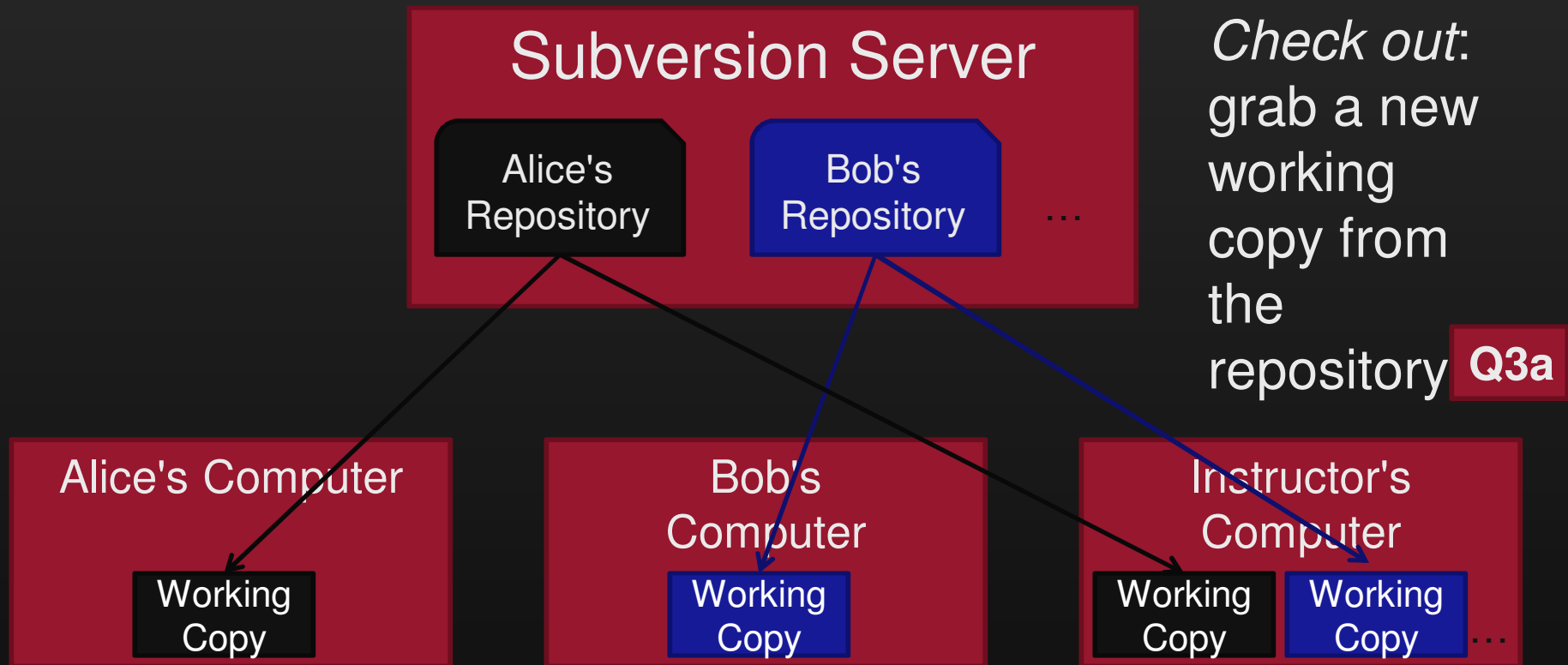
Working Copy

Instructor's Computer

Working Copy

Working Copy ...

Version Control Steps—Check Out



Version Control Steps—Edit

Subversion Server

Alice's
Repository

Bob's
Repository

...

Edit: make
independent
changes to a
working copy

Alice's Computer

Working
Copy

Bob's Computer

Working
Copy

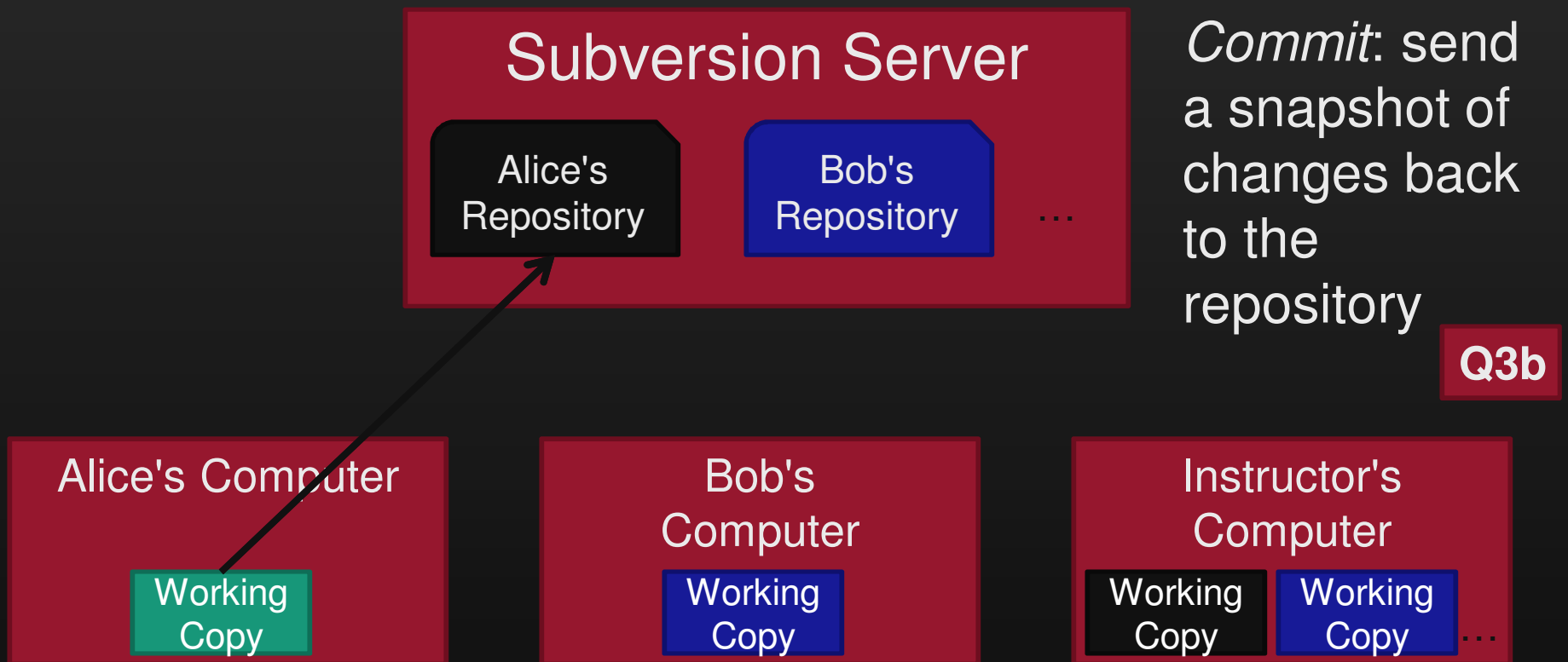
Instructor's Computer

Working
Copy

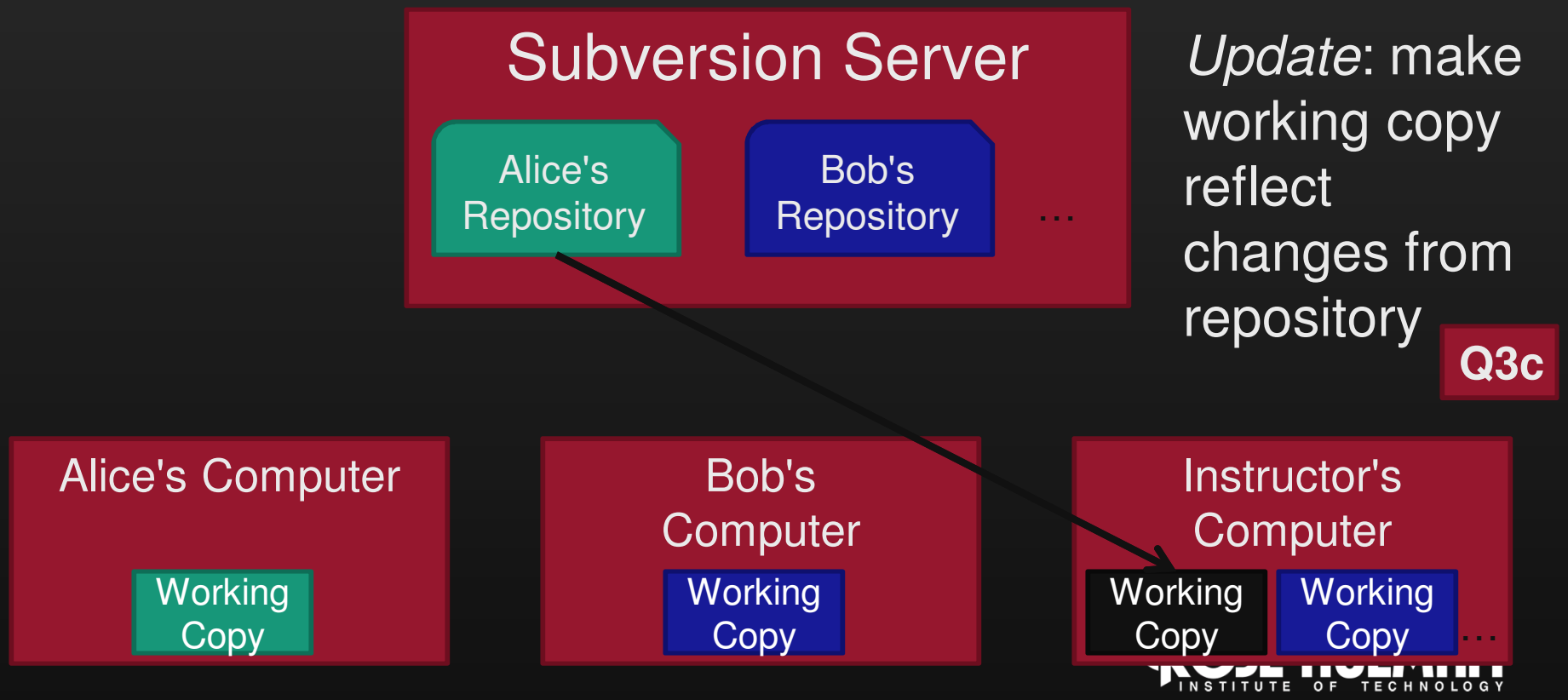
Working
Copy

...

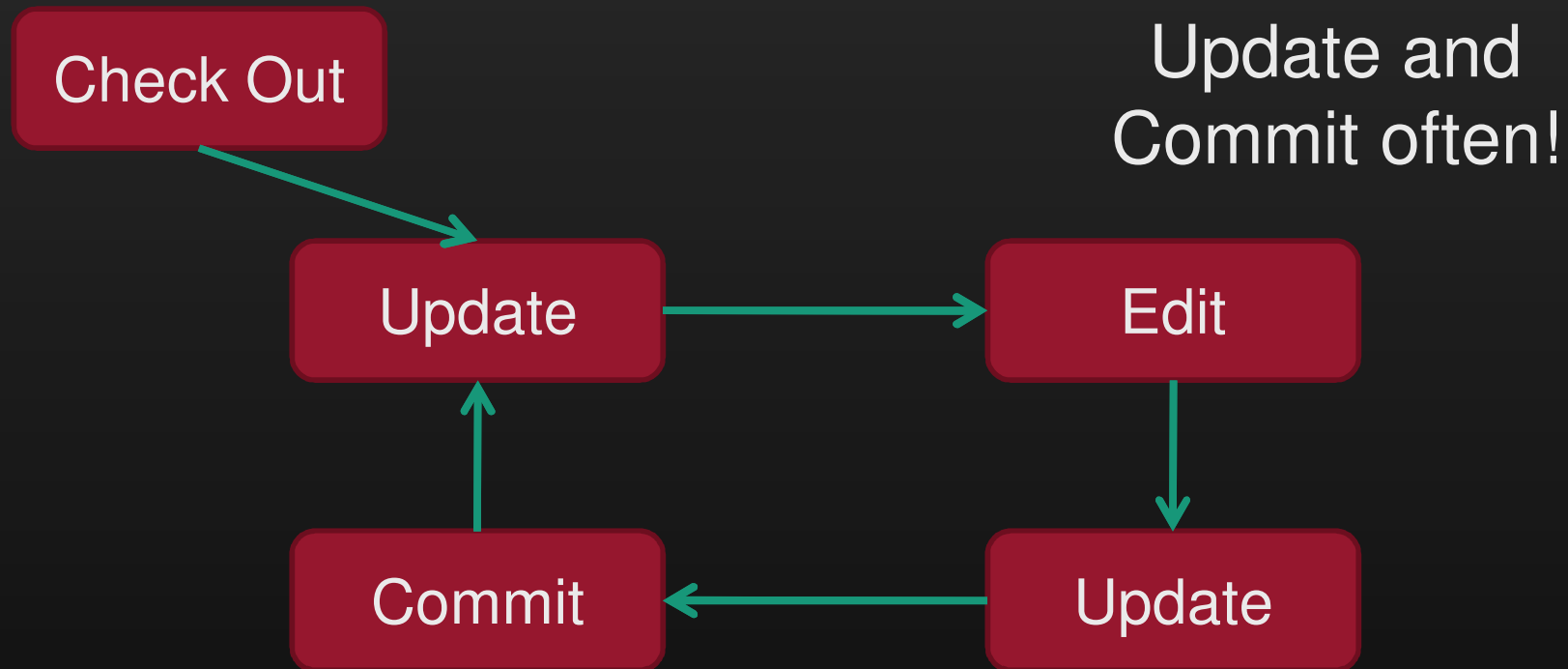
Version Control Steps—Commit



Version Control Steps—Update



The Version Control Cycle



Check out today's project

- Follow along
- You'll need:
 - Subversion URL from Homework 2
 - SVN password

Functions

- Examine the **hello.py** module in your Eclipse project.
- *Functions*
 - Named sequences of statements
 - We can *invoke* them—make them run
 - They can take *parameters*—*changeable* parts

Parts of a Function Definition

Defining a function called "hello"

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

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

Entering a blank line tells interpreter that we're done defining the hello function

Defining vs. Invoking

- *Defining* a function says what it should do
- *Invoking* (calling) a function makes that happen
 - Parentheses tell the interpreter to invoke the function

```
>>> hello()  
Hello  
I'd like to complain about this parrot
```
 - Later we'll define functions with *parameters*

Q4

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!

Q5a

Reserved Words

- Built-in names, can't be used as 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

Q5b

Don't redefine function names!

- Examples that trip me up:
 - *len* – used to find the length of a list
 - *max*
 - *min*

Expressions

- Fragments of code that produce or calculate new data values
- Examples
 - *Literals*: indicate a specific value: **2**, **3.5**, **"hello"**
 - *Identifiers*: evaluate to their assigned value: **x**, **width**
 - *Compound expressions* contain other expressions:
 - E.g., **3 + 4 * 2**, **x < 5**
- Can use parentheses to group: **(3 + 4) * 2**

Q6,7

Recall...

- 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>, ...)`
 - `print(<expr>, ..., end=<str>)`
- Are these allowed?
 - `print("The answer is:", 7 * 3 * 2)`
 - `print(3, 5, 7, end="")`
- Semantics?

A “slot” to be filled with any expression

Repeat indefinitely as many times as desired, zero or more

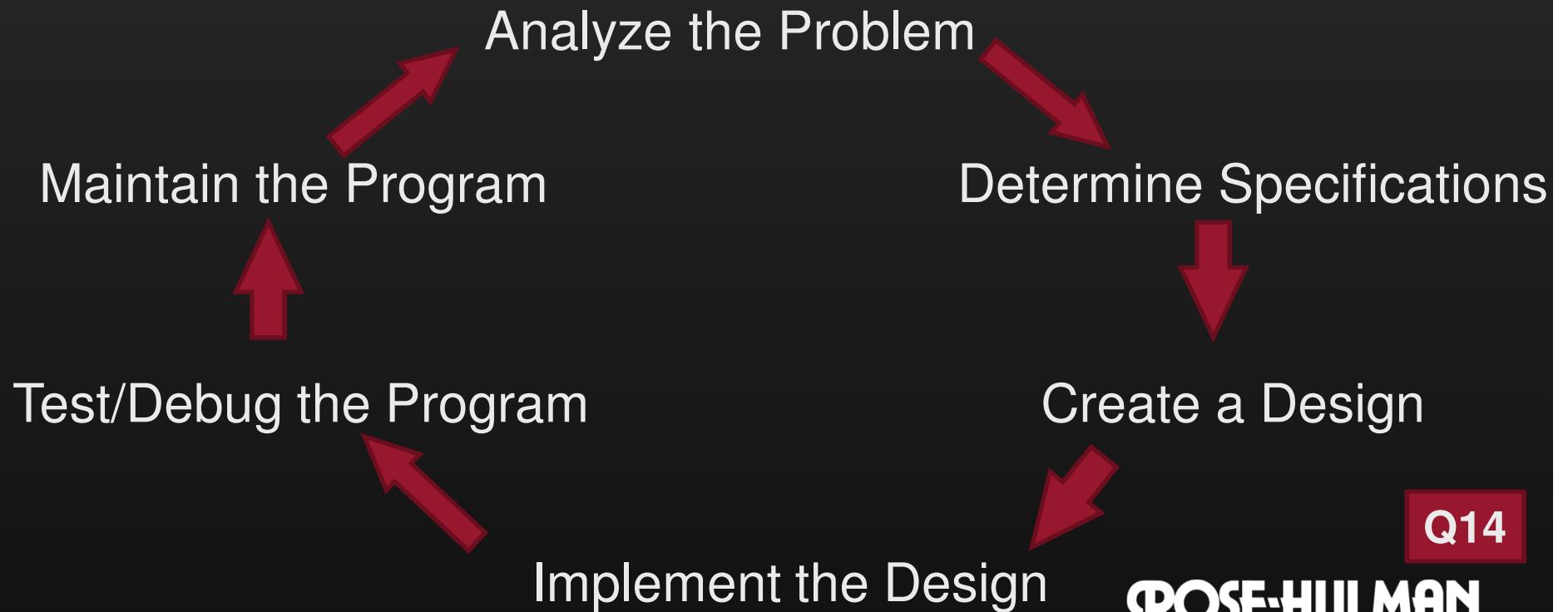
Q8

The *input-compute-output* pattern

- Examine the **chaos.py** module in your Eclipse project.
- Do the TODOs in it.
 - I'll demo some of them with you.
- Commit your code:
 - Right-click project → Team → Commit...

Q9-13

The Software Development Process



Q14

More input-compute-output practice

- Examine the **temperature.py** module in your Eclipse project
- Do the TODOs in it

Homework

- Hand in quiz
- Begin homework
 - Find it from the Schedule page
 - Reading and ANGEL quiz due Tuesday 8 AM
 - Rest (programming part) due Wednesday 8 AM

Q15