As you arrive:

1. Start up your computer and plug it in
2. ***Log into Angel*** and go to CSSE 120
3. Do the ***Attendance Widget*** – the PIN is on the board
4. Go to the course ***Schedule Page***
   - From your *bookmark*, or from the *Lessons* tab in Angel
5. Open the ***Slides*** for today if you wish

Writing simple programs

- The *input-compute-output* pattern
- Eclipse – An Integrated Development Environment (IDE)
- Subversion (SVN) for version control

Functions

- Writing functions
- Calling functions
- Functions with parameters
- The *main* function

Session 2

CSSE 120 – Fundamentals of Software Development

Announcements

- Homework assignments:
  - Reading and the Reading Quiz part of each homework assignment is due at the start of the next class
  - Other parts (typically, programming) always get at least 48 hours, so homework assigned Monday is special.

Day assigned	Reading quizzes due (next class)	Other parts of assignments due (at least 48 hours)
Monday	Tuesday	Wednesday, at the same time of day that your class meetings begin
Tuesday	Thursday	Thursday
Thursday	Monday	Monday

Q1

Show Off Some Cool Graphics

- 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?

Q2

Outline

- ✓Answer questions & review concepts from Session 1
- Tools for software development:
  - Integrated Development Environment (IDE) – Eclipse
  - Version control – Subversion (SVN)
- Writing simple programs
  - Functions
  - The *main* function
  - The *input-compute-output* pattern
  - Examples: *chaos*, *temperature*, *kph*

Integrated Development Environments (IDEs)

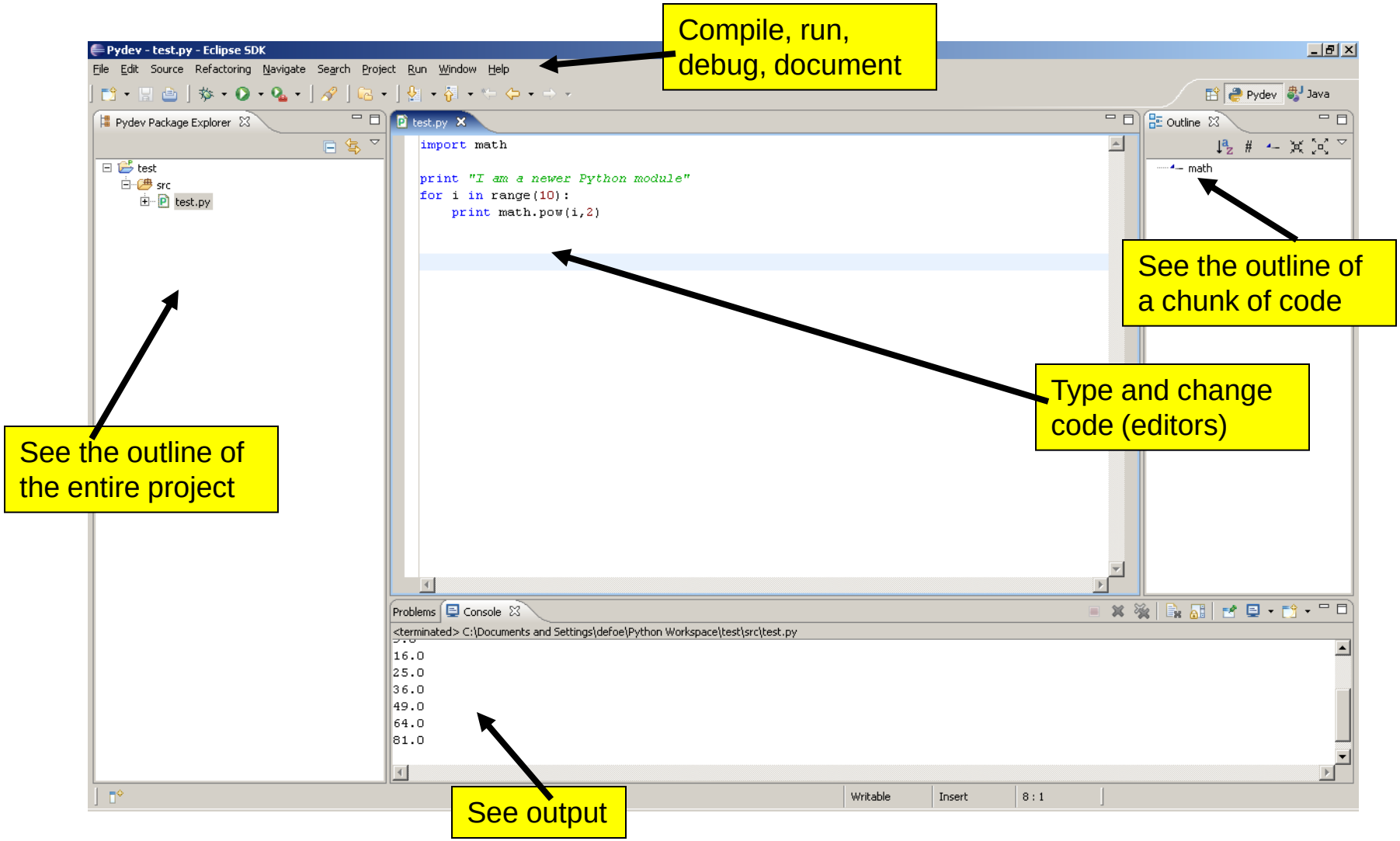
- What are they?
- Why use one?
- Our IDE – Eclipse
  - Why we chose it
  - Basic concepts in Eclipse
    - Workspace, Workbench
    - Files, folders, projects
    - Views, editors, perspectives

The next slides address the listed points

IDEs – What are they?

An IDE is an application that makes it easier to develop software.

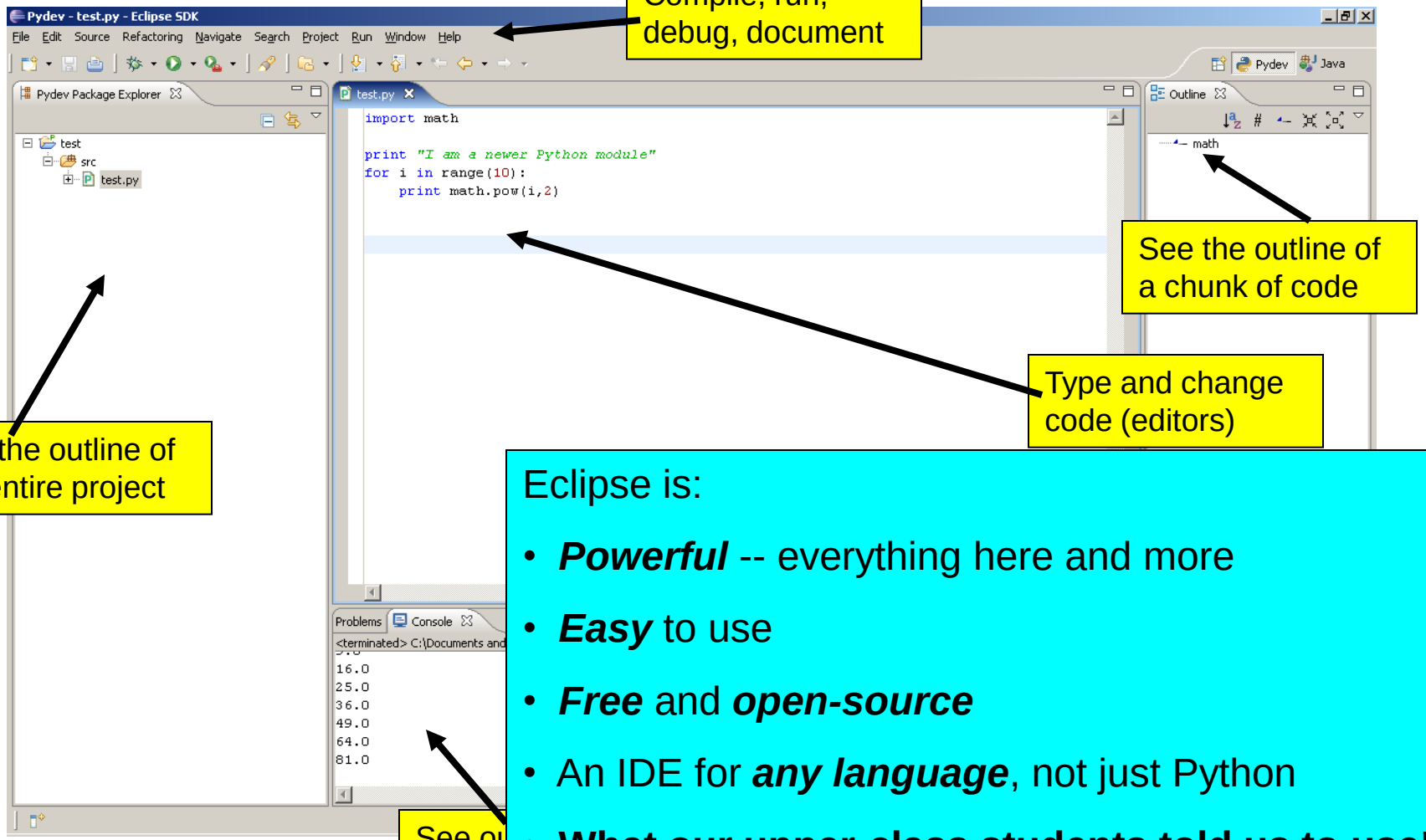
They try to make it easy to:



IDEs – Why use one?

An IDE is an application that makes it easier to develop software.

They try to make it easy to:



Eclipse is:

- ***Powerful*** -- everything here and more
- ***Easy*** to use
- ***Free*** and ***open-source***
- An IDE for ***any language***, not just Python
- **What our upper-class students told us to use!**

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*** (the code that you write)
  - ***Compiled code*** (what your source code is translated into, for the machine to run)
  - ***Design documents***
  - ***Documentation***
  - ***Tests***
    - And more that you will learn about over time
- ***Workbench*** – what we saw on the previous slide, that is, the tool in which you do your software development

Special things to do ONCE – for your first Eclipse session only

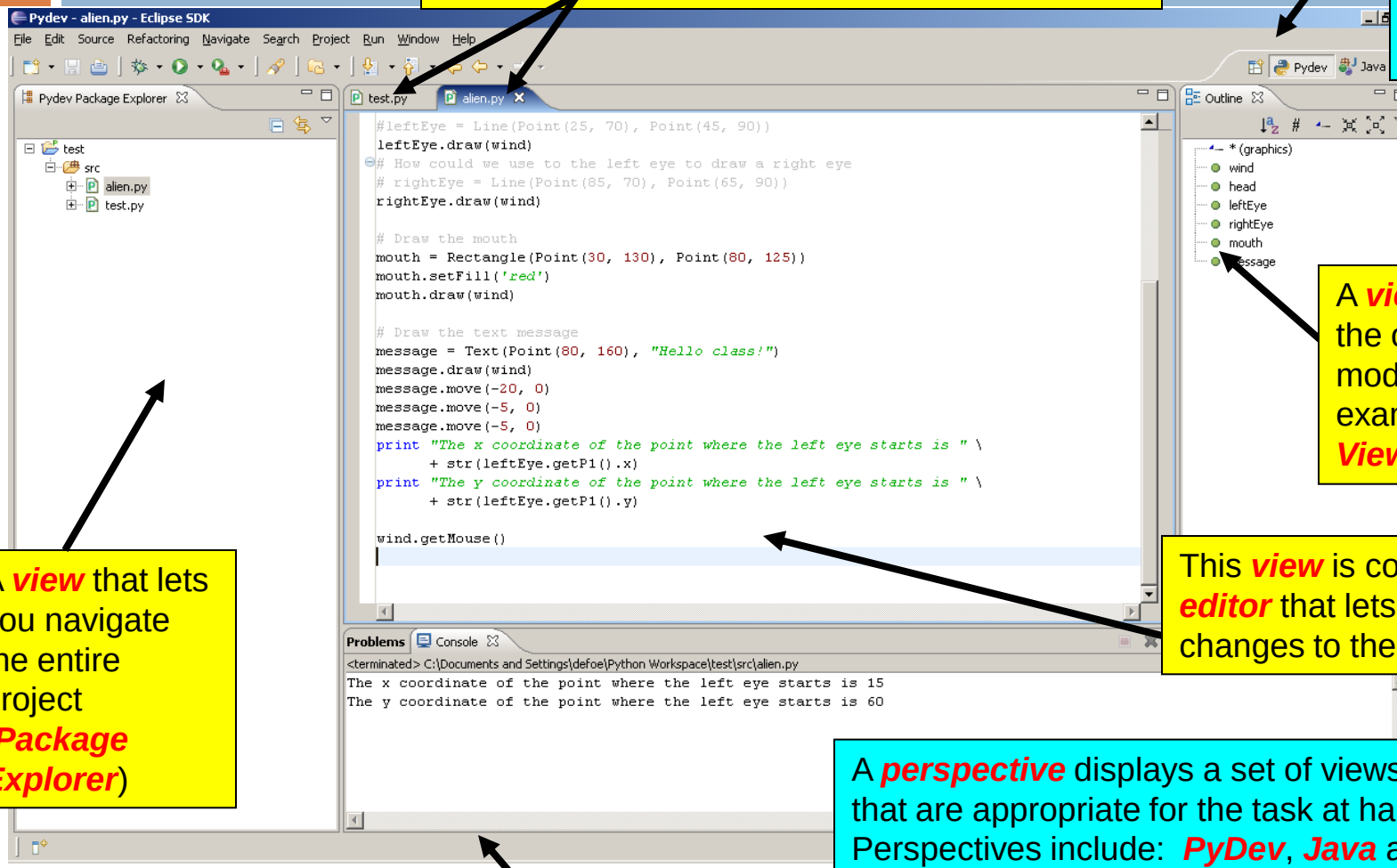
- I will demo the most common case – just follow my demo if your setup is "the usual".
  - If your setup is different, see
    - http://www.rose-hulman.edu/class/csse/resources/Eclipse/installation.htm
  - Here is a summary of what I will lead you through:
    - Open Eclipse
    - Set your workspace [careful, pick the one we set up for you]
    - Switch to the Pydev perspective

Your first Eclipse program

- Your instructor will lead you through your creation of your first Python project in Eclipse. Here is a brief summary:

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 and how clicking on it brings you to the offending line.

Views, editors, perspectives



This is the ***PyDev perspective*** but just a button click brings us to another

Tabbed ***views*** of the source code of this project

A ***view*** that shows the outline of the module being examined (***Outline View***)

A ***view*** that lets you navigate the entire project (***Package Explorer***)

This ***view*** is controlled by an ***editor*** that lets you make changes to the file

A ***perspective*** displays a set of views and editors that are appropriate for the task at hand. Perspectives include: ***PyDev***, ***Java*** and lots more

Tabbed ***views*** (***Problems***, ***Console***)

Eclipse in a Nutshell

- ***Workspace*** – where your ***projects*** are stored on your computer
- ***Project*** – a collection of files, organized in folders, that includes:
  - ***Source code*** and ***Compiled code*** and more
- ***Workbench*** – the tool in which to work
  - It has ***perspectives*** which organize the ***views*** and ***editors*** that you use
- ***View*** – a "window within the window"
  - displays code, output, project contents, debugging info, etc.

Software Engineering Tools

- The computer is a powerful tool
- We can use it to make software development easier and less error prone!
- Some 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

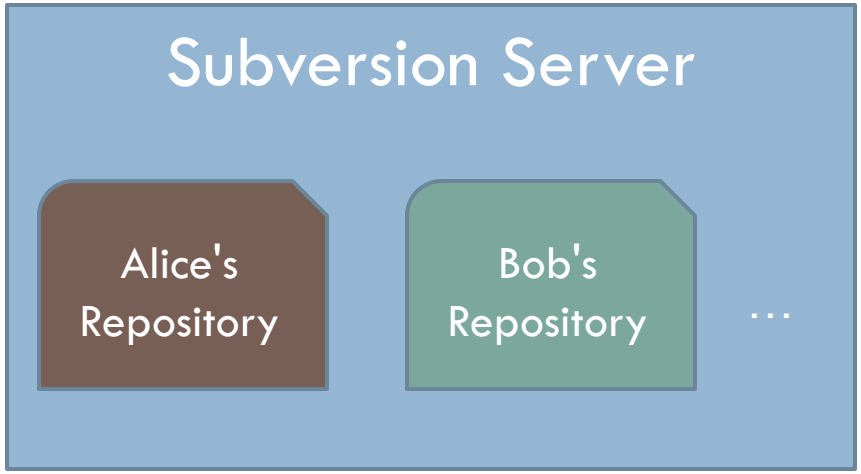
- Store "snapshots" of all the changes to a project over time
- Benefits:
  - Multiple users
    - Multiple users can share work on a project
    - Record who made what changes to a project
    - Provide help in resolving conflicts between what the multiple users do
    - Maintain multiple different versions of a project simultaneously
  - 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
  - Drop boxes are history!
    - Turn in programming projects
    - Get it back with comments from the grader embedded in the code

Our Version Control System

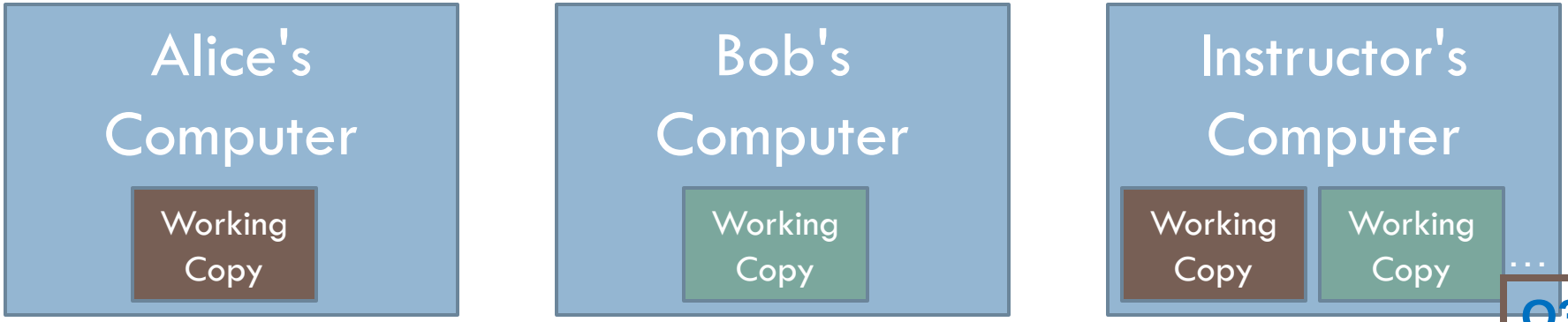
- Subversion, sometimes called SVN
- A free, open-source application
- Lots of tool support available
  - Works on all major computing platforms
  - **TortoiseSVN** for version control in Windows Explorer
  - **Subclipse** for version control inside Eclipse

Version Control Terms

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

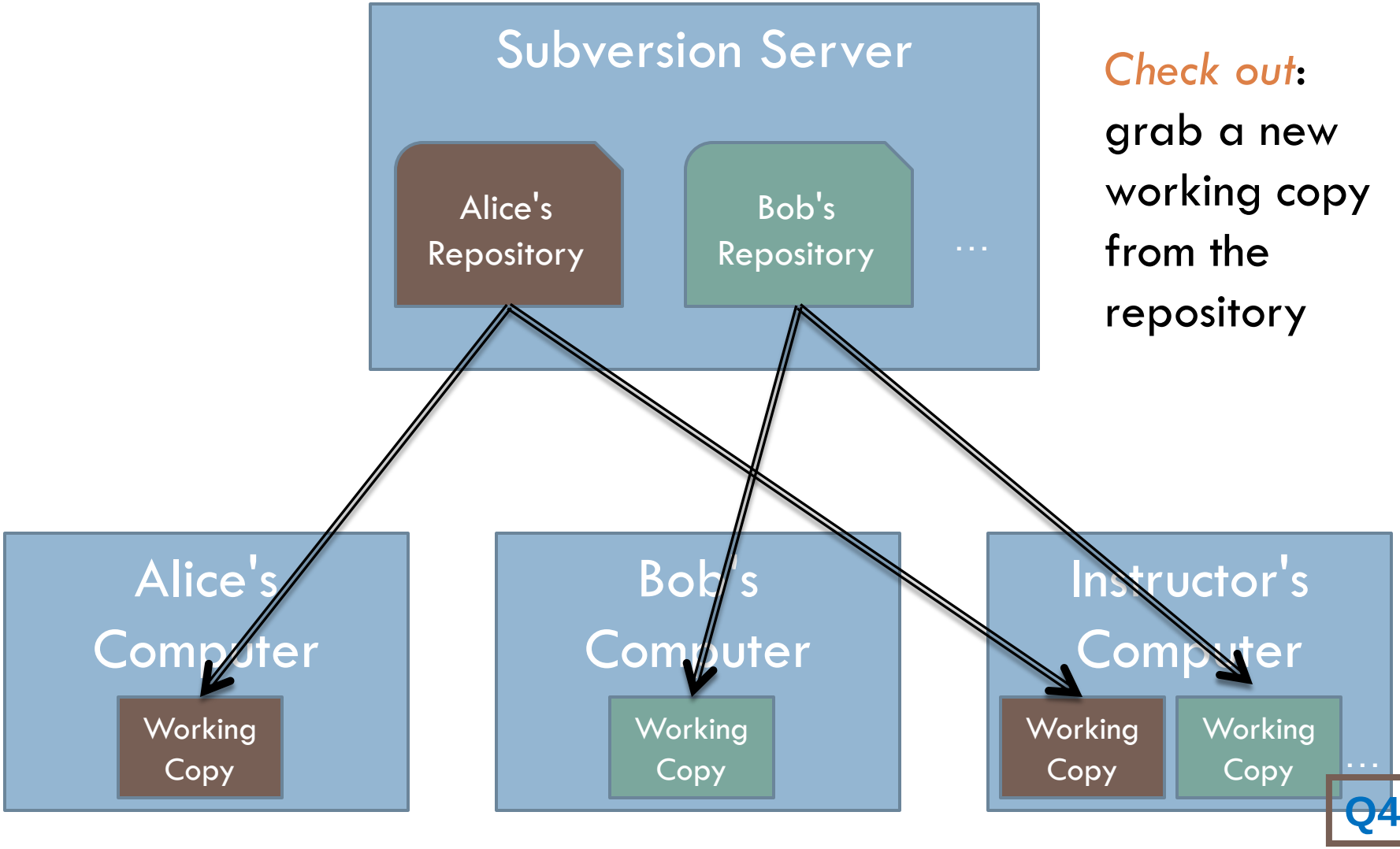


Working copy: the ***current*** version of your data on your computer



Q3

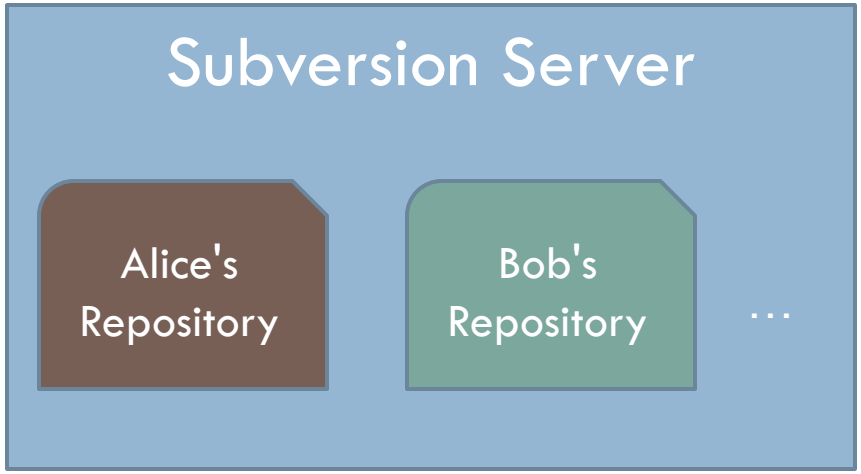
Version Control Steps—Check Out



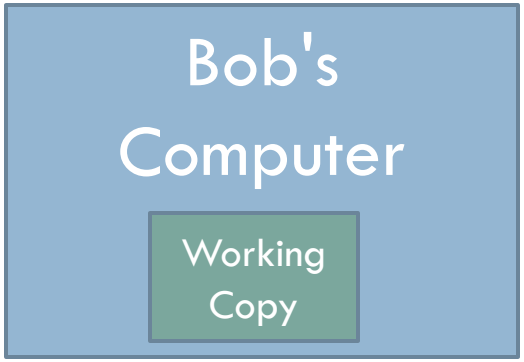
Check out: grab a new working copy from the repository

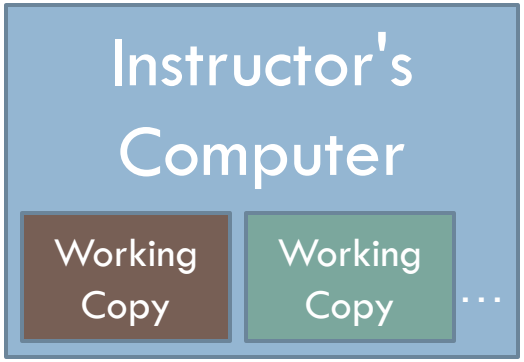
Q4a

Version Control Steps—Edit

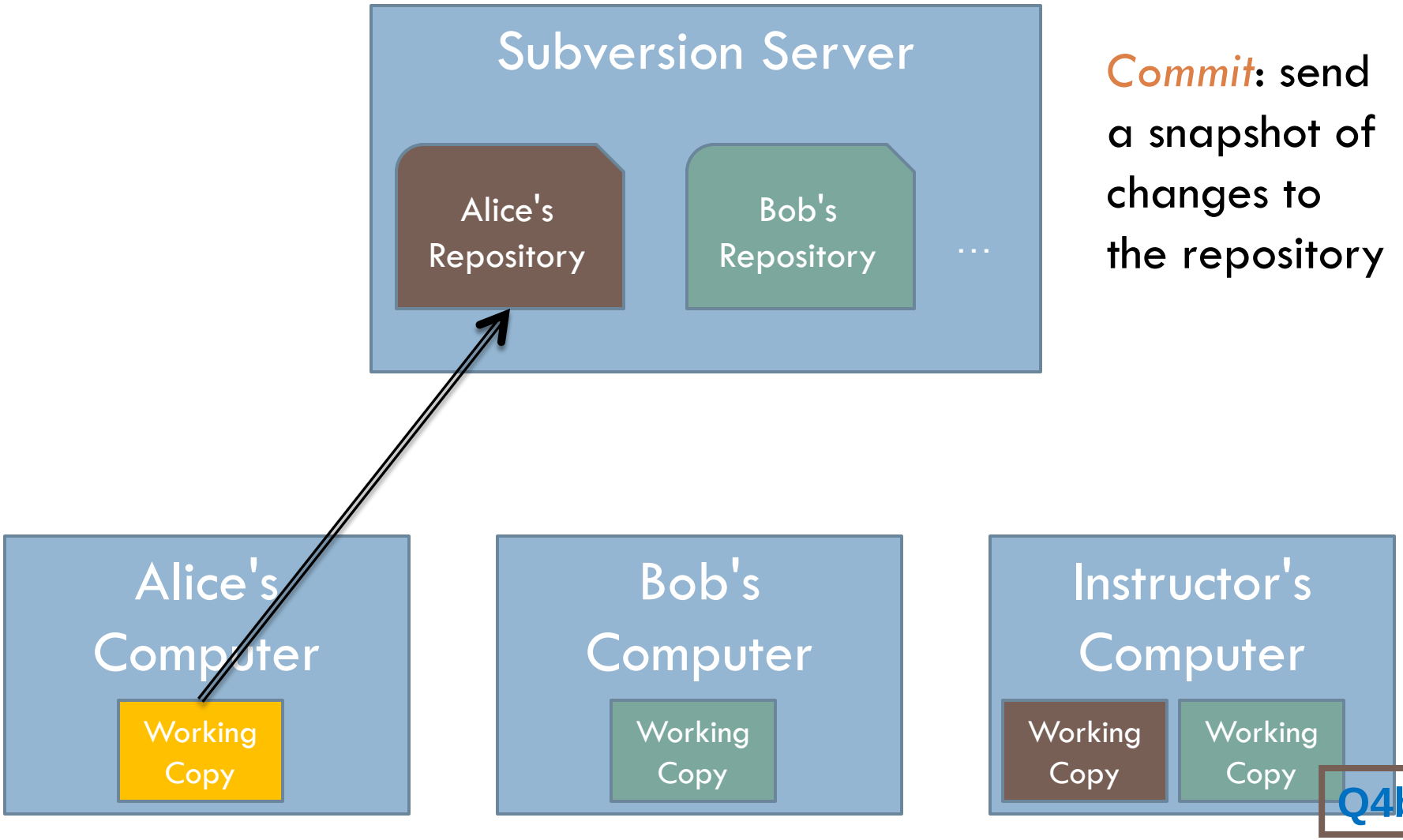


Edit: make ***independent*** changes to a working copy





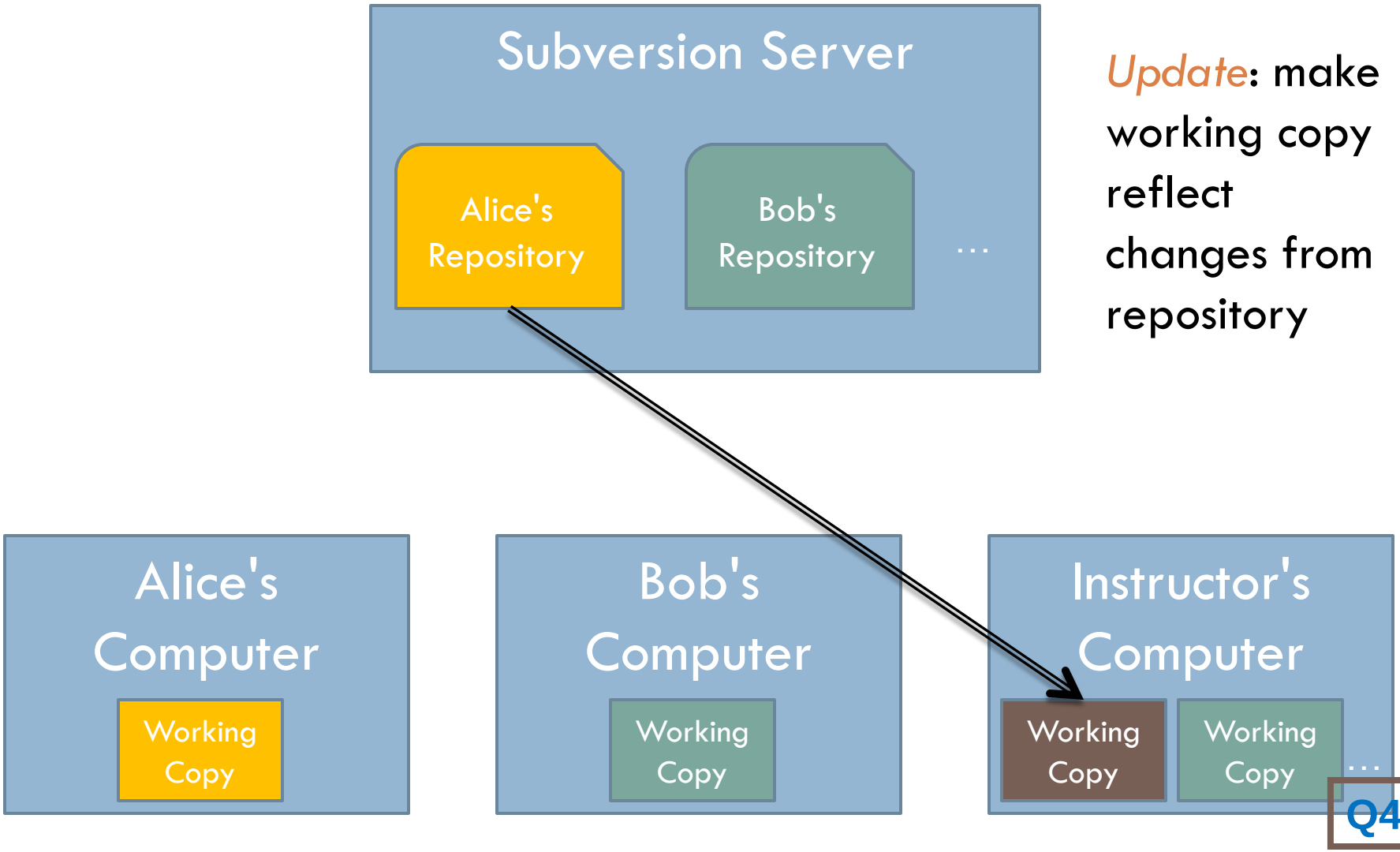
Version Control Steps—Commit



Commit: send a snapshot of changes to the repository

Q4b

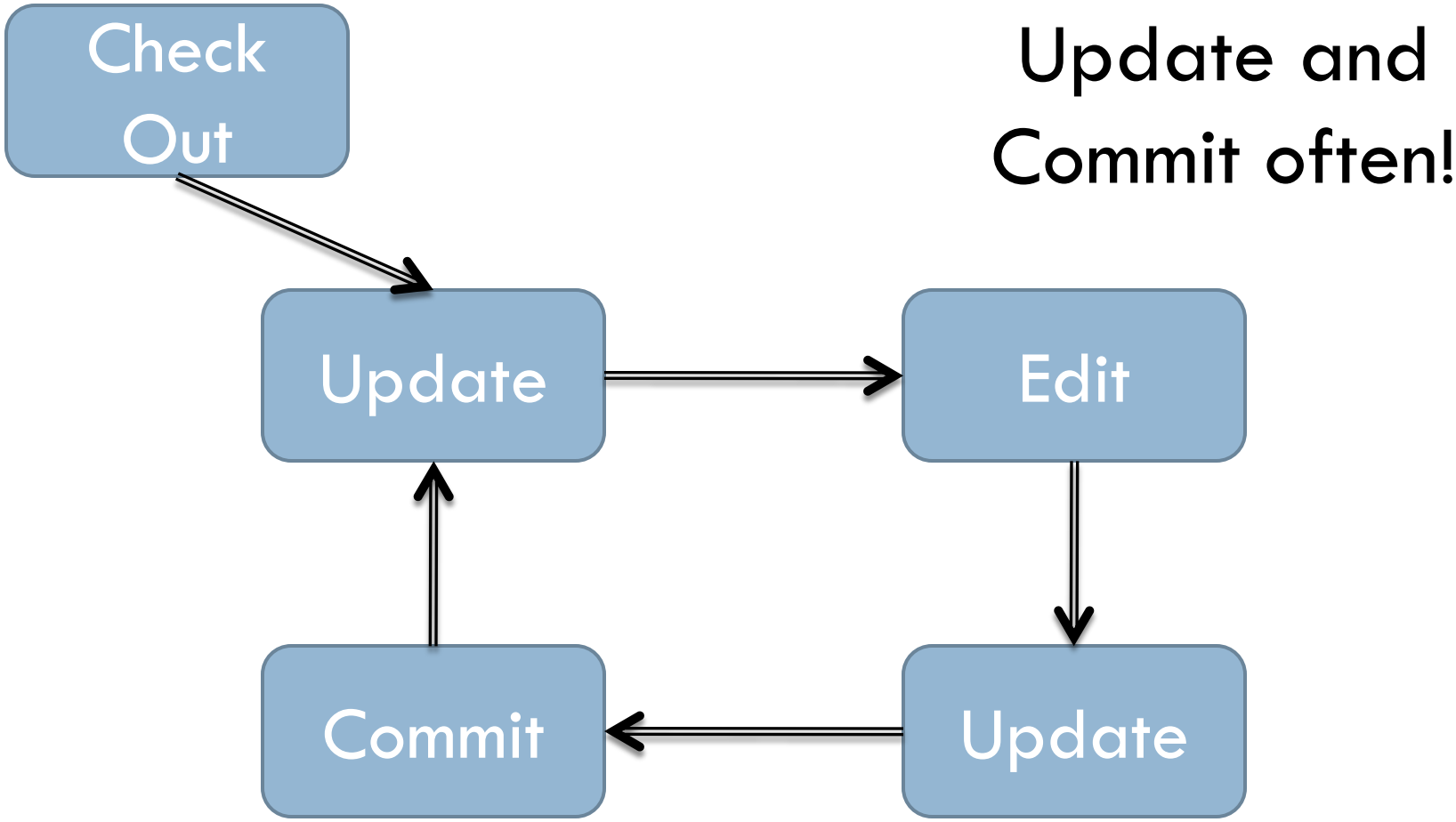
Version Control Steps—Update



Update: make working copy reflect changes from repository

Q4c

The Version Control Cycle



If you're stuck, get help and see http://www.rose-hulman.edu/class/csse/resources/Subclipse/installation.htm

Check out today's exercise

- Go to the SVN Repository view at the bottom of the workbench
  - If it is not there,
    Window→Show View→Other→SVN Repositories→OK
- Browse SVN Repository view for
  02-InputComputeOutput project
- Right-click it, and choose Checkout
  - Accept options as presented
- Expand the 02-InputComputeOutput that appears in Package Explorer (on the left-hand-side)
  - Browse the modules. We will start with ***hello.py*** (next slide)

Functions

- Examine your ***hello.py*** module in your Eclipse project.
- *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 (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

Q5

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!

Q6a

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

Q6b

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

Q7-8

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>, …, end=' ')`
- Semantics?
- Is this allowed?
  - `print("The answer is:", 7 * 3 * 2)`

A "slot" to be filled with any expression

Repeat indefinitely

Note: end = <str>

Q9

Basic program structure
The input-compute-output pattern

- Examine the ***chaos.py*** module in your Eclipse project.
- Do the TODO's in it.
  - I'll demo some of them with you.

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


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

def main():
    print "This program shows a chaotic function"
    x = float(input("Enter a number: "))
    for i in range(10):
        x = 3.9 * x * (1 - x)
        print(x)

main()
```

comments

Define a function called “main”

An *input assignment*

A *loop*

The loop’s *body*

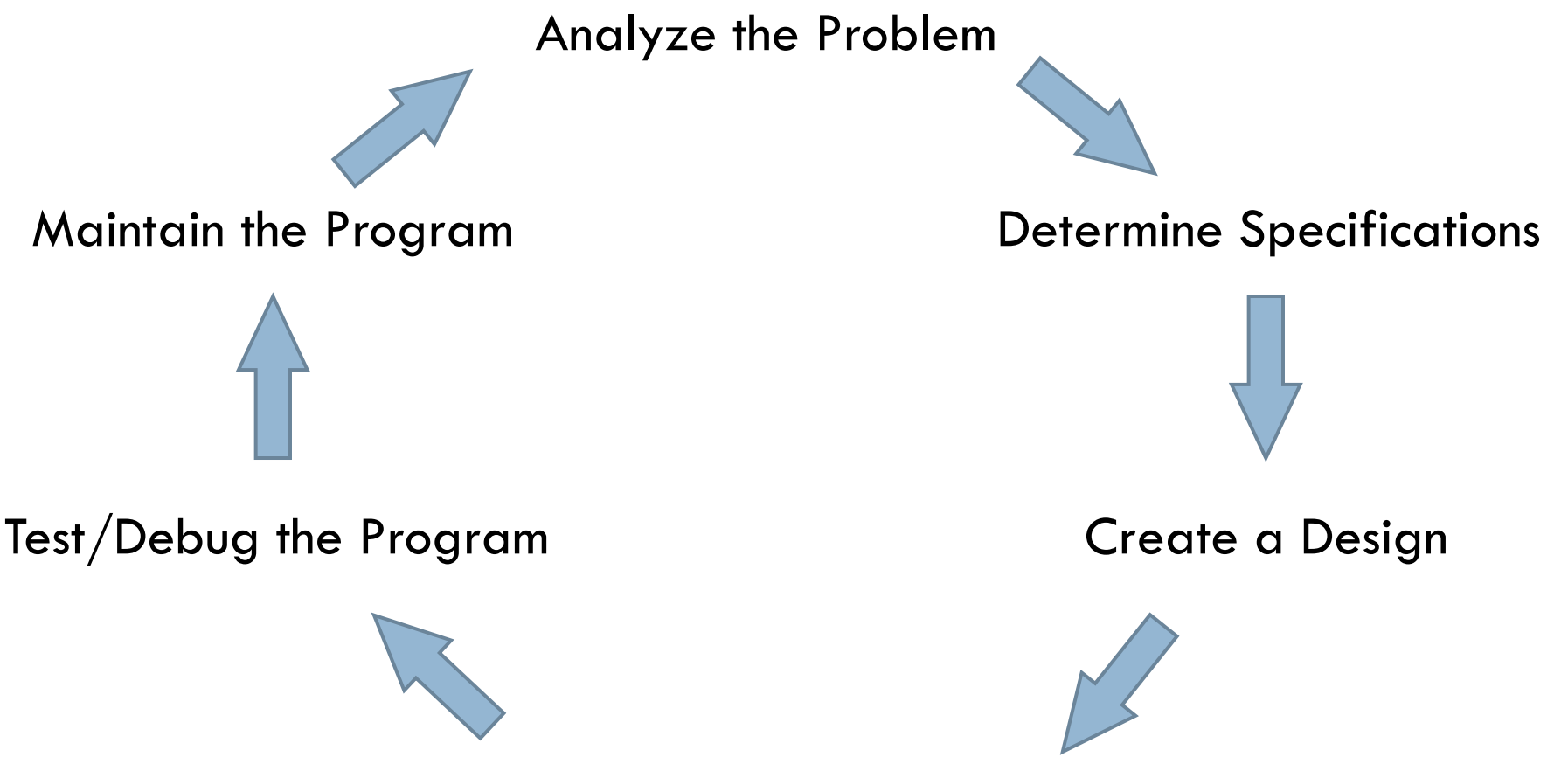
Invoke function main

A *variable* called x

Assignment statement

Q10-14

The Software Development Process



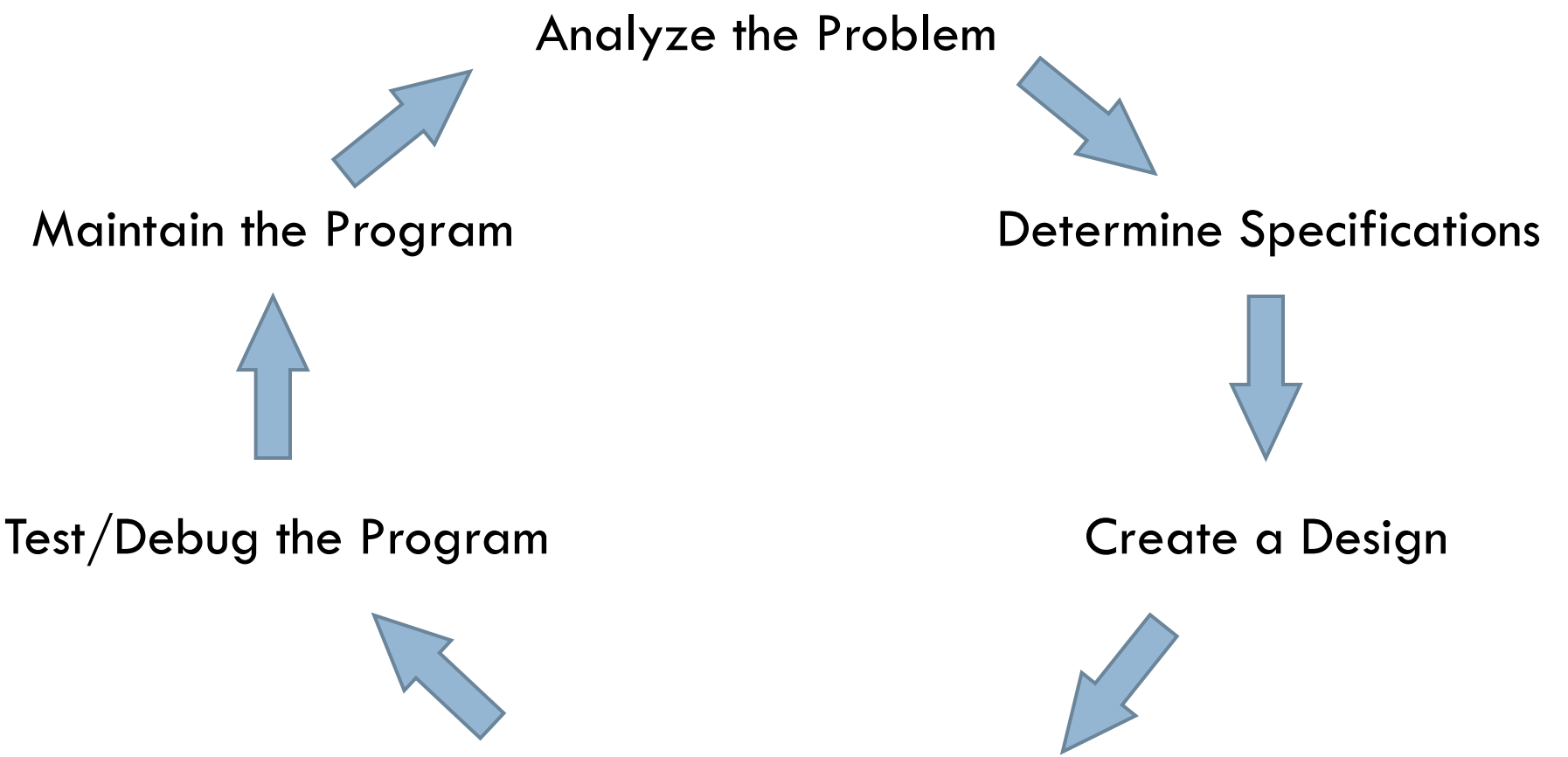
Q15

More practice at the input-compute-output pattern

- Examine the ***temperature.py*** module in your Eclipse project.
- Do the TODO's in it.
  - I'll demo some of them with you.

Road Trip!

The Software Development Process



Q16

Homework

- Hand in in-class Quiz
- Begin Homework
  - Find it from the Schedule page
  - Reading and ANGEL quiz due Tuesday.
  - Rest (programming part) due Wednesday at the time of your regular class session.