Today: Introduction to Software Development, Eclipse and Python

As you arrive:

1. Before you sit down,
get a sheet of paper with the right color:

Green: I've never written a program … and I'm proud of it!

Yellow: I've written a program or two (less than 200 lines).

Pink: I've written programs (more than 200 lines).

2. Sit next to someone with the same color sheet.

3. Start up your computer and plug it in.

4. ***Log into Angel*** and go to CSSE 120.
   Do the ***Attendance Widget*** – the PIN is on the board.

5. Go to the ***Course Schedule*** web page.
   Open the ***Slides*** for today if you wish.

The quiz lists its URL. BOOKMARK it.

Contact Before Work

- Ask your partner:

What is something interesting that you learned from a family member (parent, grandparent, sibling, cousin, aunt, …)?

When did you learn it, and how?

- Why do Contact Before Work?
  - Helps us know our teammates.
    We work better with people we know and like.
  - Helps start the meeting on time:

Outline of today's session

- Introductions: instructor, assistants, and some students
- Resources:
  - Course web site, CSSE lab (F-217) hours, `csse120-staff@rose-hulman.edu` email
- Course background:
  - What is computer science? Software development? A programming language?
- Hands-on introduction to Eclipse and Python
  - Eclipse – our Integrated Development Environment (IDE)
    - Including Subversion – our version control system, for turning in work
  - Python – our first programming language
    - A whirl-wind tour, plus your first Python program
    - Including a little *zellegraphics*

Robots starting at the *next* session

Roll Call & Introductions

- Name (nickname)
- Hometown
- Where you live on (or off) campus
- Something about you that most people in the room don't know

This means you should be answering Question #1 on the quiz.

Q1

Resources

- Course web site:

www.rose-hulman.edu/class/csse/csse120/201130

- Course schedule page – find it now (from course web site)
  - Slides, Topics
  - Activities
    - Before-class (preparation)
    - In-class
    - After-class (homework)
  - CSSE lab assistants in:
  - Email to:

```
CSSE lab: Moench F-217
   7 to 9 p.m.
   Sundays thru Thursdays
     (other times too)
```

`csse120-staff@rose-hulman.edu`

Q2-4

What is Computer Science (CS)?

- The work of computer scientists falls into three broad categories:
  - ***designing and building software***; ← *this course focuses on this*
  - developing effective ways to ***solve computing problems***, such as:
    - storing information in databases,
    - sending data over networks or
    - providing new approaches to security problems; and
  - devising new and better ways of ***using computers*** and addressing particular ***challenges in areas*** such as
    - robotics,
    - computer vision, or
    - digital forensics.

from the Association for Computing Machinery (ACM)

Q5-6

What is software development?

- Software development includes:
  - Market research
  - Gathering requirements for the proposed business solution
  - Analyzing the problem
  - Devising a plan or design for the software-based solution
  - Implementation (coding) of the software
  - Bug fixing
  - Testing the software
  - Maintenance

This course focuses on these, teaching ***good habits*** that ***scale up***.

from Wikipedia, Software Development

What is a *program*? A *programming language*?

- ***Program***
  - Detailed set of instructions
  - Step by step
  - Meant to be executed by a computer
- A ***programming language*** specifies the:
  - ***Syntax*** (form), and
  - ***Semantics*** (meaning)

  of legal statements in the language

There are thousands of computer languages. We use ***Python*** because it:

- Is ***powerful***: strong programming primitives and a huge set of libraries.
- Has a ***gentle learning curve***; you will start using it today!
- ***Plays well with others*** (e.g. COM, .NET, CORBA, Java, C) and ***runs everywhere***.
- Is ***open source***.

- See Wikipedia's History of Programming Languages for a timeline of programming languages.
- Python was introduced in 1991.
- Its predecessors include ABC, Algol 68, Icon and Modula-3.

Don't look ahead to the next slides, as that would spoil the fun!

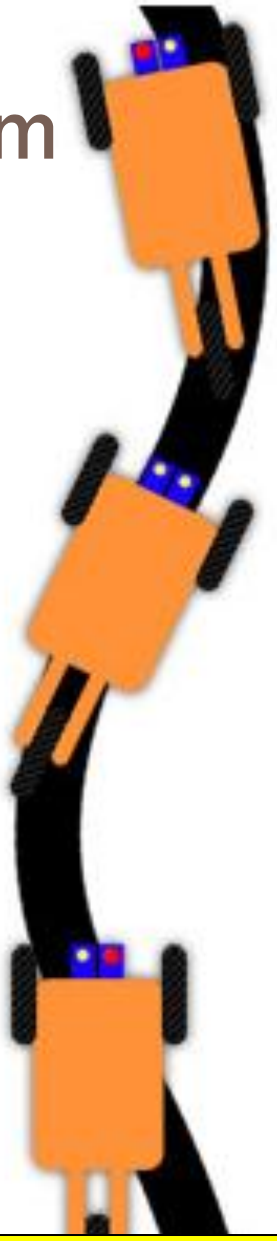
Q8-9

Your first program

- With your partner, go to the whiteboard and get a marker.
- Over the next 30 minutes, you will write a program (in English) for a ***robot that follows a black line***.
- Watch my demo of a robot that does so.
  - What physical devices on the robot allow it to move?
    - Answer: Two wheels that can move independently, each at its own speed.
  - What physical devices on the robot allow it to decide when to veer? How do those devices work?
    - Answer: Several light ("cliff") sensors. This robot is using the two front sensors that straddle the line that it is following. They shine line down and measure how much light is reflected back up.
      - Do you see why they are called "cliff" sensors?
  - What algorithm should the robot use to line-follow?
    - One answer on the next slide, but many algorithms are reasonable!

Your first program

- Here (to the right) is one line-following algorithm
  - There are many other reasonable algorithms.
  - What's best depends on the nature of the line to follow and the sensors available.
  - Important note: we can't write this program until we know what *algorithm* we intend to implement



Left light sensor sees *white* (light)
Right light sensor sees *black* (dark)
Action:
- *Veer right*

This is called bang-bang control. See why?

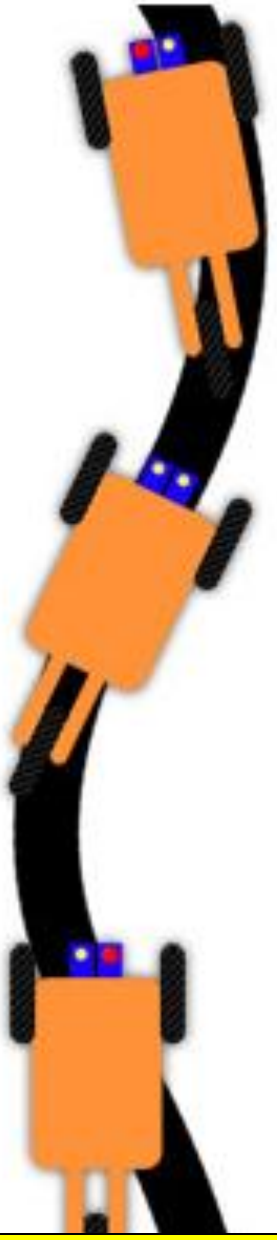
Both light sensors see *white*
(the robot is straddling the line)
Action:
- *Go straight ahead*

Left light sensor sees *black* (dark)
Right light sensor sees *white* (light)
Action:
- *Veer left*

Imagine that the sensors are a bit farther apart than shown here, as that is the case for our Create robot.

Write *main*

- Our programs traditionally begin in what's called *main.*
- Write *main* (in English).
  - Use fewer than 10 sentences.
  - It's perfectly OK if some of your sentences refer to functions (procedures) that you have not yet defined, but whose name makes it obvious what it should do.



Left light sensor sees *white* (light)
Right light sensor sees *black* (dark)
Action:
- *Veer right*

This is called bang-bang control. See why?

Both light sensors see *white*
(the robot is straddling the line)
Action:
- *Go straight ahead*

Left light sensor sees *black* (dark)
Right light sensor sees *white* (light)
Action:
- *Veer left*

Imagine that the sensors are a bit farther apart than shown here, as that is the case for our Create robot.

Let's develop *main* together

Repeat the following forever:

```
left_light = read_sensor(left_front)

right_light = read_sensor(right_front)

if left_light is “light (similar to all-white)”
    and
  right_light is “dark (similar to all-black)”:
        veer_right()

if ... go_straight()

if ... veer_left()
```

“**sleep**” the program briefly (but let the robot continue moving) so that you don't flood the robot with requests/commands

Now write the code for the `veer_right` ***function***. Then the `go_straight` and `veer_left` functions.

Approach 1

Let's develop *veer_right* together

```
def veer_right():
    on(left_motor, 100)
    on(right_motor, 50)

def go_straight():
    on(left_motor, 100)
    on(right_motor, 100)

def veer_left():
    on(left_motor, 50)
    on(right_motor, 100)
```

We ***abstract*** the two **`on`** commands into a single **`move`** function.
A simple but powerful idea!

```
def veer_right():
    move(100, 50)

def go_straight():
    move(100, 100)

def veer_left():
    move(50, 100)
```

Approach 2

```
def move(left_speed, right_speed):
    on(left_motor, left_speed)
    on(right_motor, right_speed)
```

Post-Mortem

- You have now experienced many of the fundamental concepts of procedural programming:
  - **Loops**: **"repeat forever"**
  - **Function calls**:

    Arguments

    ```
    on(left_motor, 100)
    veer_right()
    ```
  - **Function definitions**:

    Parameters

    ```
    def move(left_speed, right_speed):
        on(left_motor, left_speed)
        on(right_motor, right_speed)
    ```
  - **Parameters** and **arguments**: shown above
  - **Returned values**, **variables**, **assignment**:

    ```
    left_light = read_sensor(left_front)
    ```
  - **Conditional control flow**: `if ...`

Integrated Development Environments (IDEs) – Outline

- 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

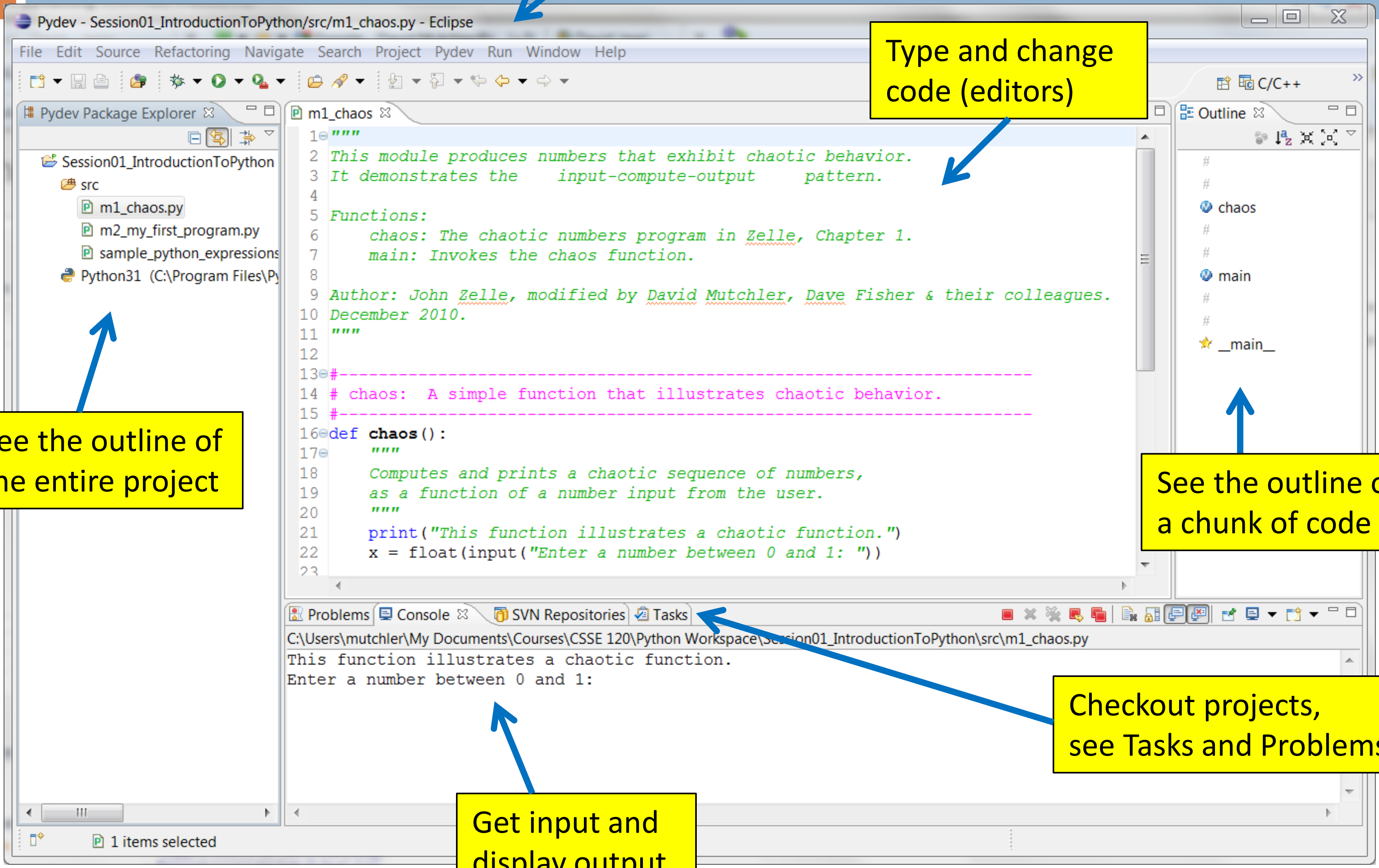
The next slides address these points about IDEs.

IDEs – What are they?

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

They try to make it easy to:

Compile, run, debug, document, and more


Pydev - Session01_IntroductionToPython/src/m1_chaos.py - Eclipse
File Edit Source Refactoring Navigate Search Project Pydev Run Window Help
C/C++
Pydev Package Explorer
Session01_IntroductionToPython
src
m1_chaos.py
m2_my_first_program.py
sample_python_expressions
Python31 (C:\Program Files\Py
m1_chaos
Outline
chaos
main
__main__
Problems
Console
SVN Repositories
Tasks
1 items selected

```
"""
This module produces numbers that exhibit chaotic behavior.
It demonstrates the    input-compute-output     pattern.

Functions:
    chaos: The chaotic numbers program in Zelle, Chapter 1.
    main: Invokes the chaos function.

Author: John Zelle, modified by David Mutchler, Dave Fisher & their colleagues.
December 2010.
"""

#-----------------------------------------------------------------------
# chaos:  A simple function that illustrates chaotic behavior.
#-----------------------------------------------------------------------
def chaos():
    """
    Computes and prints a chaotic sequence of numbers,
    as a function of a number input from the user.
    """
    print("This function illustrates a chaotic function.")
    x = float(input("Enter a number between 0 and 1: "))
```

C:\Users\mutchler\My Documents\Courses\CSSE 120\Python Workspace\Session01_IntroductionToPython\src\m1_chaos.py

```
This function illustrates a chaotic function.
Enter a number between 0 and 1:
```

Type and change code (editors)

See the outline of the entire project

See the outline of a chunk of code

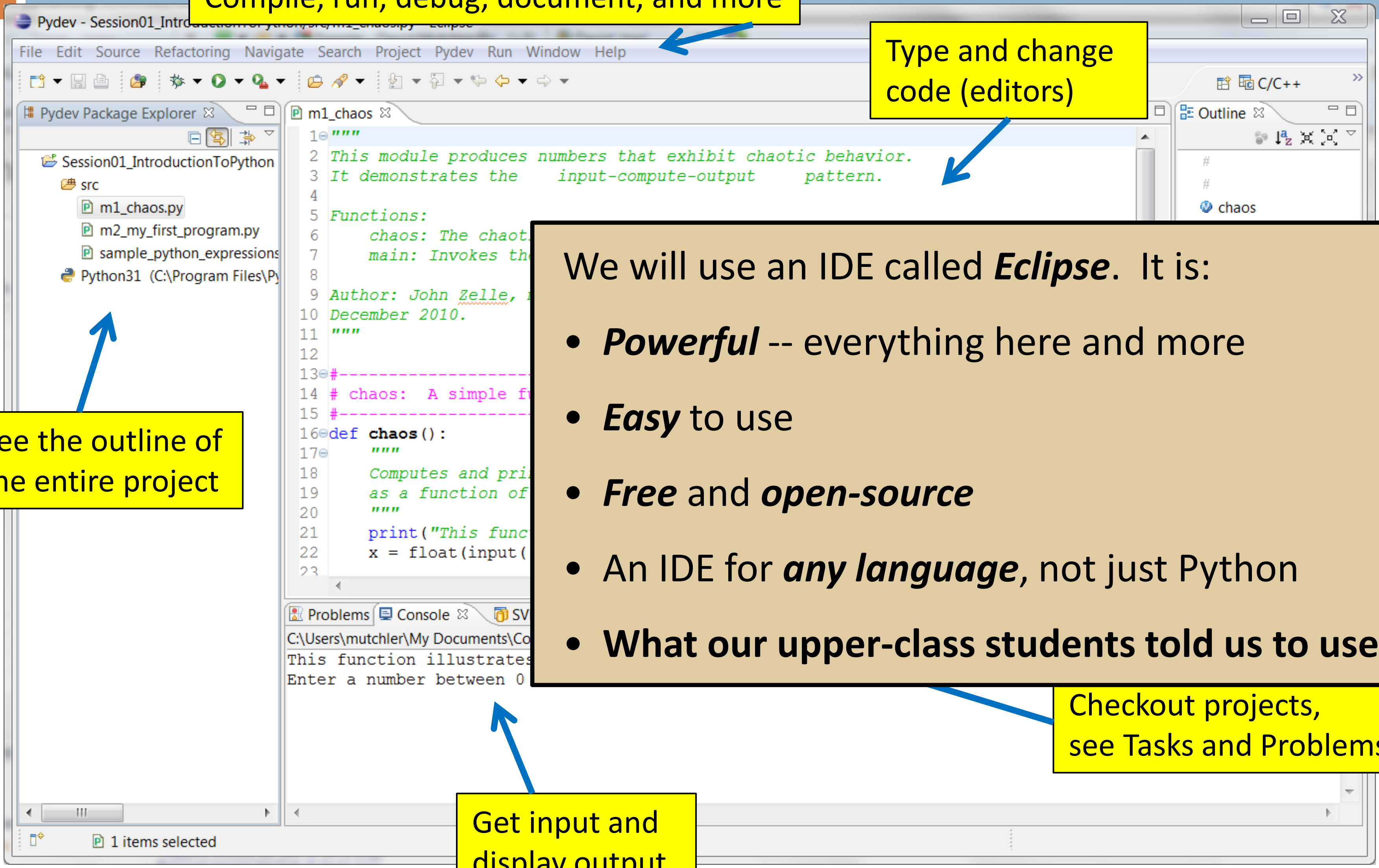
Checkout projects, see Tasks and Problems

Get input and display output

IDEs – Why use one? Why Eclipse?

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

They try to make it easy to:



Pydev - Session01_Intro
File Edit Source Refactoring Navigate Search Project Pydev Run Window Help
C/C++
Pydev Package Explorer
Session01_IntroductionToPython
src
m1_chaos.py
m2_my_first_program.py
sample_python_expressions
Python31 (C:\Program Files\Py
m1_chaos
Outline
chaos
Problems
Console
SV
1 items selected

```
"""
This module produces numbers that exhibit chaotic behavior.
It demonstrates the    input-compute-output     pattern.

Functions:
    chaos: The chaot
    main: Invokes th

Author: John Zelle, 
December 2010.
"""


#------------------
# chaos:  A simple f
#------------------
def chaos():
    """
    Computes and pri
    as a function of
    """
    print("This func
    x = float(input(
```

```
C:\Users\mutchler\My Documents\Co
This function illustrates
Enter a number between 0
```

Compile, run, debug, document, and more

Type and change code (editors)

See the outline of the entire project

Get input and display output

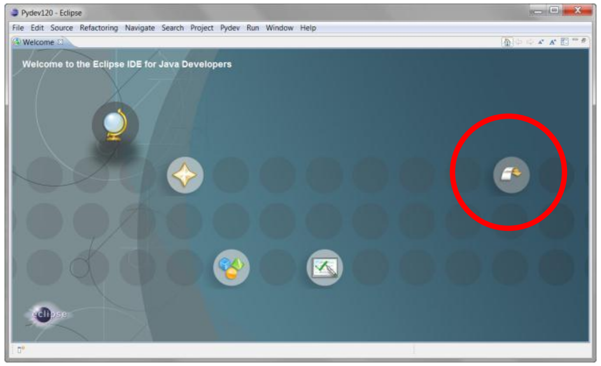
Checkout projects, see Tasks and Problems

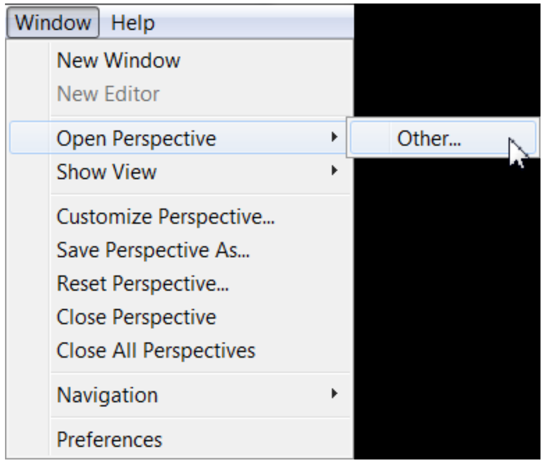
We will use an IDE called ***Eclipse***. It is:

- ***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!**

Open Eclipse

- Your instructor will show you're the highlights.
  - They are summarized on the next several slides.

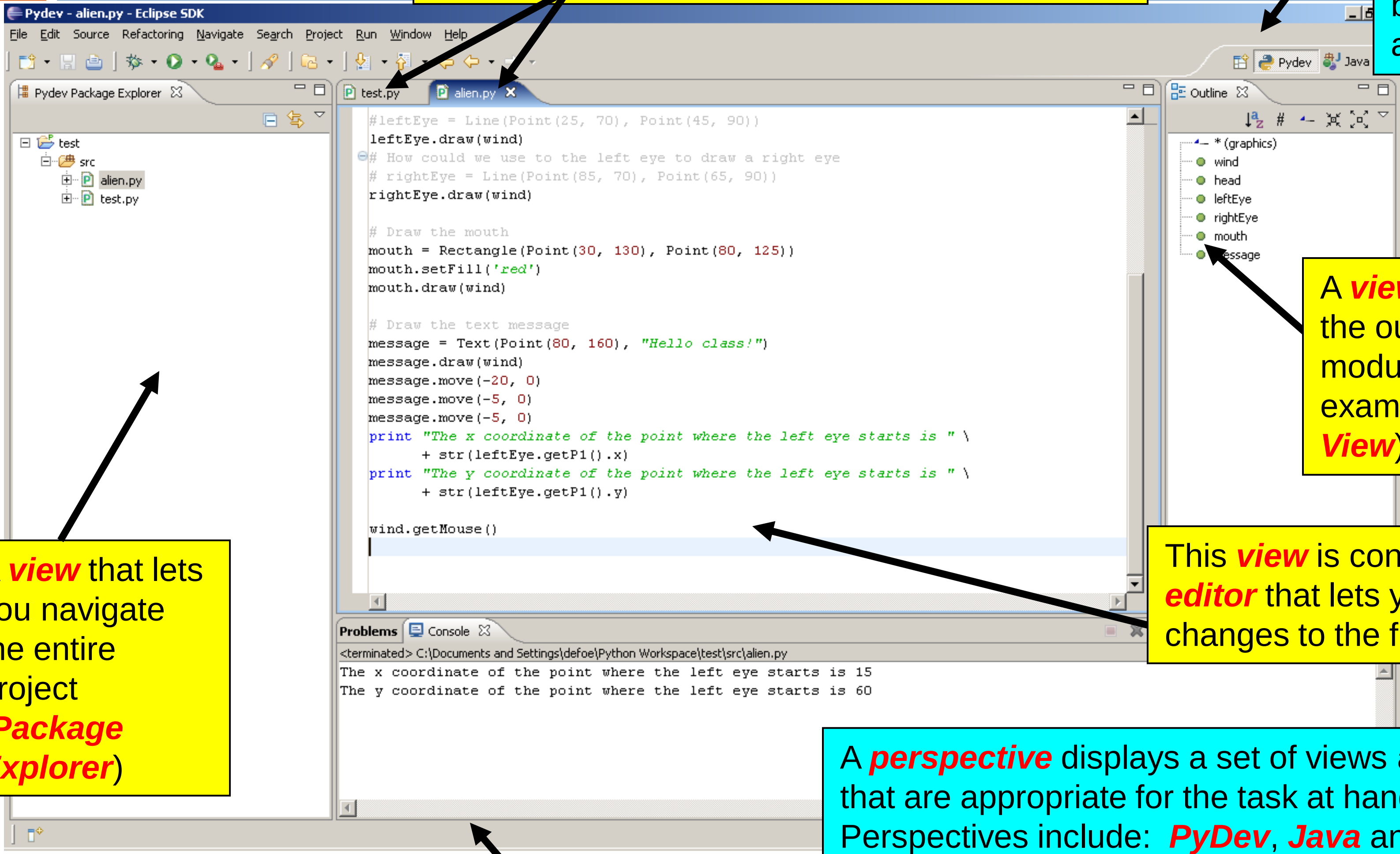

Pydev120 - Eclipse
File Edit Source Refactoring Navigate Search Project Pydev Run Window Help
Welcome
Welcome to the Eclipse IDE for Java Developers



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

Views, editors, perspectives



```
#leftEye = Line(Point(25, 70), Point(45, 90))
leftEye.draw(wind)
# How could we use to the left eye to draw a right eye
# rightEye = Line(Point(85, 70), Point(65, 90))
rightEye.draw(wind)

# Draw the mouth
mouth = Rectangle(Point(30, 130), Point(80, 125))
mouth.setFill('red')
mouth.draw(wind)

# Draw the text message
message = Text(Point(80, 160), "Hello class!")
message.draw(wind)
message.move(-20, 0)
message.move(-5, 0)
message.move(-5, 0)
print "The x coordinate of the point where the left eye starts is " \
      + str(leftEye.getP1().x)
print "The y coordinate of the point where the left eye starts is " \
      + str(leftEye.getP1().y)

wind.getMouse()
```

```
<terminated> C:\Documents and Settings\defoe\Python Workspace\test\src\alien.py
The x coordinate of the point where the left eye starts is 15
The y coordinate of the point where the left eye starts is 60
```

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***)

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

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

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

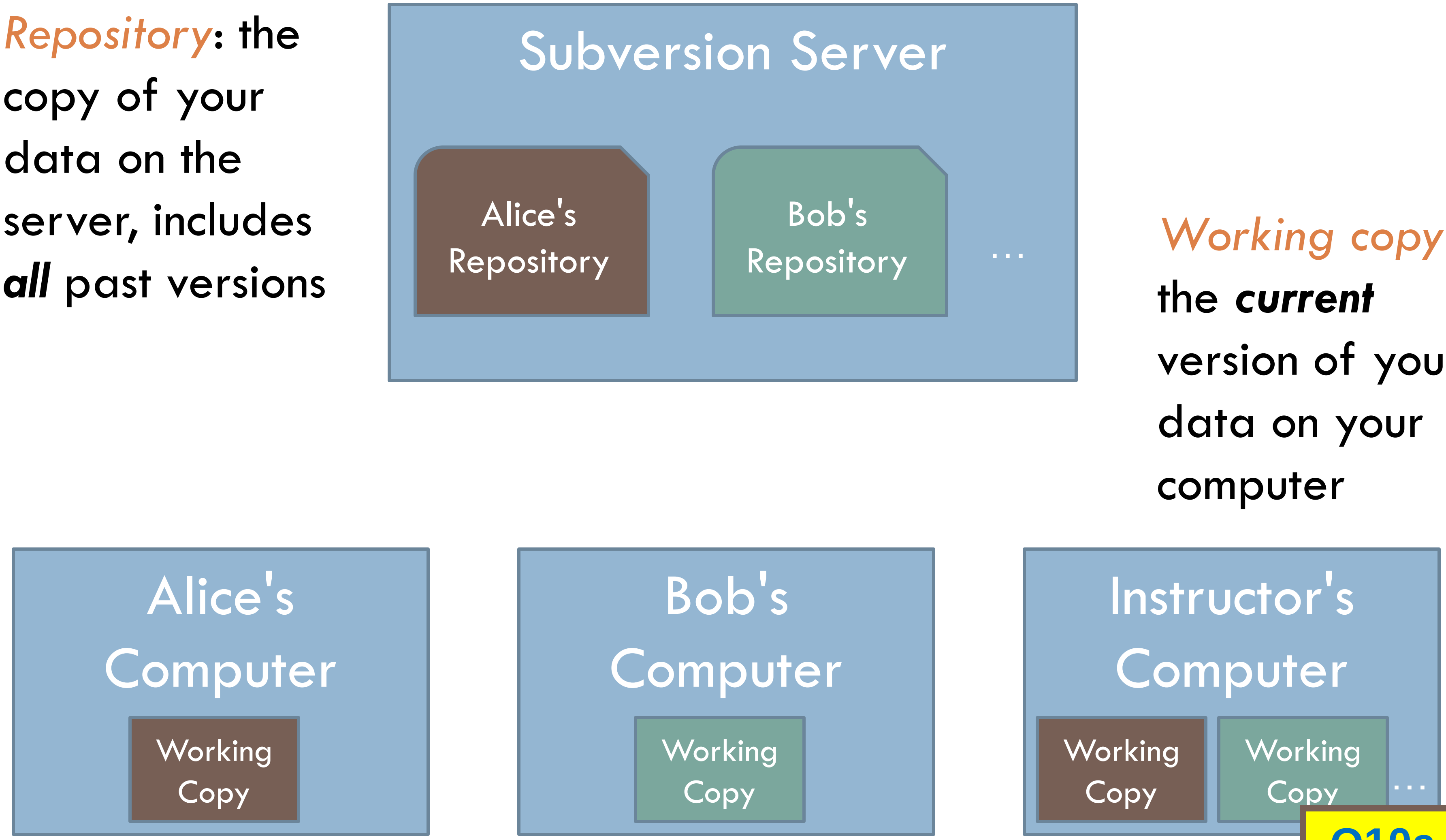
- Store “snapshots” of all the changes to a project over time
- Benefits:
  - Multiple users
    - Multiple users can share work on a project
    - Record who made what changes to a project
    - Provide help in resolving conflicts between what the multiple users do
    - Maintain multiple different 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

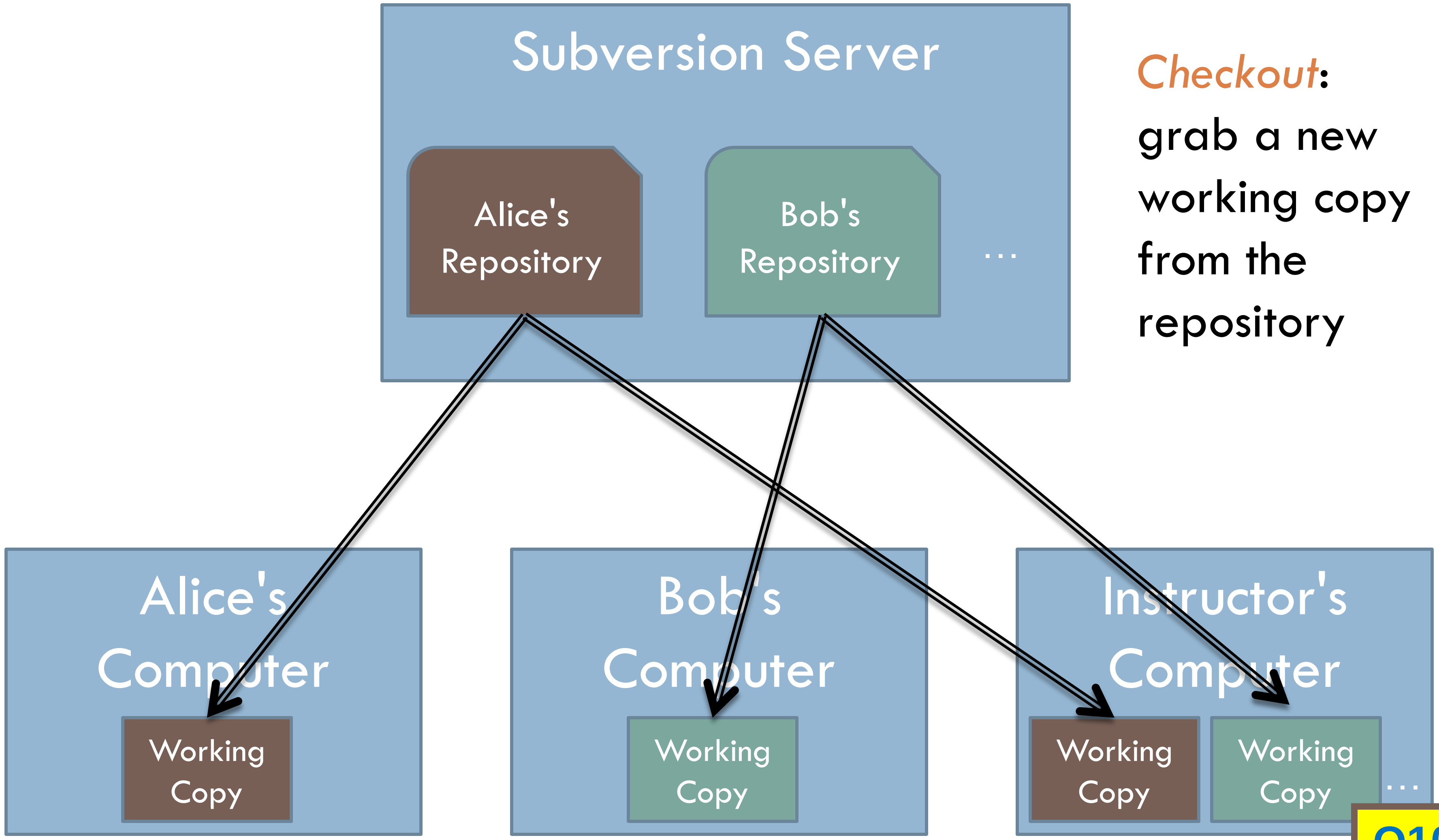
Repository: the copy of your data on the server, includes ***all*** past versions



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

Q10a-b

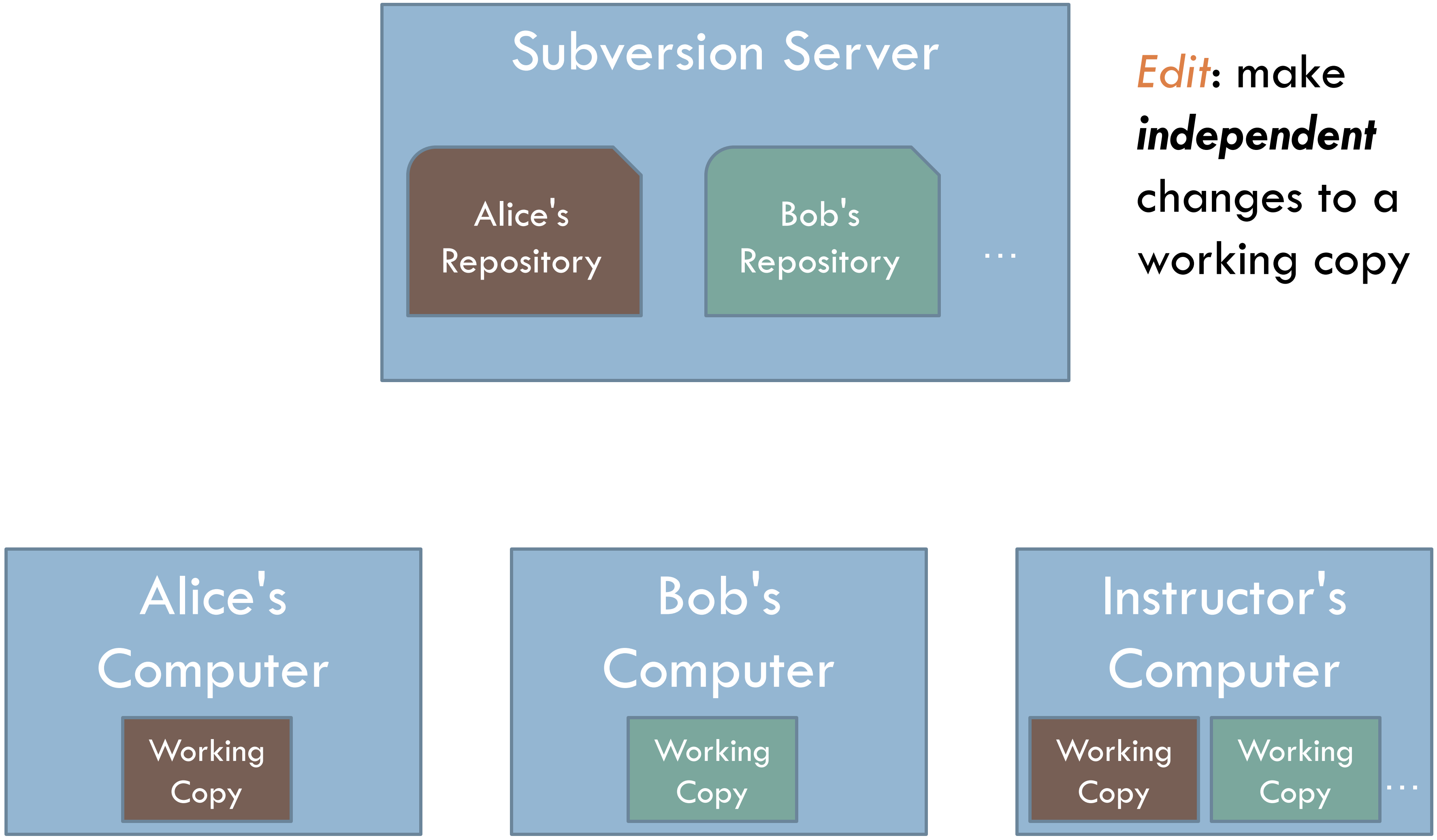
Version Control Steps—Checkout



Checkout: grab a new working copy from the repository

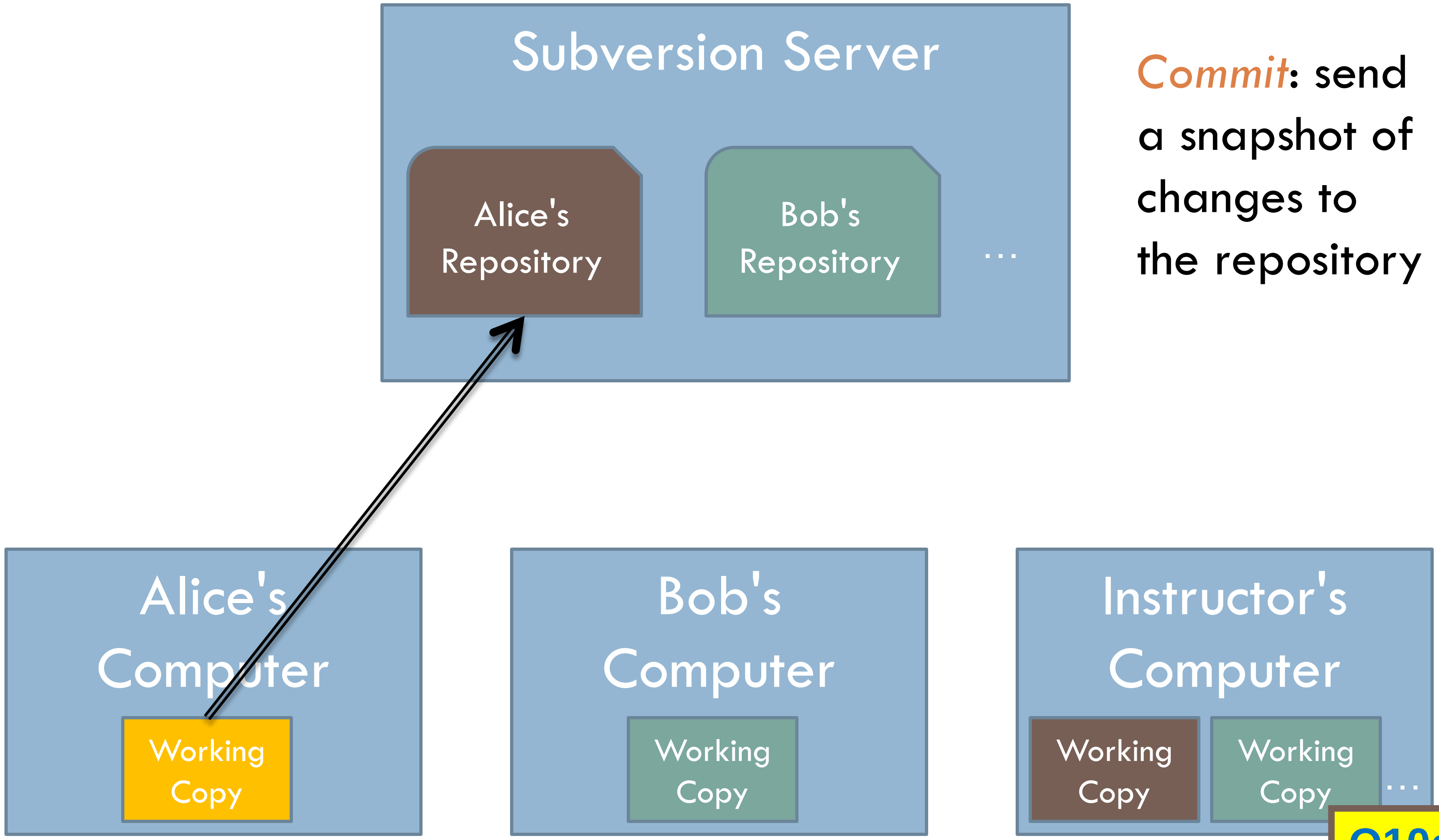
Q10c

Version Control Steps—Edit



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

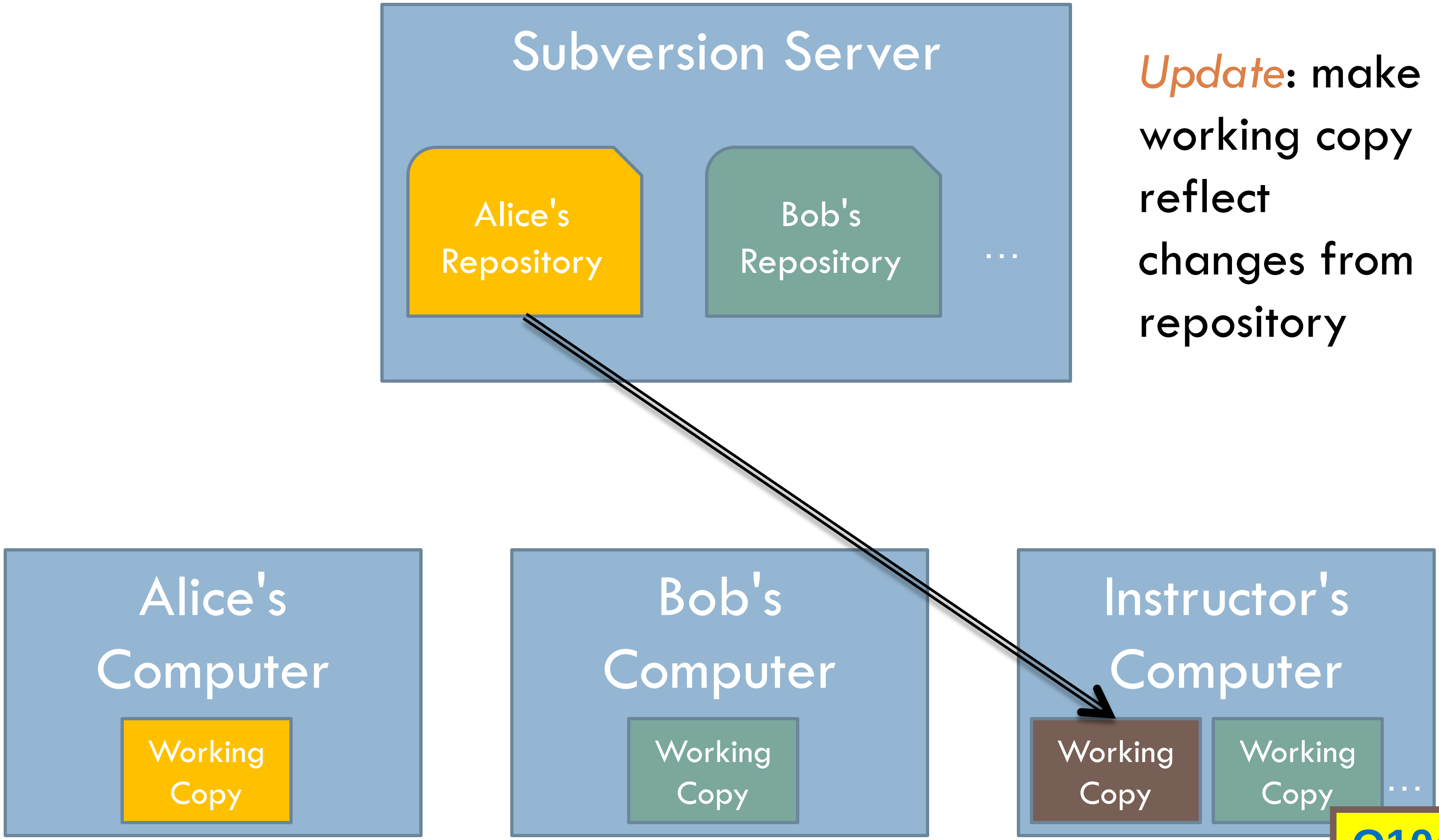
Version Control Steps—Commit



Commit: send a snapshot of changes to the repository

Q10d

Version Control Steps—Update



Update: make working copy reflect changes from repository

Q10e

Checkout today's project:

`Session01_IntroductionToPython`

Are you in the Pydev perspective? If not:

`Window ~ Open Perspective ~ Other` then `Pydev`

Messed up views? If so:

`Window ~ Reset Perspective`

Troubles getting today's project? If so:

No SVN repositories view (tab)? If it is not there:

`Window ~ Show View ~ Other`
then `SVN ~ SVN Repositories`

1. ***In your SVN repositories view (tab), expand your repository*** *(the top-level item) if not already expanded.*
   - If no repository, perhaps you are in the wrong Workspace. Get help.
2. ***Right-click on today's project****, then select* `Checkout`.
   *Press **OK** as needed.* The project shows up in the
   `Pydev Package Explorer`
   to the right. Expand and browse the modules under `src` as desired.

Your first Python example: *chaos!*

```
def main():
    """ Calls a function (chaos) which shows a chaotic sequence. """
    chaos()

def chaos():
    """
    Computes and prints a chaotic sequence of numbers,
    as a function of a number input from the user.
    """
    print('This function illustrates a chaotic function.')
    x = float(input('Enter a number between 0 and 1: '))

    for k in range(20): #@UnusedVariable
        x = 3.9 * x * (1 - x)
        print(x)

    print('Examine the sequence of numbers printed.')
    print('Does it appear chaotic?')
```

Q11

Rest of Session

- ***Check your Quiz answers*** versus the solution
  - An assistant may check your Quiz to ensure you are using the Quizzes appropriately
- ***Work on today's homework***
  - Ask questions as needed!
- **Sources of help after class:**
  - **Assistants in the CSSE lab**

    ```
    CSSE lab: Moench F-217
    7 to 9 p.m.
    Sundays thru Thursdays
    ```
    - And other times as well (see link on the course home page)
  - **Email** `csse120-staff@rose-hulman.edu`
    - You get faster response from the above than from just your instru

Q12