As you arrive:

Today: Introduction to Software Development, Eclipse and Python

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. Go to the ***Course Schedule*** web page.
   Open the ***Slides*** for today if you wish.
5. When the attendance sheet reaches you, put a check by your name.

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, assistants in CSSE labs (F-217 and D-219), **`csse120-staff@rose-hulman.edu`** email
- Course background:
  - What is software engineering? 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
    - The Python Console
    - Your first Python program

Virtual robots today, real ones in a few sessions from today.

Roll Call & Introductions

- Find 2 or 3 other people in the room and ask each to say:
  - Her name (nickname)
  - Her hometown
  - Where she lives on (or off) campus
  - Something about her that most people in the room don't know

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

Resources

- Course web site:

  www.rose-hulman.edu/class/csse/csse120/201230

- Course schedule page – find it now (from course web site)
  - Topics
  - Homework
  - These slides
  - More
- CSSE lab assistants in:

CSSE labs:
Moench F-217 and D-219
7 to 9 p.m.
Sundays thru Thursdays

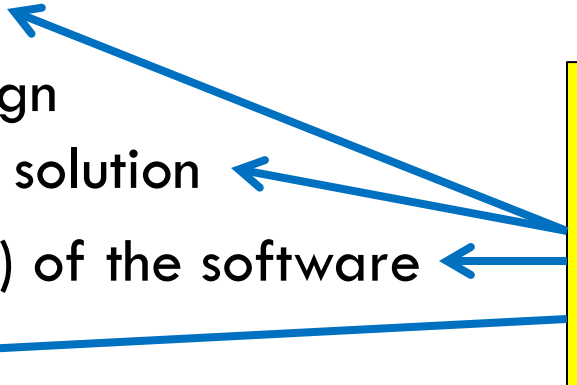
- Email to:

csse120-staff@rose-hulman.edu

Q2-4

What is *software engineering*?

- Software engineering 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, sometimes called ***software development***. We teach ***good habits*** that ***scale up***.

from Wikipedia, Software Development

Q5

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!

Q6-7

Your first program

- With your partners, go to the whiteboard and get a marker.
- Over the next 20 minutes, you will write a program (in English) for a ***robot that follows a black line for 10 seconds, turns 180 degrees, then follows the same black line for 5 more seconds.***
- 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 light 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!

The *main* function

Specification: ***A robot follows a black line for 10 seconds, turns 180 degrees, then follows the same black line for 5 more seconds.***

- Our programs traditionally begin in what's called *main*.
- Let's write ***main*** together (in English, or maybe in "pseudo-code")
  - First: You try. Use about 4 sentences or phrases.
  - 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.

Version 1

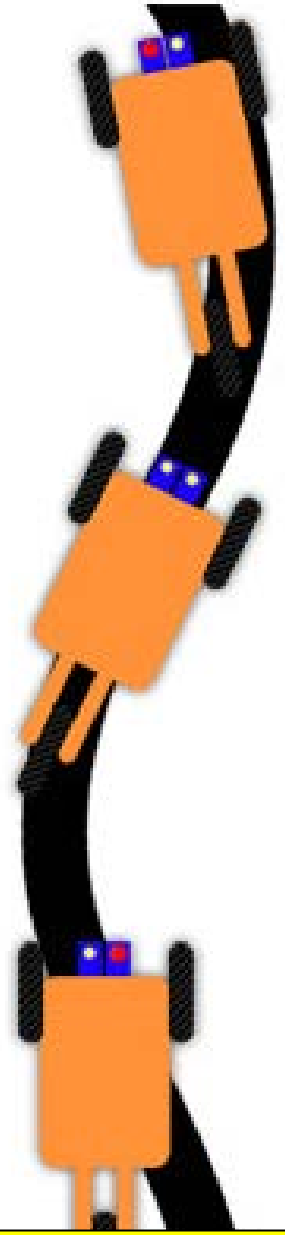
```
def main():
    calibrate_cliff_sensors()
    follow_line_10_seconds()
    turn_180_degrees()
    follow_line_5_seconds()
```

Version 2 (better, because it allows *re-use* of the functions)

```
def main():
    calibrate_cliff_sensors()
    follow_line(10)
    turn(180)
    follow_line(5)
```

Line-following

- 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.

Let's develop *follow_line* together

```
def follow_line(seconds):
    start_time = current_time()
    repeat until current_time() - start_time >= seconds:
        left_sensor = read_sensor("left front")
        right_sensor = read_sensor("right front")
        if left_sensor   indicates "white"
            and
        right_sensor    indicates "black":
            veer_right()
        if ...:
            go_straight()
        if ...:
            veer_left()
        sleep  a bit (so as not to flood the sensors)
```

Now let's write the code for the **veer_right** ***function***. Then the **go_straight** and **veer_left** functions. This process is called ***top-down design***.

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

The approach that you just saw (breaking problems into sub-problems) is called ***procedural decomposition***, aka ***top-down design***.

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

```
on(left_motor, 100)
veer_right()
```

  - **Function definitions**:

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

Arguments

Parameters

  - **Parameters** and **arguments**: shown above
  - **Returned values**, **variables**, **assignment**:

```
left_light = read_sensor(left_front)
```

  - **Conditional control flow**: `if ...`

Q8-9, turn in quiz

Break and SVN password

- You will store your programs using a tool called SVN.
- ***Select a password for your SVN account.***
  - It can be your Kerberos password or not, your choice, but it is not tied to your Kerberos password. (So changing your Kerberos password does *not* change your SVN password.)
  - Choose something that you can remember!
  - If you ever need your SVN password reset, just ask your instructor.
- ***During this break,*** come to an assistant's computer to run the short program in which you ***type in your password.***
  - When you type, it will LOOK like nothing is happening, but it is! Just type carefully!
  - It will ask you to confirm your choice by re-typing it, so no problem if you make a mistake (just start over).

Fix a bug in your setup of Eclipse

- ***Follow the instructions on the handout "A workaround for an error in our installation of Eclipse."***
- When you are done, STAND UP and help someone else who is still seated.

Eclipse by Pictures

- ***Follow the instructions on the handout "Your first Eclipse session, by Pictures."***
- When you are done, STAND UP and help someone else who is still seated.
  - If your setup is much different from what your laptop came with, or if you are an upper-class student, you'll do a more elaborate setup as part of your homework

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

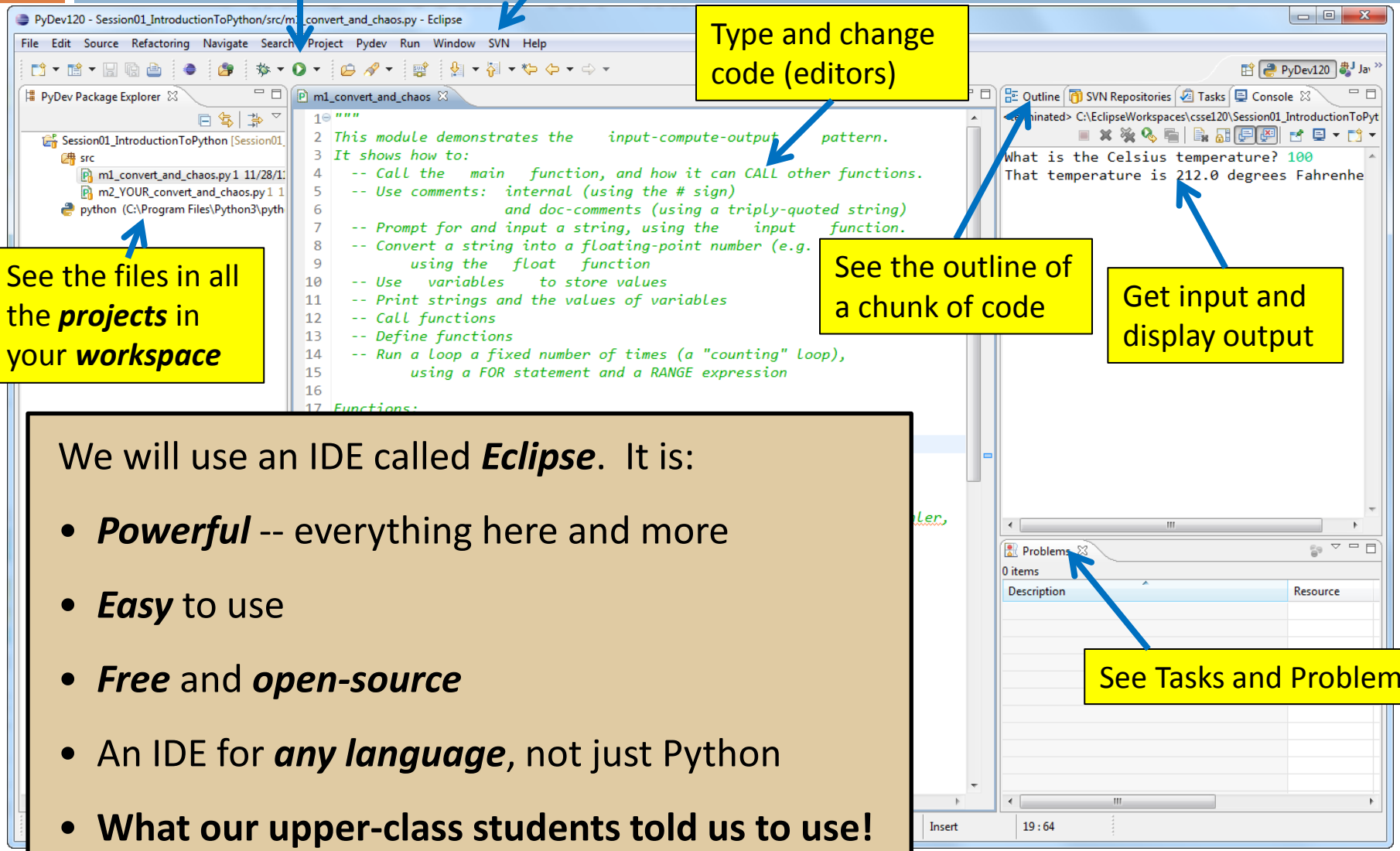
The next slide addresses these points about IDEs.

IDEs – What are they?

Checkout projects, run, debug, document, compile & more

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

They try to make it easy to:



Type and change code (editors)

See the files in all the ***projects*** in your ***workspace***

See the outline of a chunk of code

Get input and display output

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!**

The PyDev console

- If you want to try out a single statement, the PyDev Console makes that handy!
- ***Follow the instructions on the handout "The PyDev Console, by Pictures"***
  - Ask questions! You are NOT supposed to know everything yet!
  - You'll finish this exercise as part of your homework.

Checkout today's project

- At each session, we supply a ***project*** in which you work, by doing the TODO's in the files in the project.
- Each of you has a personal ***repository*** with your own copy of the initial project.
- We'll use a tool called ***SVN*** (Subversion Version Control) to ***check out*** the project and (at the next session) turn it in.
- ***Follow the instructions on the handout "Your first SVN session in Eclipse, by Pictures."***
- When you are done, STAND UP and help someone else who is still seated.

Rest of Session

- ***Work on today's homework***
  - Find it now!

    From the ***Course Schedule Page*** that you bookmarked:

    `www.rose-hulman.edu/class/csse/csse120/201230/Schedule/Schedule.htm`
  - Ask questions as needed!
- **Sources of help after class:**
  - **Assistants in the CSSE lab**

    CSSE lab: Moench F-217
    (and sometimes D-219 too)
    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 instructor