As you arrive

- Start up your computer
- Bookmark the course web site:

  www.rose-hulman.edu/class/csse/csse120/201030robotics/

CSSE 120 DAY 1

Introduction to Software Development - Robotics

Outline

- Roll call
- Introductions
- Introduction to course
- Hands-on Introduction to:
  - Python, including *zellegraphics*
  - Create Robots

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 Questions #1 and 2 on the quiz → **Q1-2**

Administrivia

- Course *web site* – note that this is a Robotics section
  www.rose-hulman.edu/class/csse/csse120/201030robotics/ (bookmark it now)
- *Syllabus*
  - Student assistants in F-217
    - Sunday through Thursday 7 p.m. to 11 p.m.
    - Monday, Tuesday, Thursday and Friday 7th to 9th
      - Consider routinely doing your homework in F-217 evenings
  - Grading plan, attendance policy
  - Late work policy
  - Email to csse120-staff@rose-hulman.edu
  - Honesty policy

No background in programming or robotics is assumed.

Q3-4

Administrivia, continued

- *Course Schedule* – find it now (from course web site)
  - *Homework 1* due at start of next class
    - Reading and Angel quiz on it
      - Don't get hung up on the reading. If necessary, skim. *Always* do the Angel quiz (you can take it up to 4 times).
    - Programming part
      - Put your **name** in a comment at the top of your Python file
      - Style requirements will be added as course progresses
      - Turn in the programming part via a Drop Box on Angel
  - These *slides* – find them now (from Course Schedule)
  - *Evening exams:*
    - Thursday, April 1, 7 to 9 p.m. (Thursday before spring break)
    - Thursday, April 29, 7 to 9 p.m.

In the future, *programming* assigned *Monday* is not due until *Wednesday noon.*

Q5-6

Administrivia, continued again

- ***Angel ~ Lessons***
  - ***Homework***
    - Where you take your Angel quizzes on the reading
    - Drop Boxes for other homework
  - ***Anonymous Suggestions Box***

How to succeed in CSSE120

- Read the textbook before each class
  - Take the ANGEL quiz over the reading
    - If you don't do well, read again and retake quiz
  - Ask questions on what you don't understand
  - Try out the code if that is helpful to you
- Start early on the programming assignments
  - Don't be satisfied with merely getting your code to "work." ***Be sure you understand it.*** If you don't, ask!
- Work and learn with other students
  - But don't let them do your work for you
- Take advantage of instructor office hours and student assistant lab hours

Basic Definitions

- Computer
  - Device for manipulating data
  - Under control of a changeable program
- Program
  - Detailed set of instructions
  - Step by step
  - Meant to be executed by a computer

The two ends of programming

1. See the Big Picture
2. Get the Details Right

Many important programming techniques are methods of getting from #1 to #2.

Some Computer Science Questions

- What can be computed?
- How to compute it efficiently?
- What is the best way to turn a mass of raw data into usable information?

Q8

What is an Algorithm?

- Step-by-step procedure for accomplishing something
- Presented at the right level of detail (and in the right language) for the one who will execute it

Algorithm Analogy -- Recipe

- Bake a cake
  - Instructions for an experienced cook
  - Instructions for a 7-year-old
  - Instructions in French

Algorithm for a very simple task

- For a student to execute.
- For a robot to execute.

Q9

Four important CS skills

- Design algorithms
- Analyze algorithms
- Evaluate algorithms
- Adapt algorithms

Human Languages vs. Programming Languages

- Ambiguous *vs.* very precise
- Syntax (form) must exactly match …
  - CaSe MAtterS
- Semantics (meaning)
- Translation
  - High-level language (Maple, Java, Python, C) to
  - Low-level language (machine language)
  - Compiler, interpreter

Python: A Programming Language!

Follow these instructions to:

1. Confirm that you have Python installed
2. Install the *zellegraphics* package

Do NOT install **Pycreate and the Bluetooth Transmitter** yet; we will do that later in today's session (or as homework, as time permits).

Key ideas from live coding session:

evaluation in the interpreter, variables (case matters!), assignment

- In the interactive Python shell (at the >>> prompt), try:
  - 3 + 4
  - 3 + 4 * 2
  - An expression that adds 3 and 4 and then multiplies the result by 2
  - width = 4
  - height = 5
  - width
  - width, height
  - width = width + 2
  - width
  - Width

The interpreter evaluates the expression that it is given and shows the result

Assignment: read it as "width GETS 4"

Terrible mathematics, but common programming paradigm: increment width by 2

Case matters. Try to decipher the error message.

Key ideas from live coding session:
defining functions, calling functions

- In the interactive Python shell (at the >>> prompt), try:
  - triangleArea = width * height / 2
  - triangleArea
  - def rectangleArea(width, height):
        return width * height

Defining a function. Note the colon and subsequent indentation.

  - rectangleArea(6, 8)

Calling a function

  - rectangleArea(9, 3)
  - width
  - triangleArea

Note the difference between *triangleArea* (a *variable*) and *rectangleArea* (a *function*).

Note that the *parameter width* in the definition of the function *rectangleArea* has nothing to do with the *variable width* defined earlier.

Key ideas from live coding session:

importing modules

- In the interactive Python shell (at the >>> prompt), try:
  - abs(-7)
  - sin(pi/3)

    You'll get an error message from the above
  - import math
  - math.sin(math.pi / 3)
  - from math import *
  - sin(pi/3)

Some functions are built-in

Some aren't. Importing module X lets you use *X.name* to refer to things defined in module X

Do you see the difference between
    import X
and
    from X import *
Use the latter with caution.

Key ideas from live coding session:

strings and comments

- In the interactive Python shell (at the >>> prompt), try:
  - "hello"
  - 'hello'
  - width + height
  - "width" + "height"
  - "width" * height
  - "width" * "height"
  - # This is a comment.
  - # It is ignored by the interpreter, but important help to human readers.

Double-quotes …

… are the same in Python as single-quotes (not typical of other languages)

Do you see the difference between variable names and string constants?

This one is cool! Can you guess what will happen? Note that *height* is NOT in quotes.

The same thing with *height* is quotes yields an error. Do you see why?

Key ideas from live coding session:
saving and running a Python script

- Do *File ~ New*, then *File ~ Save* and save the file as (say) *Session1.py*.
- Put into the file
  - *5*
- Then run the file by *Run ~ Run Module* (or just F5 if you prefer). Nothing happens. Then add
  - print *5*

  to the file and run the file again. Also try both of the above in the interpreter shell.
- Now add to the file
  - print width

  and run again. Note the error message and where it appears.

Do you see the difference between evaluating in the interactive Python Shell and running a script?

And how print relates to that?

And where error messages appear when you run a script?

Key ideas from live coding session:

zellegraphics! Constructing and using objects!

- Put the following into your file (erasing what was there). As you type each line, run the file and see what results.

```
from zellegraphics import *

win = GraphWin('Our First Graphics Demo', 700, 500)
line = Line(Point(20, 30), Point(300, 490))
line.draw(win)
thickLine = Line(Point(30, 490), Point(200, 30))
thickLine.setWidth(5)
thickLine.setOutline('red')
thickLine.draw(win)
circle = Circle(Point(500, 100), 70)
circle.setFill('blue')
circle.draw(win)
```

Constructs a GraphWin and makes the variable *win* refer to it

Constructs Point objects, then a Line object from them

As you type this, pause after typing the dot. Cool, huh?

Changes the characteristics of the Line to which *thickLine* refers

Add more stuff to your drawing. Experiment!

Key ideas from live coding session:

Loops! and *range!*

- Back in the interpreter (at the >>> prompt), try:
  - range(12)
  - range(2, 12)
  - range(2, 12, 3)
  - for k in range(6):
    print k, k * k

Note that this yields 0 to 11 (not 12)

Note the colon and subsequent indentation

Your turn: Write a *for* loop that prints:

0, 8
1, 7
2, 6
3, 5
4, 4
5, 3
6, 2
7, 1

Key ideas from live coding session:

Loops and zellegraphics => animation!

- Back in your Session1.py file, add:
  - for k in range(7):

    circle = Circle(Point(50, 50), k * 8)

    circle.draw(win)

Again note the colon and subsequent indentation

Cool, yes?!

- Then add:
  - rectangle = Rectangle(Point(350, 450), Point(400, 500))

    rectangle.setFill('green')

    rectangle.draw(win)

    import time

    for i in range(300):

    rectangle.move(-1, -1)

    time.sleep(0.01)

```
for k in range(7):
    circle = Circle(Point(50, 50), k * 8)
    circle.draw(win)
```

```
rectangle = Rectangle(Point(350, 450), Point(400, 500))
rectangle.setFill('green')
rectangle.draw(win)
import time
for i in range(300):
    rectangle.move(-1, -1)
    time.sleep(0.01)
```

Better style: put this line at the beginning of your file

Animation! Questions?

You'll need to figure out how to "un-draw" a graphical object.
Remember that typing a dot and pausing gives help!

Q10-12

Begin the programming problem in Homework 1, as follows:

- Create a new file called homework1.py
  - Please name it *exactly* like that – all lower case, no spaces, ends in .py
- Your file should implement a Python program that creates a graphical scene. Your scene must include some animation, via a loop.
  - Be creative and have some fun with this!
- The first lines of the file ***must*** be:
  - A comment with your name, followed by:
  - A comment that is a 1-sentence description of your scene.
- Ask questions as needed!