As you arrive

- Start up your computer and plug it in.
- Find the course web site, by visiting:
  - `www.rose-hulman.edu/class`
  - Then `csse`
  - Then `csse120`
  - Then `201110` for Delvin's sections
    `201110robotics` for David's sections
- *Bookmark that course web site*

CSSE 120 DAY 1

Introduction to Software Development - Robotics

Outline

- Introductions: students and instructor
- Administrative details, tour of web resources
- Course background:
  - What is computer science? Software development?
- Hands-on introduction to Python
  - Including *zellegraphics*
  - Today in the IDLE interactive shell, next session in Eclipse

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

Administrivia – Syllabus

- **Course web site** (bookmark it now):

  www.rose-hulman.edu/class/csse/csse120, then:

201110	for Delvin's sections
201110robotics	for David's sections

- **Syllabus** – find it now (from course web site)
  - Student assistants in F-217
    - Sunday through Thursday evenings 7 p.m. to 11 p.m.

    Weekdays 7th to 9th periods
  - Email to

    `csse120-staff@rose-hulman.edu`
  - Grading plan, attendance policy
  - Late work policy, honesty policy

No background in programming or robotics is assumed.

Consider routinely doing your homework in F-217 evenings.

Q2-3

Administrivia – Schedule Page

- ***Course Schedule*** – find it now (from course web site)
  - ***Homework 1*** due ***at start of next class***
    - Reading and Angel quiz on it
    - Programming part
      - Turn in the programming part via Subclipse (details next session)
      - Homework 1 is an exception: follow its instructions re Angel drop box
    - These *slides* – find them now (from Course Schedule)
  - ***Evening exams:***
    - Tuesday, September 28, 7 to 9 p.m.
    - Thursday, October 21, 7 to 9 p.m.

Exception: In the future, for HW assigned Monday:
- reading quiz is due Tuesday
- rest is not due until ***Wed. noon***

Mark your calendar!
No regular class those days.

Q4-5

Administrivia, Angel

- ***Angel ~ Lessons***
  - ***Attendance Widget***
    - Do it now, and at the beginning of each session.
  - ***Homework***
    - Where you take your Angel quizzes on the reading
    - *Always* do the Angel quiz (you can take it up to 4 times).
    - Drop Boxes when needed
      - For homework 1 and occasionally thereafter.
  - ***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

What is Computer Science (CS)?

- The work of computer scientists falls into three 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)

Q6

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

– from Wikipedia, Software Development

What is a Computer?

- Computer
  - Device for manipulating data
  - Under control of a changeable program

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 will use Python because:
    - It is powerful: powerful programming primitives and a huge set of libraries
    - It has a gentle learning curve; you will start using it today!

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.

Q7

What is an Algorithm?

- 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
- Analogy – 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

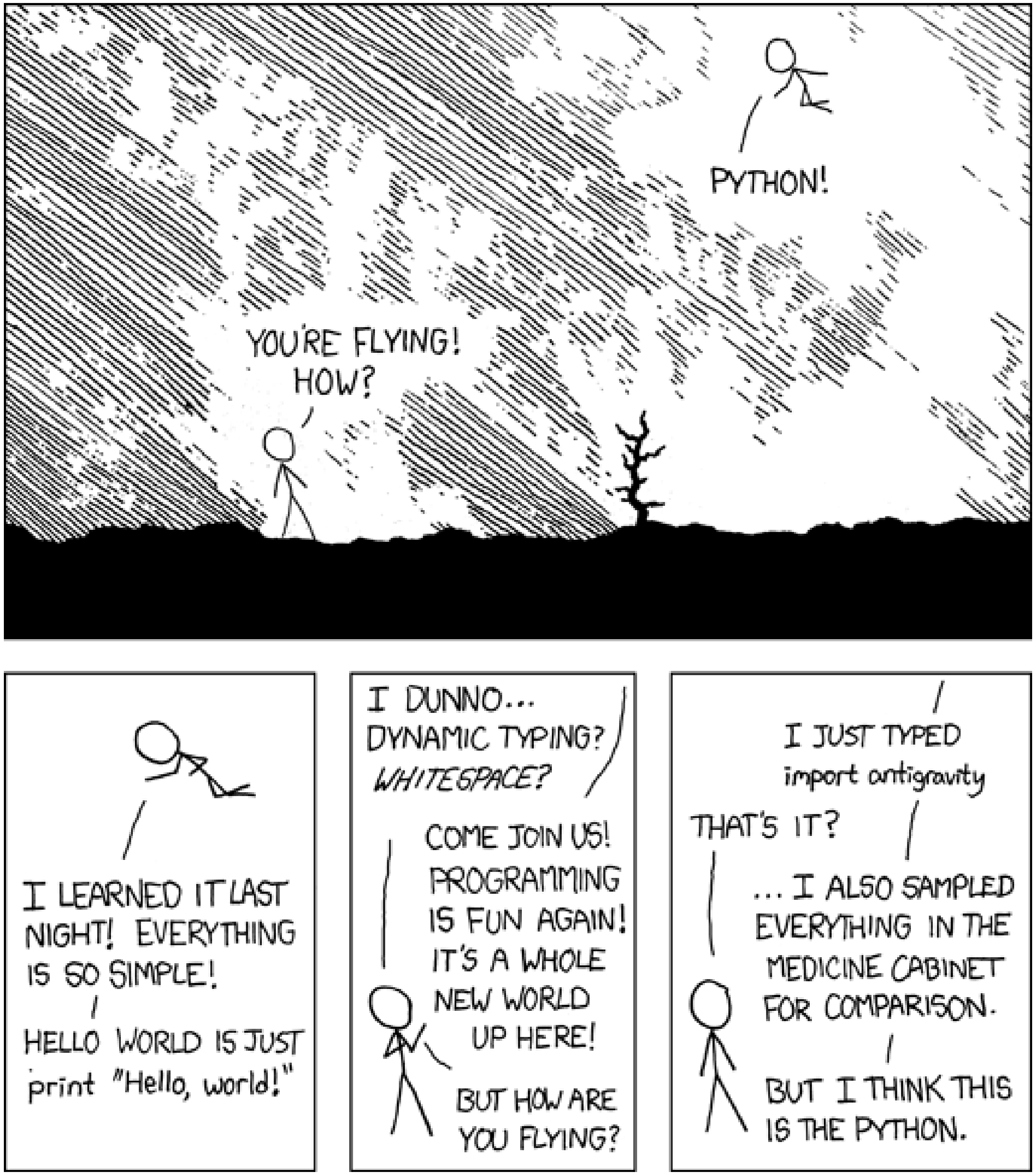
Four important Computer Science skills:

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

Q8

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!
YOU'RE FLYING! HOW?
I LEARNED IT LAST NIGHT! EVERYTHING IS SO SIMPLE!
HELLO WORLD IS JUST print "Hello, world!"
I DUNNO... DYNAMIC TYPING? WHITESPACE?
COME JOIN US! PROGRAMMING IS FUN AGAIN! IT'S A WHOLE NEW WORLD UP HERE!
BUT HOW ARE YOU FLYING?
I JUST TYPED import antigravity
THAT'S IT?
...I ALSO SAMPLED EVERYTHING IN THE MEDICINE CABINET FOR COMPARISON.
BUT I THINK THIS IS THE PYTHON.

If you had any trouble confirming that your Python 3 setup was correct (per email we sent you), or if you think that it might not be correct, ask an assistant for help now with these instructions for installing Python.

http://xkcd.com/353/

PYTHON: A PROGRAMMING LANGUAGE!

- We will see a quick view of Python programming today, but we will examine all of today's ideas in more detail in forthcoming sessions.

- Follow me as I demonstrate how to program in a Python Shell:
    - Start ➢ All programs ➢ Python 3.1 ➢ IDLE (Python GUI)
    - Make sure that when it opens, it says ***Python 3.1.2***
    - ***Follow me. You'll get a summary and transcript later.***

- ***Get an assistant to help if you have any troubles during ANY of this "live coding" session.***

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`

The interpreter evaluates the expression that it is given and shows the result. Note the use of "precedence".

  - `width = 4`
  - `height = 5`
  - `width`
  - `width, height`

Assignment: read it as "width GETS 4"

  - `width = width + 2`
  - `width`
  - `Width`

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

    ```
  - `area1 = rectangleArea(6, 8)`
  - `area2 = rectangleArea(9, 3)`
  - `area1`
  - `area2`
  - `width`
  - `triangleArea`

Defining a function. Note the colon, subsequent indentation, and blank line after the indented line(s).

*Indentation matters in Python! (**not** typical of other languages)*

Calling a function (twice in this example)

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

Note that the parameter **width** *in the definition of the function* **rectangleArea** *is completely independent of 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 is 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 module (script)

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

  to the file and run the file again. Also try both of the above in the interactive Python 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 module?

And how **`print`** *relates to that?*

And where output and error messages appear when you run a module?

Key ideas from live coding session:
zellegraphics! Constructing and using objects!

- Put the following into your *Session1.py* file (erasing what was there). Run the file and see what results.

```
    from zellegraphics import *

    win = GraphWin('Our First Graphics Demo', 700, 500)
```

Import graphics library

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

```
win.getMouse()

win.close()
```

Click to close window!

Key ideas from live coding session:

zellegraphics! Constructing and using objects!

- In your *Session1.py* file, type each line starting below declaring the win variable, then run the file and see what results.

```
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 ***`Point`*** *objects, then a* ***`Line`*** *object from them*

As you type this, ***pause after typing the dot and count to 3****. Hints for completion pop up!*

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:
  - `list(range(12))`
  - `list(range(2, 12))`
  - `list(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):

```
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)
```

Better style: put the **`import time`** *line at the beginning of your file.*
Aside: in fact, you can get away with omitting the **`import time`** *in this module, because* ***zellegraphics*** *imports it and you imported* ***zellegraphics****.*

Pauses the animation for .01 seconds.
Do you see how this loop yields an animation?

You'll need to figure out how to "un-draw" a graphical object. Remember that typing a dot after a variable that refers to a graphical object and then pausing (count to 3) gives help!

Begin the programming problem in Homework 1, as follows:

- In IDLE, 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!