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 `201220`
- ***Bookmark that course web site***

CSSE 120 DAY 1

Introduction to Software Development

Outline

- Introductions
- Administrative details
- Course background
- Hands-on introduction to Python

Roll Call & Introductions

- Name (nickname)
- Hometown
- Where you live on (or off) campus
- Something you enjoy doing or are good at

This means you should be answering Questions 1 on the quiz.

Q1

Administrivia

- Course web site
  - Bookmark it!
  - *www.rose-hulman.edu/class/csse/csse120/201220*
- Syllabus
- Schedule page
- ANGEL

How to Succeed in CSSE120

- Read
- Start early
- Work and learn with other students
- Take advantage of instructor and assistants

What is Computer Science?

- Designing and Building Software
- Developing effective ways to solve computing problems
- Devising new and better ways of using computers in areas like:
  - Robotics
  - Computer vision
  - Digital forensics
  - Scalable networks

Q6

What is Software Development?

- Researching the market
- Gathering requirements
- Analyzing the problem
- Designing a software-based solution
- Testing and implementing the software
- Maintaining the software for customers

http://en.wikipedia.org/wiki/Software_development

What is a Computer?

A device
for manipulating data
under the control of a changeable
program

What is a Program?

- Detailed, step-by-step set of instructions
- Meant to be executed by a computer

What is a Programming Language?

- A programming language specifies the:
  - Syntax (form)
  - Semantics (meaning)
- That is:
  - What we're allowed to say
  - What the computer will do in response

Why Python?

- There are thousands of computer languages!
- Python is powerful
  - Powerful programming primitives
  - Huge set of libraries
- Python has a gentle learning curve
- You'll start using it today!

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.

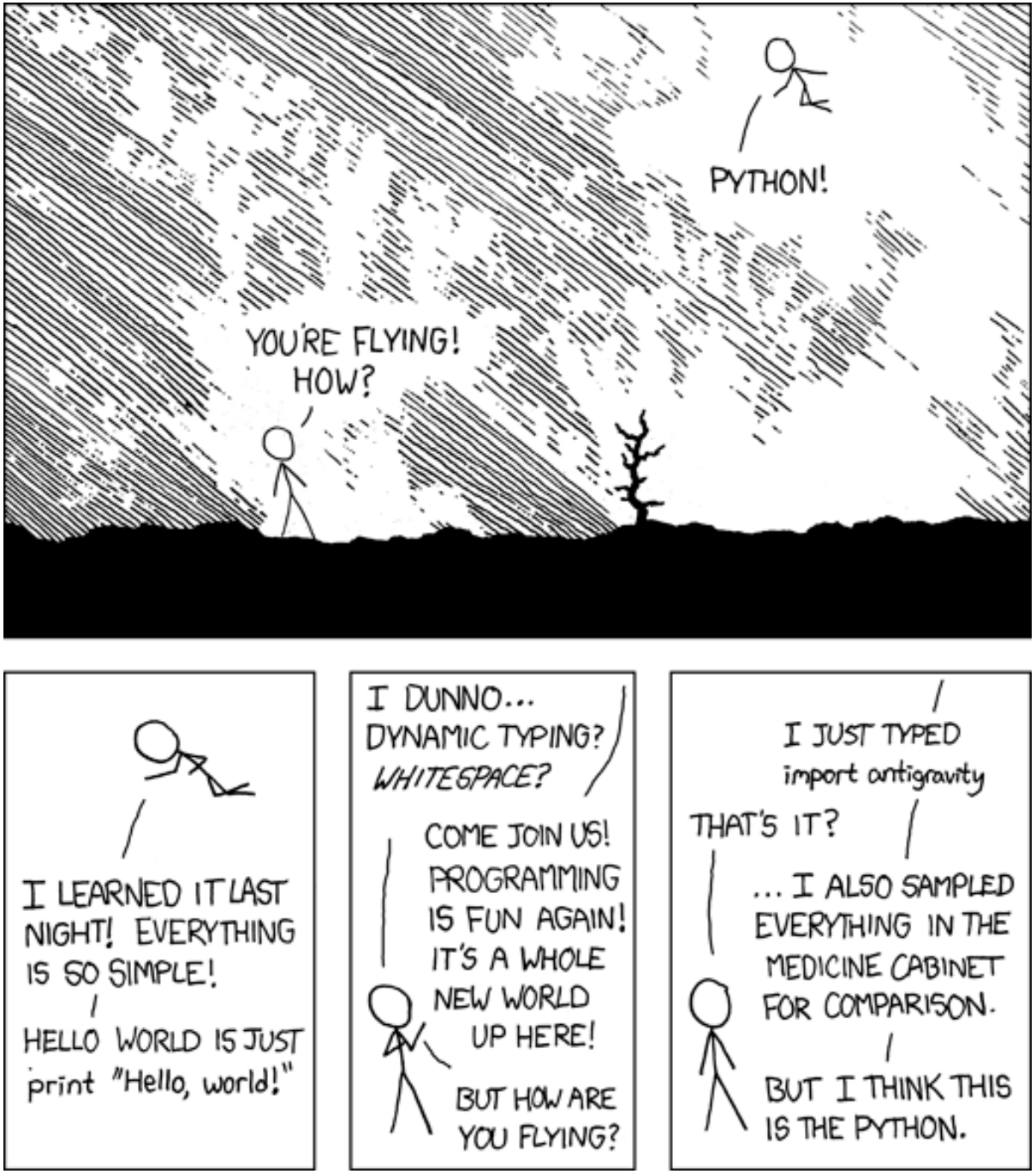
What is an Algorithm?

- What is an Algorithm?
  - Step-by-step procedure for accomplishing something
- Analogy – Bake a cake
- Example algorithm for a very simple task…

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)
  - Done by compiler or interpreter

Break



http://xkcd.com/353/

Let's Play!

- Live coding, please code along!
- Get help from an assistant if you're stuck
- Rest of the slides have examples to refer to later

Begin Homework 1 Program

- In IDLE, create a new file called *homework1.py*
  - 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.
  - 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.

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

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. Note the use of "precedence".

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

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

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

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

***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

Some functions are built-in.

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

  - `import math`
  - `math.sin(math.pi / 3)`
  - `from math import *`
  - `sin(pi/3)`

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):

```
    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!