

Introduction to CS111

Part 2: Big Ideas



CS111 Computer Programming

Department of Computer Science
Wellesley College

What is Computer Science?

- It's not really about computers.
- It's not really a science.
- It's about **imperative** (“how to”) **knowledge** as opposed to declarative (“what is”) knowledge.
- Imperative knowledge is expressed via **algorithms**: **computational recipes**.
- “A computer language ... is a novel formal medium for expressing ideas about methodology, not just a way to get a computer to perform operations. Programs are written for people to read, and only incidentally for machines to execute.”
-- *Harold Abelson and Gerald J. Sussman*

Five big ideas

- Five important concepts are at the core of this course:
 1. Abstraction
 2. Modularity
 3. Problem Solving Strategies
 4. Models
 5. Interdependence
- These interrelated ideas are important in almost every discipline, but they're at the core of CS.
- We will illustrate these ideas in several ways.
- Our goal is to help you think about problem solving in new ways.



Big #1: Abstraction



Google Search

I'm Feeling Lucky

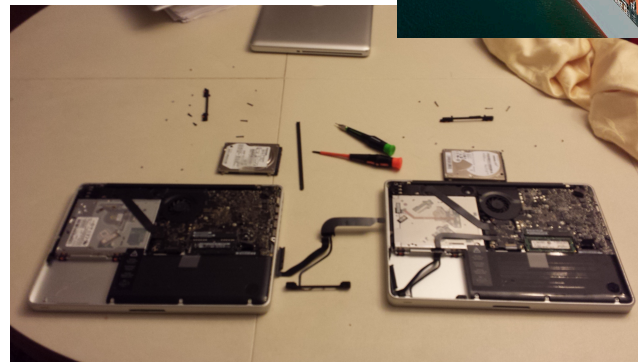
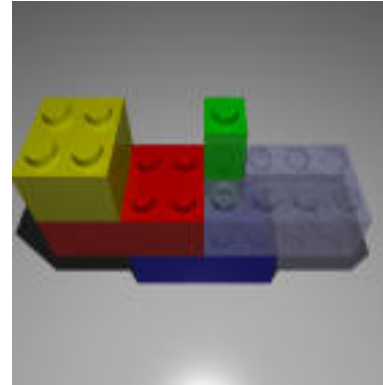
The essence of abstraction is taking complex things, hiding unimportant details, and presenting essential features to users in a simple contract.

Big



#2: Modularity

- Large systems are built from components called **modules**.
- The interfaces between modules are designed so they can be put together in a mix-and-match way.
- In computer programming, the goal is to design packages for maximum reusability.



Big #3: Problem Solving Strategies

Example: Divide/Solve/Combine

Divide

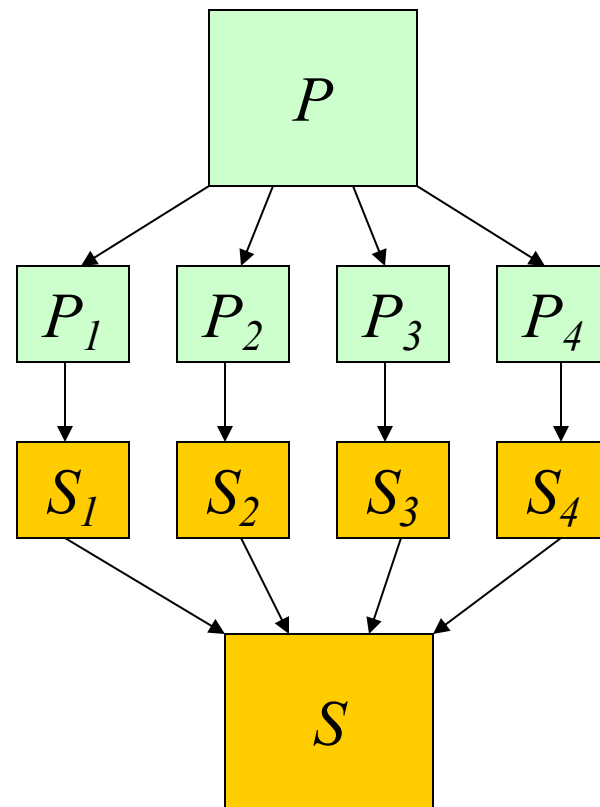
problem P into subproblems.

Solve

each of the subproblems.

Combine

the solutions to the subproblems into a solution S for P .



Other Strategies/Skills

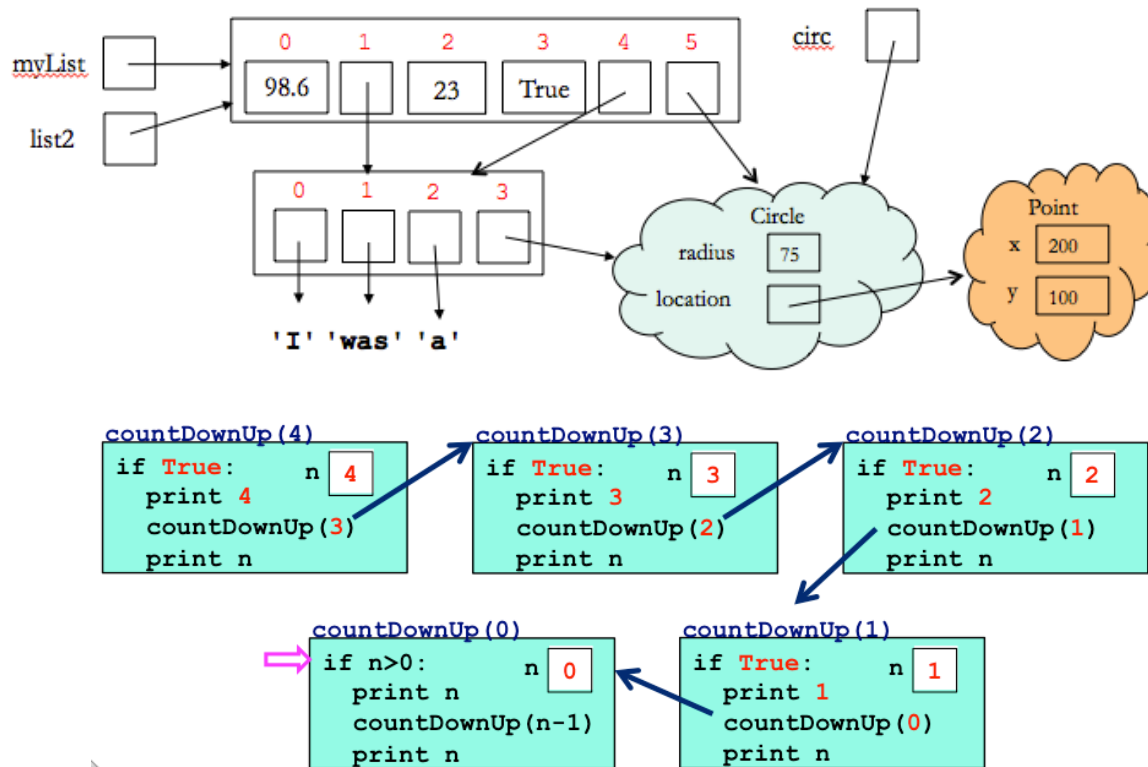
- Incremental/iterative development
- Testing & Debugging

Big



4: Models

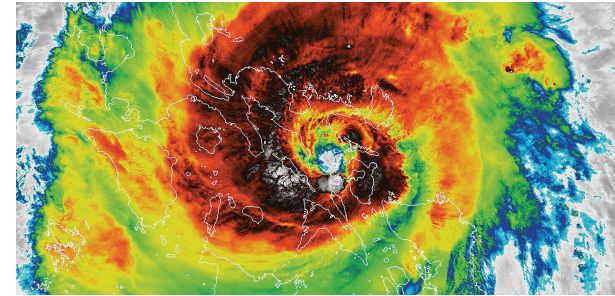
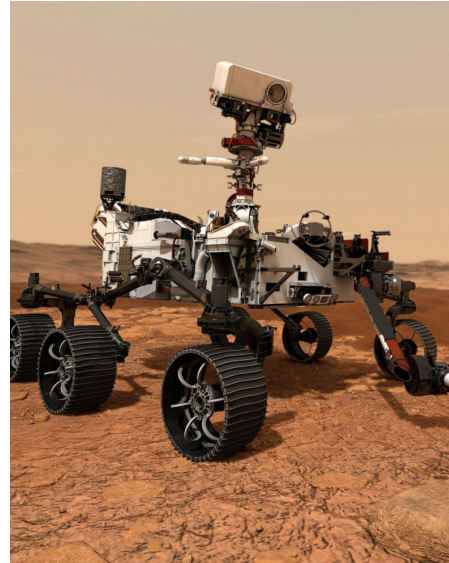
- Need simple models to understand complex artifacts and behaviors.
- We'll draw lots of diagrams to predict what programs will do.



Big #5: Interdependence



- Most of our social, economic, industrial, scientific, cultural activities are mediated via computational technologies. They have improved our quality of life.
- At the same time, companies that build these technologies are asserting too much power in our lives, and authoritarian regimes use them for surveillance and oppression.
- A liberal arts education helps you ask the right questions about computational technology.



What will you build with these ideas?

We'll start with numerical calculations and graphics

What is your name? **Valentina**

How many classes are you taking this semester? **5**

What is the average time in class per week this semester? **2.5**

How many hours per week do you spend on extracurricular activities (including jobs)? **15**

How many hours per day do you sleep on average? **8**

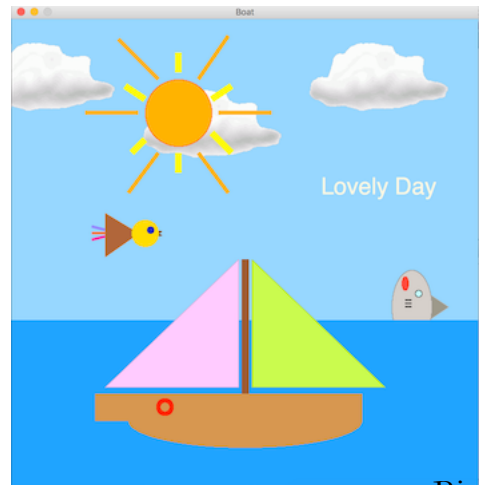
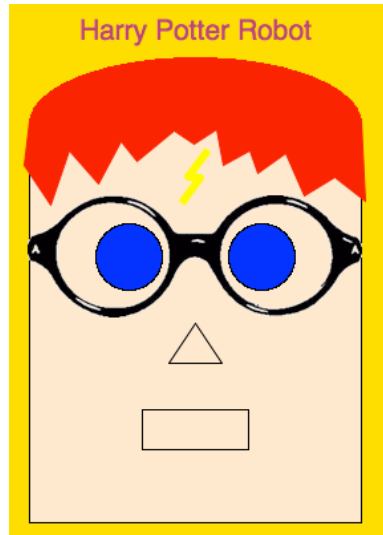
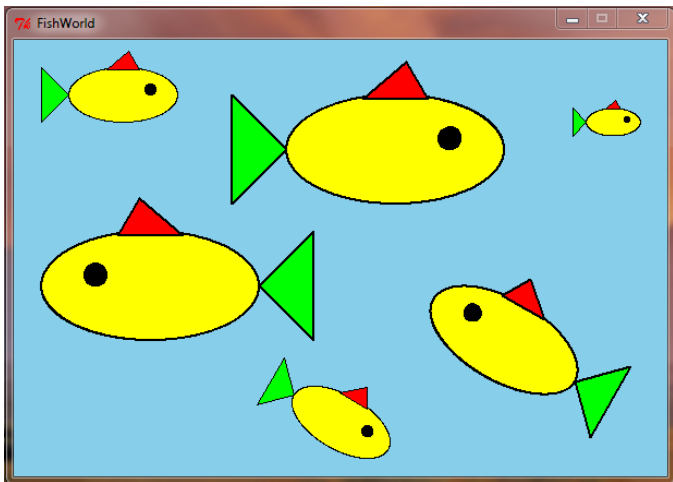
Weekly time profile for Valentina:

37.5 class hours: CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

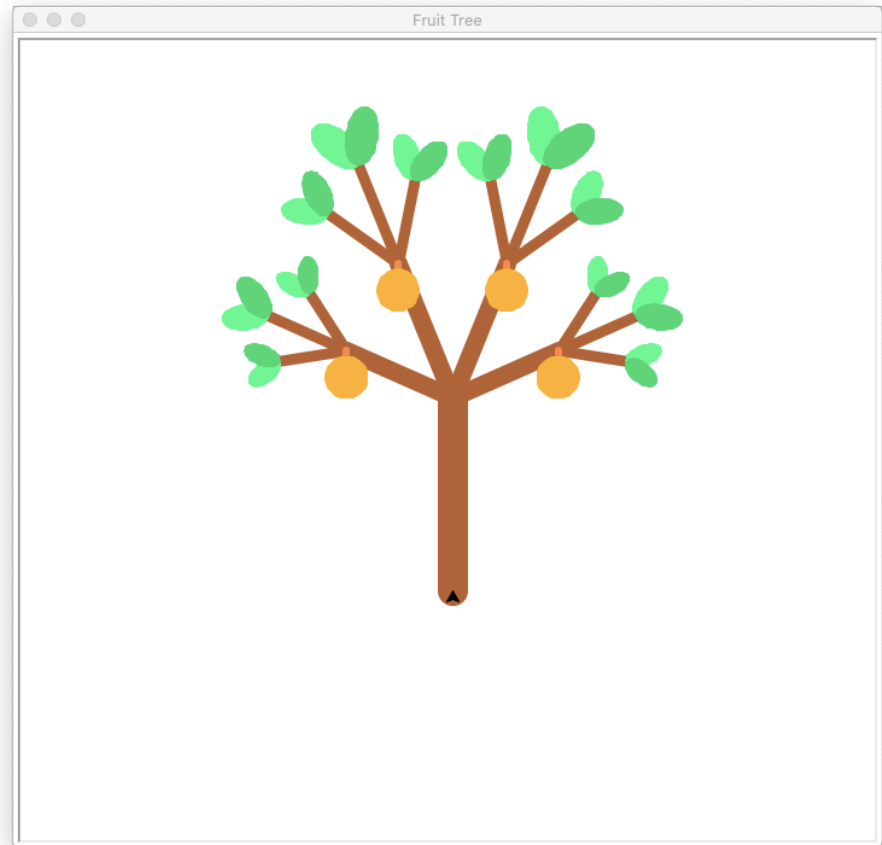
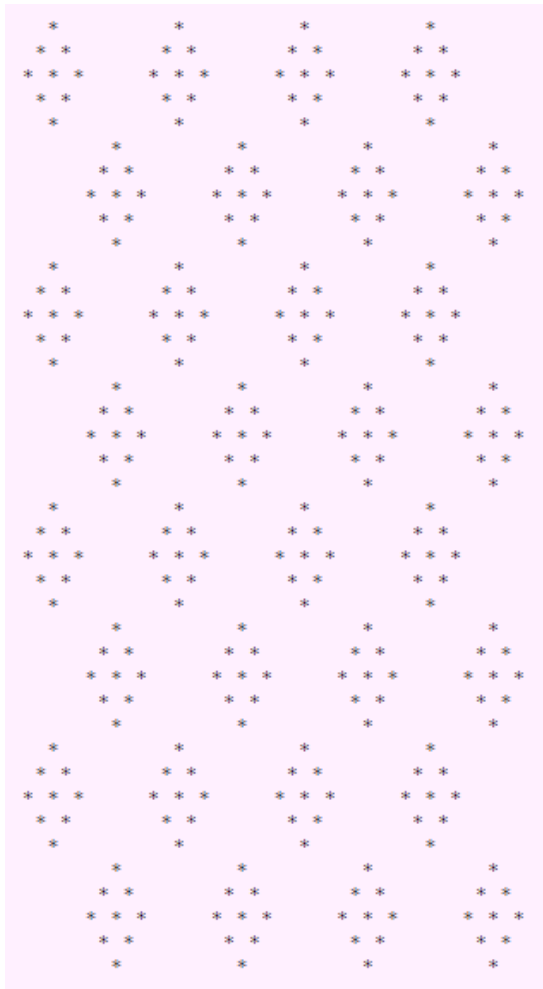
15.0 extracurricular hours: XXXXXXXXXXXXXXXXX

59.5 free hours: FF

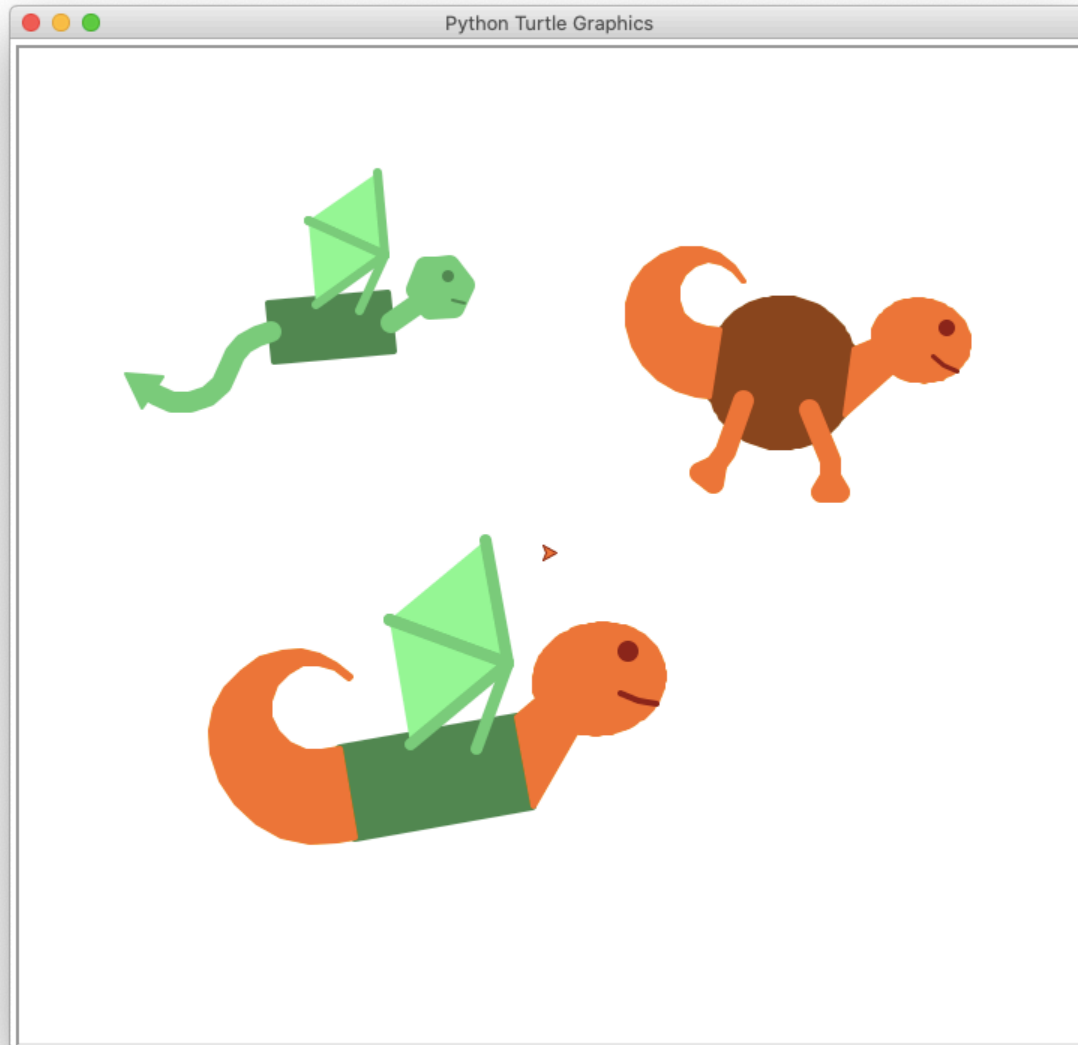
56.0 sleep hours: SSS



Define functions to capture common patterns



Apply different problem solving strategies



Build games and solve mazes with conditionals

Playing rock-paper-scissors (to 3 wins)

Each round, pick a gesture from 'rock', 'paper', or 'scissors'.

Round 1 (0 wins, 0 losses, and 0 ties)

What gesture will you use? rock

Computer chooses scissors

You win!

Round 2 (1 win, 0 losses, and 0 ties)

What gesture will you use? rock

Computer chooses paper

Opponent wins!

Round 3 (1 win, 1 loss, and 0 ties)

What gesture will you use? paper

Computer chooses rock

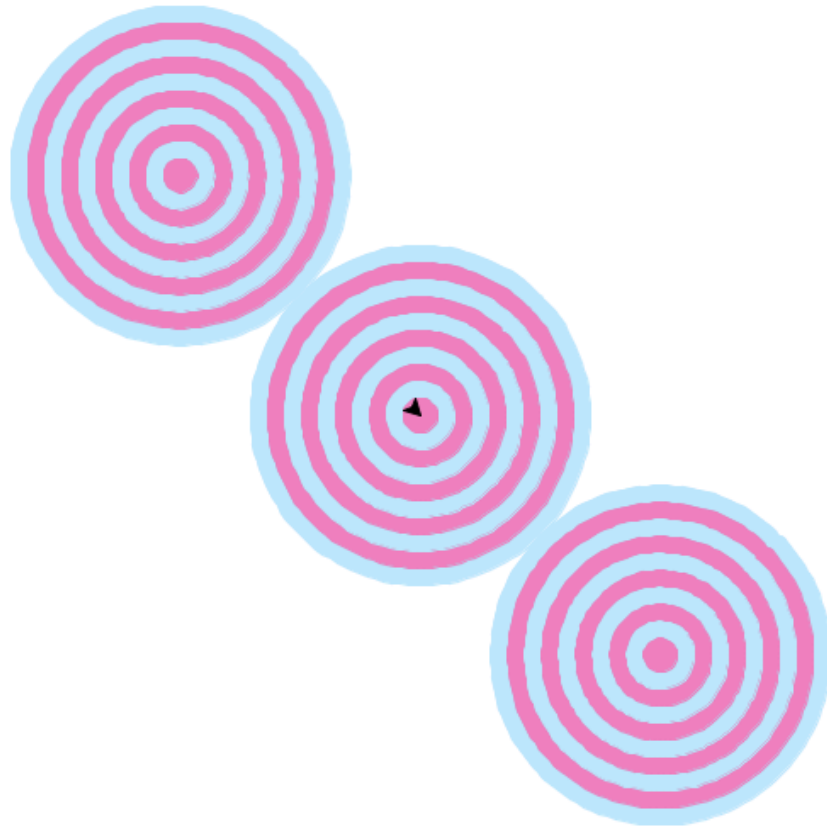
You win!

Step #0



Fuel remaining: 40

Use the power of iteration to generate pictures or music and analyze poetry



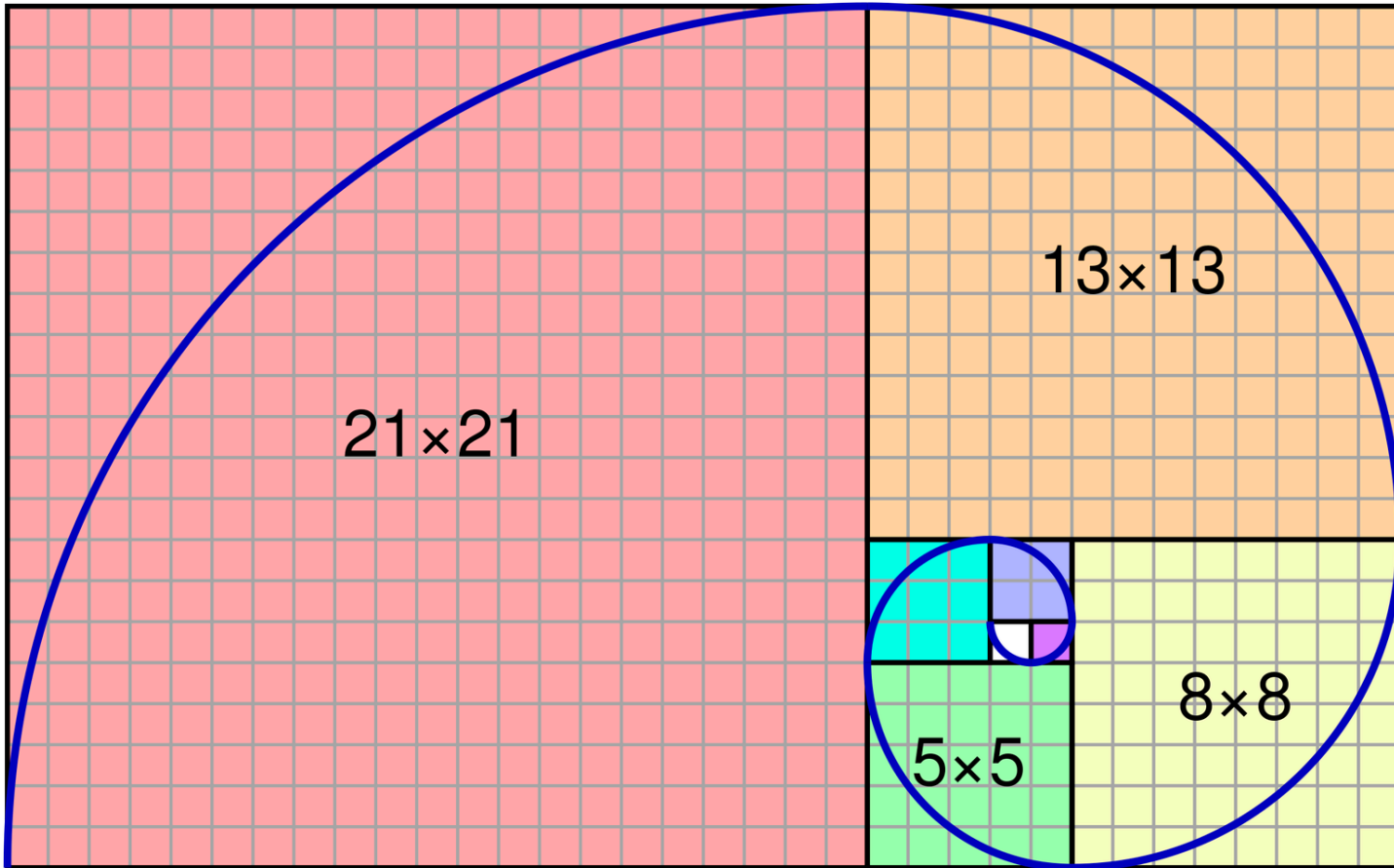
in summer rains
the crane's legs
become short

4-4-3

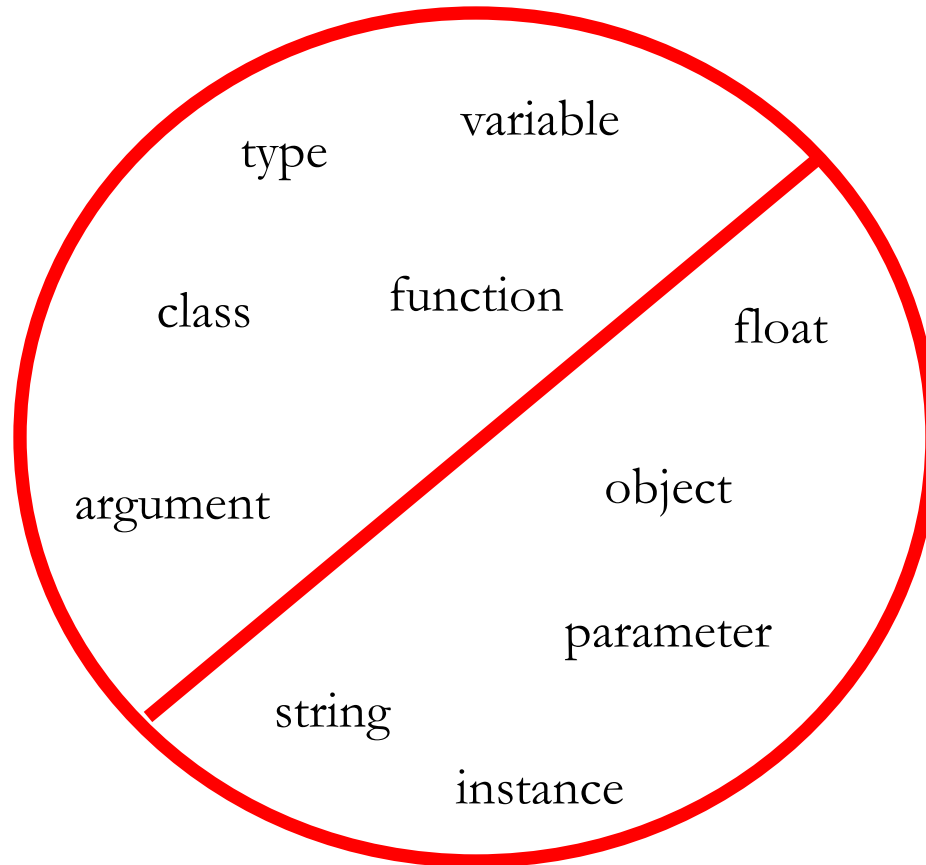
Learn Data Structures: Lists and Dictionaries

```
miniDict = {  
  "PMH4oUa-bWELKogdtkWewg": {'state': 'ON', 'address': '100  
City Centre Dr', 'review_count': 16, 'stars': 2.0,  
'name': 'GoodLife Fitness', 'city': 'Mississauga',  
'categories': ['Fitness & Instruction', 'Sports Clubs',  
'Gyms', 'Trainers', 'Active Life']},  
  "XguKrY0dAuaK1W6HULUQ1Q": {'state': 'OH', 'address': '547  
Sackett Ave', 'review_count': 29, 'stars': 3.5, 'name':  
"Retz's Laconi's II", 'city': 'Cuyahoga Falls',  
'categories': ['Italian', 'Restaurants', 'Pizza']},  
  "Wpt0sFHcPtV5M09He7yMKQ": {'state': 'NV', 'address':  
'3020 E Desert Inn Rd', 'review_count': 20, 'stars': 2.0,  
'name': "McDonald's", 'city': 'Las Vegas', 'categories':  
['Restaurants', 'Fast Food', 'Burgers']},
```

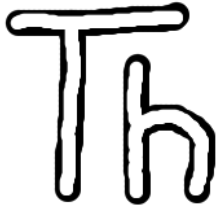
Problem solving with Recursion



On to Python! Unlearn what you have learned



I don't think that word means what you think it means



Thonny to edit and execute Python programs in projects

```
Thonny - /Users/andrewdavis/Documents/CS/Wellesley/CS111/cs111-site/cont...
turtleBeads.py x
1 """
2 turtleBeads.py
3
4 Turtle graphics library for drawing various shapes centered on
5 cursor.
6
7 In general, these functions draw things centered at the cursor,
8 the cursor back where it started afterwards. Set the pensize an
9 before drawing a shape to control what is drawn. For most shape
10 also use fillcolor and begin_fill/end_fill to fill in the shape
11 """
12
13 import math
14
15 from turtle import *
16
17 # Setup function
18 #-----
19
20 def setupTurtle():
21     """
22     Sets up the turtle window using default size, speed, pen si
23     pen/fill colors.
24     """
25     try:
26         setup()
```



Jupyter notebooks for hands-on lecture activities and weekly graded drill-like exercises

lecture-02 Last Checkpoint: Last Wednesday at 3:23 PM (unsaved changes)

File Edit View Insert Cell Kernel Help

Code CellToolbar

CS111 Lecture 2: Introduction to the Python Language

Viewing Advice: If you don't see a menu bar at the top of this document, navigate to Canopy | Preferences | Python | PyLab backend and set it to *Inline(SVG)*.

This notebook accompanies the lecture notes of [Lecture 2: Introduction to the Python Language](#).

The following examples will familiarize you with the Python programming language. The code is provided in the **input cells** (notice the labels In []:). To run the code in a cell, select it (by putting the cursor in the cell) and then click the Run button. (it looks like the Play in a Music Player interface). Alternatively, press Shift+Return in your keyboard. You'll see the result in the Out []: cells. You can rerun the code in a cell at any time. Feel free to change the code to experiment.

Simple Expressions: Python as a Calculator [Slide 2-4]

The Python interactive interpreter can perform calculations of different expressions just like a calculator. **Try to guess the result of each input, and then run the code to see the result.** The phrases preceded by # are comments, they are ignored during the code execution.

```
In [ ]: 3 + 4 * 5 # precedence
```

```
In [ ]: (3 + 4) * 5 # override precedence
```

Python Intro Overview [Continues next lecture]

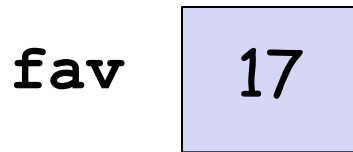
- **Values:** `10` (integer),
`3.1415` (decimal number or float),
`'wellesley'` (text or string)

- **Types:** numbers and text: `int`, `float`, `str`
`type(10)`
`type('wellesley')`

Knowing the **type** of a **value** allows us to choose the right **operator** when creating **expressions**.

- **Operators:** `+` `-` `*` `/` `%` `=`
- **Expressions:** (they always produce a value as a result)
`len('abc') * 'abc' + 'def'`
- **Built-in functions:** `max`, `min`, `len`, `int`, `float`,
`str`, `round`, `print`, `input`

The First Model: Variable as a Box

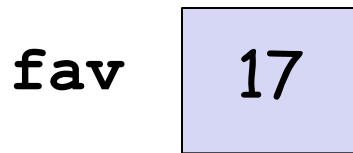


```
fav = 17 # assign
```

```
# lookup and reassign  
fav = fav + 3
```

- Variables are names we make up (but, there are rules for creating these names)
- They represent a piece of data and can take on many forms (numbers, words, a data structure containing multiple pieces of data, etc).
- A variable name should appear for the first time in an **assignment statement**.

The First Model: Variable as a Box



- A value is stored in a “box”.
- The variable “labels” the box.
- When a variable is used in expressions, we lookup for the “box” with that name and read its value.
- We can reassign a (new) value to a box.
- If we use a name in an expression without using it in an assignment first, we get a **NameError**.

```
fav = 17 # assign
```

```
fav + 3 # lookup
```

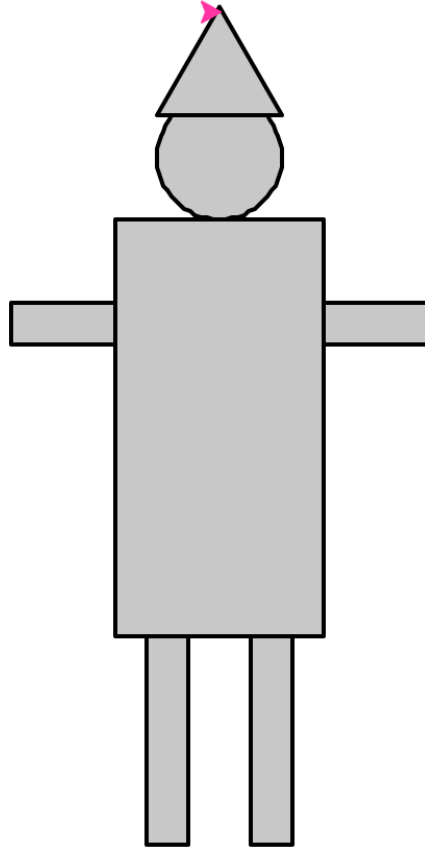
```
# lookup and reassign
```

```
fav = fav + 3
```

Example with turtle graphics

lec_course_intro.ipynb

If only I had a heart!



turtle – A library for turtle graphics

```
from turtle import *  
from turtleBeads import *
```

Defines names used in this program and provides support for graphics

```
setupTurtle()
```

Setups up the turtle and screen for drawing

```
# the head  
teleport(0, 130)  
fillcolor('grey')  
begin_fill()  
drawCircle(30)  
end_fill()
```

A series of function calls that move the turtle to the right location for drawing, setups the turtle to fill a shape with the color grey, and draws a circle.

```
# SOME CODE OMITTED
```

```
teleport(0, 200)  
color('deep pink')  
style = ('Courier', 40, 'italic')  
write('If only I had a heart!', font=style, align='center')
```

Function calls for drawing. Note the third line starting with “style” is a statement that assigns the object on the right to the variable style.