Welcome to CS61A Disc 4 Sect. 29/47 :D

Dickson Tsai
dickson.tsai+cs61a@berkeley.edu
OH: Tu, Th 4-5pm 411 Soda

Previous: Recursion, Tree Recursion **>>>**
Today: Data Abstraction >>>
Next stop: Trees

Section Quiz Logistics

- Please take out a piece of paper
- Put your name, login (e.g. cs61a-ty), SID, and section number (#47 for 11-12:30 pm, #29 for 6:30-8pm)
- Graded on effort: effort = revise your quiz with diff color

Section Quiz

1. How can I make your lab/discussion experience more worthwhile?
2. What is an object? Think of a real life example.
    1. Hint: They consist of _____ and _____, bundled together to create _____
3. An abstract data type (ADT) allows us to treat _____ as units. Think of a real life example.
4. What is data abstraction? Think of a real life example.
    1. Hint: A methodology by which functions enforce an _____ between r_____ and u___

Section Quiz [Solutions]

1. What is an object? Think of a real life example.
    1. Hint: They consist of data and behavior, bundled together to create abstractions
2. An abstract data type (ADT) allows us to treat compound objects (object that combines objects) as units. Think of a real life example.
3. What is data abstraction? Think of a real life example.
    1. Hint: A methodology by which functions enforce an abstraction barrier between representation and use

Section Quiz [Solutions]

1. What is an object? Think of a real life example.
    1. A car. Has properties (data) such as color, model, etc. Has behavior such as drive, brake, etc.
2. Abstract data type (ADT) Think of a real life example.
    1. UC Berkeley is collection of classrooms, labs, faculty, etc.
    2. 'University' – institution for teaching, research
    3. University rankings: manipulate univ. as single unit

Top Public Univ.
1. UC Berkeley
2. UCLA

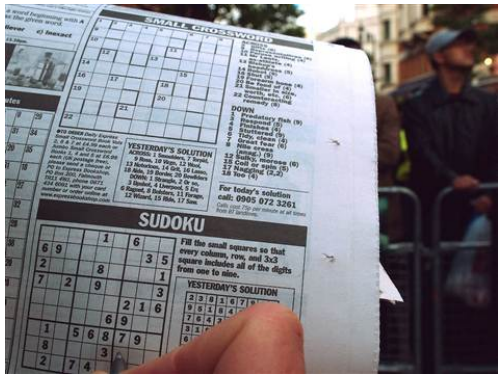
Section Quiz [Solutions]

1. What is data abstraction? Think of a real life example.
    1. Real life example:

    You can play Sudoku anywhere! How's that possible???

    - The appearance may change, but the rules never change!

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9



Anuncios

- Midterm 1 is graded. Exams require practice!
- HW3 due next Wednesday 10/1.
- Reminder: My OH are Tuesdays, Thursdays 4-5 PM in Soda 411
- Guerrilla section on recursion will be held this Saturday. Remember, it's a place where you can collaborate with people and do practice problems
- Do not post any of your code on a public-viewable platform online. (e.g. Github, Pastebin)
- Email me to schedule an appointment if you want me to go over the midterm with you personally.

Your feedback

- 5-10 minutes to talk about HW/Project/Q&A
- More time for quiz?
  - I'll make them easier/shorter b/c lots to cover
- More examples and explain them
  - I have found that helpful as well – will do more easy ones
- Discussion questions of different difficulty
  - You can work ahead! I usually skip questions.
- Hit more difficult questions
  - Will attempt 1 difficult question/disc using rules
- More analysis of writing code
- Funny videos, CS in the news, practice midterm Q, educational vids
- Allow food -> You can eat during discussion! Just clean up after yourselves.
- More fill-in-the-blank questions to start with
- More walkthrough of my approach

5-min Review Recap

- Monday Midterm: 4b Section avg. 1.311/4

```
def if_fn(cond):
  if cond:
    return lambda a, b: a
  else:
    return lambda a, b: b

def factorial(n):
  def base():
    return 1
  def rec():
    return n * factorial(n-1)

  return _______________
```

Remember that boolean operators are also operators! (e.g. >=, ==, …)

Function boxes!!!!!!!!!!

Tip of the Day

- Focus on your thought process, not the solutions!
  - When reviewing your midterm, don't just memorize the solutions. We're smart enough not to throw the same tricks at you.
  - Write as much as you can in a test!
    - Will help you remember more thoughts while you take the test.
    - Will help you determine what thought processes worked
  - If your answer is right… what worked?
  - If your answer is wrong…
    - What could you have done differently?
    - E.g. use function boxes, call expression trees next time

New Way of Running Disc

- Form groups of 2-3 as usual
- Write up your solution (function boxes + answers) on the board!
  - If you are concerned about putting about an incorrect answer, you shouldn't be. It's better to make mistakes here than on the test.
    - Plus, everyone can see common mistakes and learn together.
  - More practice with writing code
  - Chalkboards should make you feel smart.
  - Rotate people every problem so everyone gets a chance.

Data Abstraction

- The general technique of isolating the parts of a program that deal with how data are represented from the parts that deal with how data are manipulated

Representation – stored as 2x2 matrix here

Class city

- Name
- Latitude
- Longitude

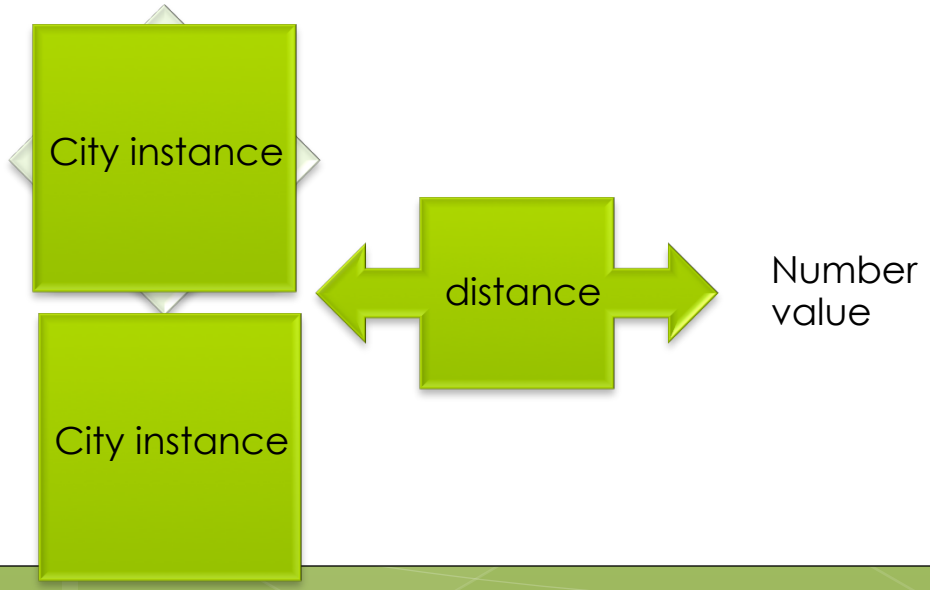
Data Abstraction

- You can do cool stuff, like define functions that take in cities, or make lists of cities, etc.
- A city is now a new type, alongside built-in types like booleans, ints, floats, strs, functions, lists, etc.

Use – don't care about how data's stored

Class city

- Name
- Latitude
- Longitude



Data Abstraction

- Another real life example (because examples help with learning).

Representation

Facebook friend

- Id1=190324823234
- Id2=435920809920
- Shared photos computed by query
- (Not the actual representation. Just trying to get message across)

Abstraction barrier: Non-disclosure agreement

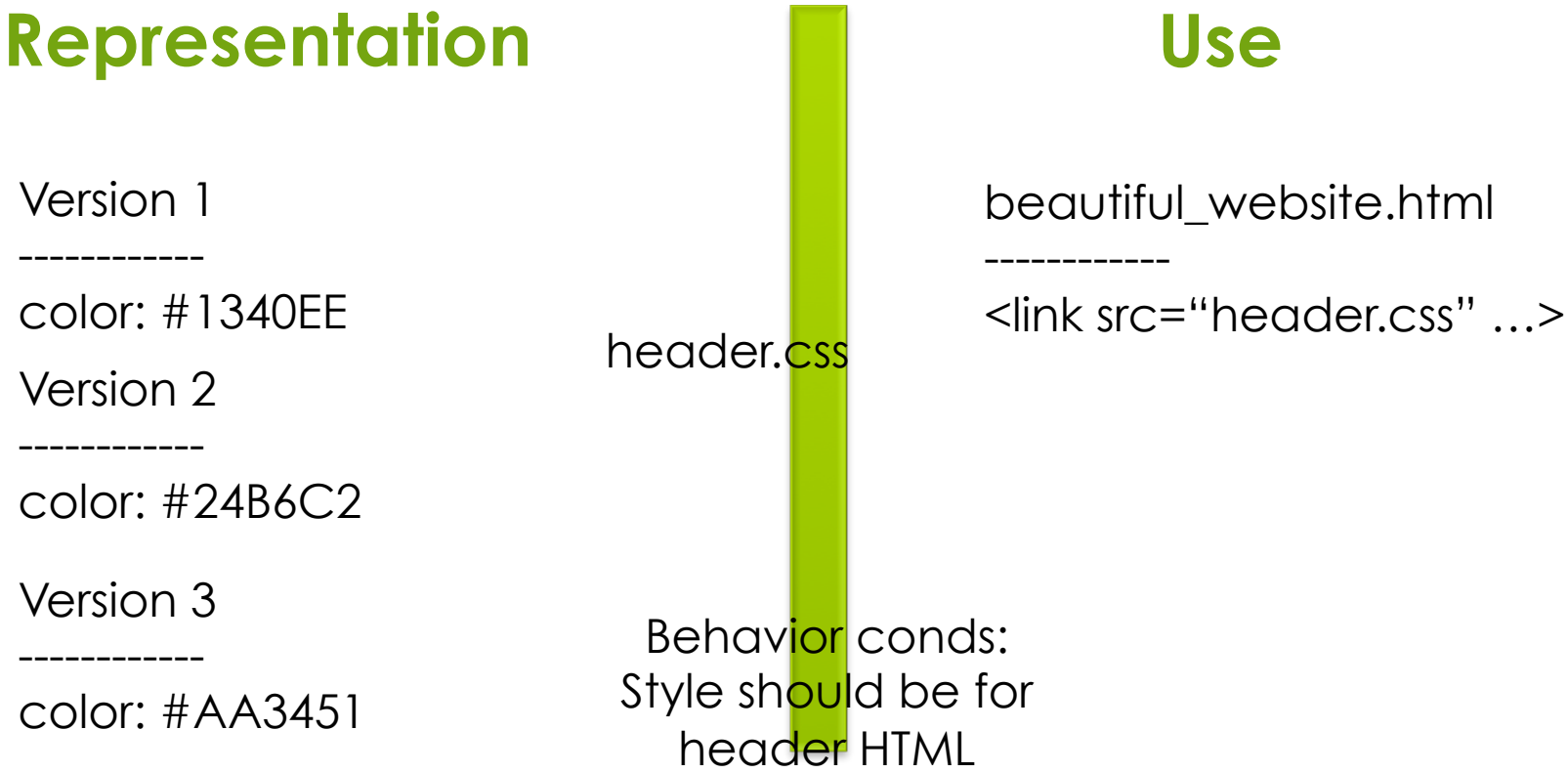
Use

Facebook friend

- Mutual friends with me
- Date we first met
- Shared tagged photos
- Behavior: Add friend
- Behavior: Delete friend

Data Abstraction – More Examples

- Another real life example (because examples help with learning): file names



Data Abstraction – Python

- Did you know? Python files in your current directory can be imported as modules!

Representation

constants.py

SLIDES = 8

constants.py

Use

life.py

```
from constants import *
print("I made {0} slides for my
        section".format(SLIDES))
print("The {0} slides are made
        with care".format(SLIDES))
```

Behavior conds:
Style should be for
header HTML

Data Abstraction – Python

- Then, for next lab:

Representation

constants.py

SLIDES = 9

Wow, I only had to change constants.py! I just use SLIDES without worrying about the actual count.

constants.py

Behavior conds: Style should be for header HTML

Use

life.py

```
from constants import *
print("I made {0} slides for my
        section".format(SLIDES))
print("The {0} slides are made
        with care".format(SLIDES))
```

Data Abstraction Qs

1. Representation: How will I store my data?
    1. Choose a data structure! They can range from a simple list to a complicated class.
    2. Not all values the user asks for has to be stored. They can sometimes be computed.
        1. def get_distance(self, city2):
2. Constructors: What's the interface available for me to make objects of a certain ADT?
3. Selectors: How to get info from an object of a certain ADT?
4. Working with the type: What functions can I use to manipulate objects of an ADT? Or should I write my own?

Data Abstraction Qs Example

Rational: numerator / denominator

1. Representation: How will I store my data?
   1. A simple list suffices: [numer, denom]. Note: order is important here!
2. Constructors: What's the interface available for me to make objects of a certain ADT?

```
def make_rational(n, d):
    return [n, d]
```

```
def make_rational(w):
    return [w, 1]
```

3. Selectors: How to get info from an object of a certain ADT?

```
def numer(r):
    return r[0]
```

```
def denom(r):
    return r[1]
```

4. Working with the type: What functions can I use to manipulate objects of an ADT? Or should I write my own?

add_rationals(r1, r2) mul_rationals(r1, r2) reciprocal(r1)

Application of Data Abstraction from my classes

- E.g. states in AI search problem
  - Each state is a combination of information (game board, where the agents are, etc.)
  - They are treated as a unit when passed into functions expecting state-types.
  - E.g. heuristic functions take in a state and returns a value for that state.
- E.g. synchronization primitives (locks, semaphores) in operating systems
  - Very low level: disabling interrupts
  - However, having to implement that all the time is tedious! We want a type of object that has that desired behavior.

Sample Data Abstraction Problems

- Given this selector function, implement the corresponding constructor of the object.
- Use this type's interface to create a function that manipulates this type.
- How do we represent the unit ADT using the Pair ADT?
  - Related question: how do we use the Pair ADT (holds 2 objects) to create a list of any length?

- There are so many questions to explore! Try working on programs to practice data abstraction!
- Reminder to finish discussion packets at home.

Optional Conceptual Questions (Wednesday 9/24 lecture)

1. What are advantages of object-oriented programming syntax?
2. What is the type of an object called?
    1. [Experienced programmers:] Can you consider the ____ an object as well?
3. What functions do we need to define an ADT and therefore create data abstraction? Con_____ and sel_____.
4. What 2 objects does the rational ADT combine?
    1. What does the con_____ look like? Give the domain and range.
    2. What do the sel_____ look like? Give the domain and range.
5. How do you describe the domain/range of add_rationals(x, y)?
6. Why is it that divide_rational = lambda x, y: [x[0]*y[0], x[1]*y[1]] violates the abstraction barrier set up by the rational constructors/selectors?
7. What are behavior conditions for ADTs? Why can (and should we) recognize ADTs by their behavior, not by their class?