

ECS 10

3/15

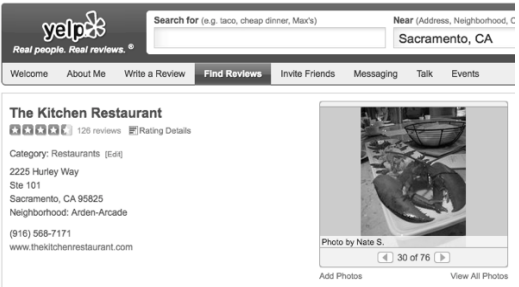
Announcements

- Course evaluations today. Need people to handle them, I am supposed to be out of the room.
- Review for final on Monday.
- Final Wds 1-3 PM. Bring a Scantron. Open notes.
- Program 6 due Sunday night.
- Practice final, extra programming problem in Resources on SmartSite.

Programming problems

- Build a program out of parts:
 - ▣ Read a file and build a dictionary, possibly combining items.
 - ▣ Read a file and build a list (or list of lists)
 - ▣ Get user input and look up item in a dictionary
 - ▣ Put dictionary items into a list
 - ▣ Sort a list
 - ▣ Write an output file, or write dictionary contents out to a file.

Yelp - find best restaurants



The input

- Input is file of 100,000 user rankings
 - ▣ Each line has restaurant, number of stars
 - ▣ Same restaurant shows up many times
 - ▣ Arbitrary order

Burgers and Brew	3
Thai Kitchen	4
Taqueria Davis	4
Burgers and Brew	5
Hunan	4
Thai Kitchen	2

The output

- Get average number of stars per restaurant
- Output list of restaurants ordered from best to worst.
- Average number of stars
 - ▣ (total stars) / (number of ratings)
 - ▣ Example: ***, **, ****, ** = (3+2+4+2) / 4 = 2.75

Burgers and Brew	4.3328
Sam's Mediteranean Cuisine	4.2876
Taqueria Davis	4.2463

Look at the output first!

- What is the output?
- What data structure produces that output?
- How can I get that data?
- Is this the most efficient way to get that output? That data?

An algorithm

- Read file, store in dictionary using restaurant name as key.
- Values are [total stars, number of ratings]
- When we get a new rating for restaurant,
 - ▣ Create new dictionary entry if necessary
 - ▣ Add to total stars, number of ratings for the restaurant
- For loop on dictionary
 - ▣ Compute average for each restaurant
 - ▣ Put [avg, restaurant name] into a list
- Sort the list

Program structure

- Four functions:
 - ▣ Loop1 (file read, build dictionary),
 - ▣ Loop 2 (for loop on dictionary, write list),
 - ▣ Loop 3 (sort list, loop to to produce output),
 - ▣ Main.

```
def main():
    rDict = makeDictionary()
    rList = makeList(rDict)
    rList.sort()
    rOutput(rList)
```

Classes

- Modules often define new kinds of objects - classes.
- If you want to understand the code in some existing module, or maybe change it, it will help to understand how classes are made.
- Let's look at some object-oriented programming from the inside...

Card class

```
class Card:
    def __init__(self, suit, num):
        self.suit = suit
        self.num = num
```

- Class code goes into block under class statement.
- `__init__` method makes a new instance
- Looks like three inputs, but really two: suit and number; here, these become attributes.

Attributes

- Begin with "self", like


```
self.suit
```
- Global within the module.
- Invisible outside it (local to the module).

Make it print out pretty

- `__str__` function determines what the class will look like when converted to string.
- Used by the print statement.

```
def __str__(self):
    return self.suit+str(self.num)
```

Deck of cards class

- Attribute will be a list of cards
- Make one card for every suit and number.

```
# factory function; always called __init__
def __init__(self):
    self.cards = []
    for num in range(1, 13):
        for suit in ["H","C","S","D"]:
            # append a card
            self.cards.append( Card(suit,num) )
```

Methods

- Other function definitions inside the class definition.
- Things you can do with a deck of cards...

```
# a method!
def shuffle(self):
    shuffle(self.cards)
```

Dealing a hand

```
def deal(self, numToDeal):
    handL = []
    for i in range(0,numToDeal):
        # pop takes the last card off the list
        card = self.cards.pop()
        handL.append(card)
    return Hand(handL)
```