Announcements

- Homework
  - Homework 7
    - Due on Friday
    - Make sure to at least be getting started on Problem 2
- Midterm 1 week from Friday
  - I'm going to try to get some study materials made up this weekend
  - I will also post last semester's test
  - No homework due next week so study up!
- Polling: `rembold-class.ddns.net`

Review Question

What is the 2nd element (index 1) of the below list?

```
[(i-1)*j for i,j in enumerate('cow')]
```

A) `''`
B) `'w'`
C) `'o'`
D) This would give an error

Solution: `''`

λ functions

- Often need a quick, one use function
- Annoying to have to define an entire function for that
- Can write anonymous functions using `lambda` expressions
  - Anatomy: `lambda <variables>: <expression>`
  - Example:

```
xs = [1,2,3,4,5]
ys = [r for r in map(lambda x: x**2, xs)]
```

Understanding Check

One of the below expressions returns something different from the others. Which is the odd one out?

A) `[x+3 for x in range(10)]`

B) `[x for x in map(lambda y: y + 3, range(10))]`

C) `[lambda x: x+3 for x in range(10)]`

D) `[y-3 for y in map(lambda x: x + 6, range(10))]`

Solution: `[lambda x: x+3 for x in range(10)]`

Define it

- A dictionary is an *unordered* set of objects
- Since unordered, we can't index them with a number
- Instead index them with keys
- Any Python dictionary is a set of key/value pairings
- Delimited with {} to denote an unordered set
- Pairings use a : between key and value

```
A = {'Jim':45, 'Bob':True, 34:'Betsy'}
```

A Key Concept

- We still index dictionaries with square brackets
- Pass the desired key inside

```
print(A['Jim'])
print(A[34])
```

- Keys need to be unique!!

Dictionary Operations

- Append:
  - Just add a new entry
  - `D['new key'] = 20`
- Combine dictionaries:
  - This extends a dictionary in place!
  - `{'A':1, 'B':2}.update({'C':3, 'D':4})`
- Remove pairs:
  - Removes a particular key/value pair
  - `del D['key to remove']`
- Loop over dictionary:
  - Loops over the keys
  - `for key in {'A':1, 'B':2, 'C':3}:`

Views

- Can also access keys or values from a dictionary with special methods:
  - Keys: `D.keys()`
  - Values: `D.values()`
- Both return a view object, which is dynamic!
  - Will change whenever the dictionary changes
- Can loop over or test if things are in view objects
  - Trying to change the size a dictionary while looping over it will result in an error

Get it?

- Commonly might want to access a key from a dictionary, but may be unsure if it exists
  - Trying to access an invalid key will give an error
- Use the `.get` dictionary method, which allows you to provide a fallback option if the desired key is not in the dictionary
  - `D.get('desired key', 'fallback value')`