- Questions

Data!

- how is the course going
- use of class time

Homework 5

in hw5_warmup.py

array indexing

- row, column
- slicing along multiple dimensions

- **broadcast assignment**

Dictionary

Bauer 2020

need a volunteer database, match names with emails

- national political movement, too many volunteers to have a separate variable for each

how might we represent this data in Python?

- dictionary (key-value pairs, keys are unique, values can be anything)

Today's mission: (aside from launching my political career) open black box—what's going on inside dict

what operations we care about?

- contains: key in dict (is this volunteer signed up?)
- write: dict[key] = value (update phone number)
- read: print(dict[key]) (display phone number)

- **mutable!**

Good rule of thumb: try the simplest thing first

- **often works just fine**
- **easier to get right**
- **good reference when attempting something more complicated**

Simplest approach

- **work with those around you**, try and sketch out how you would support those operations
- **maintain a list of tuples (key, value)**
  - append new tuples as new keys are added, replace when overwritten
- **each operation involves a search through the list**
  - i.e., we have to “lookup” the index of the key every time
- **problem: operations take number of steps proportional to size of dict**

Data structure exists that will let us look up a key in a single step no matter the size of the dict

- **hash table (also called a hash map)**
- **still have a list, key idea is we have something called a hash function**
  - function from possible keys to indices
  - list needs to be fixed size
    - we never want to generate an invalid index, a fixed size list defines a range of valid indices
  - write: hash key, write value to corresponding index
  - read: hash key, read value at corresponding index
  - contains: hash key, check if value present at corresponding index
  - **quick check**: where would these values be inserted?
    - hash function: h(s) = s[0] mod T
      - use Unicode value for character

- table size 10
- A = 65, C = 67, I = 73, Y = 89, R = 82
- Afura Jordan 111-1111, Catalina Romero, 222-2222, Issac Asimov 333-3333, Yun-En Liu 444-4444, Rahul Banerjee 555-5555

what if we have a collision?

- Take CS 201!

- **activity: hash tables provide efficient insert and delete—brainstorm what scenarios/operations**
- **Hash tables good for: inserts, deletes, contains**
- **Hash tables are bad for: minimum, maximum, next closest key, sorted order**

Practice

- ```
  s = "how now brown cow"
  char_counts = {} # creates a new dictionary
  for c in s:
      char_counts[c] += 1
  print(char_counts)
  ```
- ```
  s = "how now brown cow"
  char_counts = {} # creates a new dictionary
  for c in s:
      if c in char_counts:
          char_counts[c] += 1
      else:
          char_counts[c] = 1
  print(char_counts)
  ```
- read in a file and count how many times each word occurs

Quiz reflection

- due Monday before class, up to 2 points back per question with accurate correction and clear discussion

- Considering working with a partner on final project?