Outline for February 26, 2018

Reading: §7

1. Reading a URL [*geturl.py*,*geturl2.py*]
   a. Opening a URL
   b. Reading the page as a string
2. Dictionary
   a. Collection of key-value pairs
3. Creating dictionaries
   a. Using `d = {}`
   b. Using `d = dict()`
4. Methods for dictionaries
   a. `k in D`: True if dictionary `D` has key `k`; else False
   b. `D.keys()`: list of keys in `D`
   c. `D.values()`: list of values in `D`
   d. `D.items()`: list of tuples (key, value) in `D`
   e. `D.get(k, d)`: if key `k` in `D`, return associated value; else return `d`
   f. `del D[k]`: delete tuple with key `k` from `D`
   g. `D.clear()`: delete all entries in `D`
5. Example: memos
   a. Remember how slowly the recursive Fibonacci number program *rfib.py* ran? Here is a faster recursive version that uses memos [*rfibmemo.py*]
6. Sorting the dictionary
   a. **sorted** sorts based on keys
7. Example: word frequency count
   a. Unsorted [*wfc-1.py*]
   b. Sorted alphabetically [*wfc-2.py*]
   c. Sorted alphabetically, but dictionary order (note `key=str.lower()` in **sorted** [*wfc-2a.py*]
   d. Sorted by frequency (treat `lambda x: x[1]` as an idiom to reference the *value* of the dictionary entry, not the *key*—to go from highest to lowest, replace `x[1]` with `-x[1]`) [*wfc-3.py*]
   e. Sorted by frequency first, then alphabetically—note use of function `alphafreq(x)`; you can use any function here, and the parameter is the item [*wfc-4.py*]