CompSci 101
Fall 2021

Lecture 15

Reminders

- Assignments
  - APT 5 due

- No lab tomorrow

Key instructions

- Input ✔
- Output ✔
- Assignments* ✔
- Math/Logic ✔
- Conditionals ✔
- Repetition ✔

**not listed in book*

Python Data Types

- int, float, bool ✔
- Collections
  - Strings ✔
  - Lists ✔
  - Tuples ✔
  - Sets ✔
  - Dictionaries ⟵

PFTD

- Dictionaries cont.
  - VenmoTracker
  - Functions
- Jotto!
  - How to approach a large project
  - Splitting functionality
  - Putting it all together

Using fastcount

- Update count if we've seen word before
  - Otherwise it's the first time, occurs once

```
28  def fastcount(words):
29      d = {}
30      for w in words:
31          if w in d:
32              d[w] += 1
33          else:
34              d[w] = 1
35      return sorted(d.items())
```

Activity 1: https://bit.ly/101f21-10-28-1

VenmoTracker Revisited

- Instead of parallel lists, use a dictionary, e.g., where **`d.items()`** might be

```
[('drew', 10.0),('owen', -30.0),
 ('robert', 10.0),('susan', 10.0)]
```

- How do we extract values to return? What do we know about order in which names appear?
  - From idea to all green

Dictionary Metaphors

- Look up Duke Blue and get #001A57 or (0,26,87)
  - Key is name of color, value is hex or (r,g,b)

- Look up your address, your physical home
  - Key is address, value is the actual home

- Look up www.cs.duke.edu and get 152.3.140.1
  - Key is URL, value is IP address, browsers

Dictionary Syntax and Semantics

Syntax/Function	Meaning
`d = {}`	Initialize empty dictionary d
`d.keys()`	Collection of keys in dictionary
`d.values()`	Collection of values
`d[key]`	Value associated with key (error if key not in d)
`d.get(key,dv)`	Value associated with key (dv if key not in d, dv is optional)
`d.items()`	Collection of (key,value) tuples in d

Dictionary Iteration (unordered!)

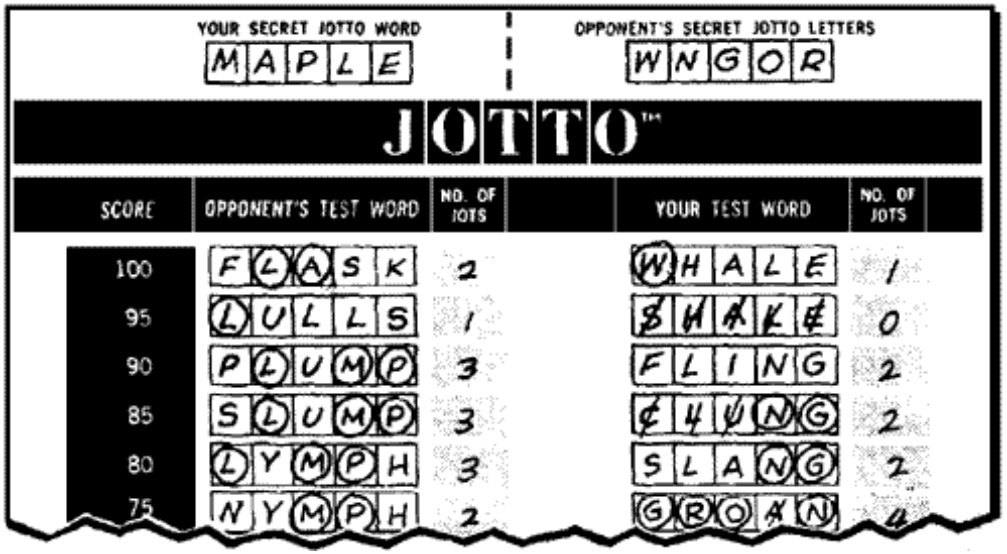
- Iterate through keys:
  - `for k in d:`
  - `for k in d.keys():`

- Iterate through pairs:
  - `for (k,v) in d.items():`
  - `for k,v in d.items():`

Activity 2: https://bit.ly/101f21-10-28-2

How to approach Hangman: Jotto

- https://en.wikipedia.org/wiki/Jotto
- http://jotto.augiehill.com/single.jsp
- Shall we play a game?



Computer Guesses Your Word

- **Brute force, no thinking or eliminating letters**
  - Pick a word at random, guess it
  - If x letters in common? Only keep words with x letters in common
  - Repeat until guessed

Approaching the Implementation

- Think Pair Share
  - Implement from empty file!
  - What is needed?
  - What order should the code do things?

Start with Blank Screen

1. Computer gets a list of words
2. Computer chooses a word at random
3. User/player enters # letters in common
4. Only keep words with that # in common

Initialization

Loop

Iterative Programming!

- Start with a task
- Implement only that task
- Write some code to check that the code works
  - Run and debug until it works
- Repeat

Should you write all the code first and then run it?

"Debugging is twice as hard as writing the code in the first place. Therefore, If you write the code as cleverly as possible, you are, by definition, not smart enough to debug it."
- Brian Kernighan (original Unix contributor)

SimpleJotto.py

- We have a file of five letter words: **`kwords5.txt`**
  - Would you like to play a game?

- Let's start! Simple version that sort of works ☺

Jotto Step 1

- Read the file kwords5.txt
  - Read the file kwords5.txt
  - Don't continue until we know this works

Jotto Step 2

- Pick a word at random
  - Pick a word at random, show the user
  - Don't continue until we know this works

Jotto Step 3

- Get number of letters in common
  - Get # letters in common, *do something?*
  - Don't continue until we know this works

Jotto Step 4

- Only keep words with that number in common

- Example: User enters 0, keep only words with 0 letters in common (replace 0 with 1, 2, …, N)
  - What are the steps here?
  - What's similar?
  - What do we do to solve this?
- Helper function: commonCount

commonCount

- Given two words, return # letters in common
- Similar to SandwichBar?
  - Sandwich ingredients -> letters in a word

Finishing SimpleJotto

- When is the game over? How to signal that
  - Interaction is via # letters in common
- Add functionality: number of guesses?
  - Remind the user where they are

Writing and Testing functions

- We'd like to test the function isolated from game
  - Ensure we don't have to play to test it
  - Unit testing similar to APT tests

- Can do this in your main!
  - Remove for final submission for hand grading
- Also use the APT testing framework and Gradescope

Summary: Jotto

- Break down entire game into steps
- Recognize what steps belong where
  - Initialization: Before loop
  - Inside loop
  - Anything after loop? End of game stuff
- Code one step at a time (or one function)
  - Test if that step worked (or submit to Gradescope)

Assignment 4: Hangman

- We give you most of the functions to implement
  - Partially for testing, partially for guiding you
- But still more open ended than prior assignments
- If the doc does not tell you what to do:
  - Your chance to decide on your own!
    - Okay to get it wrong on the first try
  - Discuss with TAs and friends, brainstorm!
- Remember: sorted(…) – cover next lecture

Reminders

- Work smarter, not harder
- Design first
- Try to identify where you are stuck
  - Identify resources to help solve problem
- Leverage your design and PythonTutor to understand program flow of control
  - http://pythontutor.com