Compsci 101
Clever Hangman, Exam, Modules
Live Lecture

‘tri_' : ['trio'] # length 1
‘trim’ : ['trim'] # length 1
s is for ...

• Software
  • Joy, sorrow, fun, changing the world

• System and sys
  • Connecting to the machine at different levels

• Sorting
  • From hat to tim to more
Announcements

• APT-6 due TODAY
• APT-7 out TODAY! Due April 8

• Assignment 5 Clever Hangman due Tues. April 6

• Lab 9 Friday
  • There is a prelab!

• Exam 3 April 13
  • Old exams on old tests page
More Announcements

• APT Quiz 2 is April 8-11
  • Topics through today
  • Topics through APT-6
Dr. Tessa Lau

• Founder, Dusty Robotics
• M.S./Ph.D., CS
  • University of Washington
• B.A./B.S., CS, Applied & Engineering Physics
  • Cornell
PFTD

• Finish Clever Hangman details

• Modules: reducing program complexity, re-use
  • pathlib library for reading files and folders
  • Using import to develop your own modules
CH Review: there will be letters

- The letter “u” has been guessed and is the 2nd letter
  Ex: _ u _ _ _ _ and user guesses ‘r’

- ["ruddy", "rummy", "rungs", … "rusty"]
  - 5 words start with “ru” and no other “r” or “u”

- ["burch", "burly", "burns", … "turns"]
  - 17 words only ‘u’ as second letter and only ‘r’ third letter

- ["bucks", "bucky", … "tufts"]
  - 98 words with only “u” second letter and no ‘r’

- What should our secret word be? "ruddy", "burch" or "bucks"?
Greedy Algorithms

- “Choosing largest group” -> greedy algorithm
  - Make a locally optimal decision that works in the long run
  - Choose largest group to make game last …

- Greed as in “it chooses the best current choice every time, which results in getting the best overall result”

- Canonical example? Change with coins
  - Minimize # coins given for change: 57 cents
Making change for 57 cents

- When choose next coin, always pick biggest
- With half-dollar coins

- With quarters and no half dollars
When greedy doesn't work

• What if no nickels? Making change for 31 cents:

• Can we do better? Yes!
Want to frustrate your friends? Use these techniques to pick a good hangman word or just pick from the list of words that have proved to be the most challenging to guess.

Hard Hangman Words:

- abruptly
- askew
- azure
- bayou
- blizzard
- absurd
- avenue
- bagpipes
- beekeeper
- boggle
- abyss
- awkward
- bandwagon
- bikini
- bookworm
- affix
- axiom
- banjo
- blitz
- boxcar
Clever vs Plain Hangman

- **Minor changes, though they require coding**
  - Regular: show ‘a e t w’
  - Clever: show ‘bcd fghijklmnopqrs uv xyz’
  - User inputs added – debug mode, length of word
  - processUserGuessClever

- **Major changes**
  - Debug mode
  - List of potential words changes at each turn
  - Function `getNewWordList` and `createTemplate`
Testing your code

• Alternative to lowerwords.txt? This is large!
  • Create your own file of words. Small file
  • Facilitates testing

• Call random.seed(…)
  • If seeded with same number
  • Same words/order every time you play
  • Reproduce errors more easily
Testing your methods

• CheckMyFunctions.py
  • Can you add anything to check/test things?

• Testing `getNewWordlist(guess, letter, words)`
  • From watching the debug game play?
  • Better: Test in isolation from game

• `getNewWordList` calls `createTemplate(template, word, letter)`
  • How we test one without the other?
  • Test `createTemplate` function first and separately
Running program again

- Create a small file for testing
Edge Case

- Words left: ['trim', 'trio']
- Hangman template is: 'tri_
- Users guesses 'm'
  - What should the secret word be? 'trio'!
  - But the dictionary has a tie!
    - 'tri_': ['trio'] # length 1
    - 'trim': ['trim'] # length 1
- `getNewWordList` should take this into account
  - Pick the template with most ' _'
Why use modules?

• Easier to organize code

• Easier to reuse code

• Easier to change code
  • As long as the “what” is the same, the “how” can change
    • Ex: sorted(...), one function many sorting algorithms
WOTO-2 Party Planning