Announcements

• No reading for next class, No RQ
• Assignment 7 due Thursday
• APT 9 out, due next Tuesday
• Do not discuss exam until it is handed back.

Snarky Hangman

• Dictionary of categories
• Start with list of words of correct size
• Repeat
  – User picks a letter
  – Make dictionary of categories based on letter
  – New list of words is largest category
    • Matched letters
    • Letters guessed by not chosen
    • List shrinks in size each time

Regular Expressions

• Powerful language for matching text patterns
• Part of the compiler process
  – Can write a regular expression for each type of word in a programming language
  – Example
    • Key words – if, else, elif, while
    • Integers – 456, 78, 2, -56
    • Float – 3.14, 7856.2345
    • String – ‘word’, “this is a phrase”
    • Special symbols – [ ] + %
Regular Expressions

- a - a
- a* - a repeated 0 or more times
- a+ - a repeated 1 or more times
- a? - a 0 or 1 time, so a is optional
- ^ - match at the beginning of the string
- $ - match at the end of the string
- . - matches anything
- [abc] - match a, b, or c
- [a-z] - match any character from a to z
- [^a] - match any character but a

More on regular expressions

- | - or
- \b - word boundary
- \s - whitespace character
- \d - match any digit
- When using backslashes – must use r in front of string
- r for raw string - r’a phrase’

Regular expressions with re

- import re
- re.sub(pattern, repl, str) – return string that replaces the pattern matches with repl in string str – looks from left end of string
- re.compile() – create a pattern
- re.findall()

Regular Exp – match and group

```python
phrase = "bogus 75 rodger@cs.duke.edu a test"
match = re.search(r'\[\w]+@[\w.]+', phrase)
if match:
    print match.group()

match = re.search(r'(\[\w]+)@(\[\w.]+)', phrase)
if match:
    print match.group(1)
    print match.group(2)
```
More on sort

• Import operator
  – fruit = [(“pear”,5), (“apple”,9)]
    • fruit = sorted(fruit)
    • fruit.sort() OR fruit = sorted(fruit)
  – arguments
    • key=itemgetter(0)
    • reverse=True

fruit = sorted(fruit, key=operator.itemgetter(1))
fruit.sort(key=operator.itemgetter(0), reverse=True)