Compsci 6/101: Random debugging?!#

- The joys and rewards of writing code to solve a problem
  - How do we know where to begin?
  - How do we know we’re making progress?
  - How do we know when we’re done?

- Make it run, make it right, (make it fast, small)
  - If we don’t have a program that runs, can’t make it right!
  - Where to begin? Do something relevant to the problem
  - Later you’ll learn more about understanding design

- Once the program is running, how to fix mistakes?

Compsci 6/101: Coding in Context

- Understand design and debugging in context
  - How do we combine all the totem functions?
  - How do we generate ‘random’ totem heads?

- More context:
  - How do we read a .csv file to find information about tracks in our iTunes library? How do we create the file?

- Representation and programming/coding
  - How do we represent a hand, deck, card for ...
  - How do we simulate card games, e.g., to validate our software for our online site, e.g., gamblingsoftware.com

Coping with totem

- Stats for totem assignment:
  - Combined: 2,142 functions; 22,247 lines of code
  - Coping with two hairBald and three hairCrazy functions?
  - Snarf the Alltotem classwork to see:

- Python has facilities that allow code to ‘inspect’ itself
  - Reflection or introspection, knowledge useful, not required
  - Understanding the possibilities helps determine how to proceed

Coping with totem

- What’s good, bad, ugly in Alltotem.py:

```python
allfuncs = dir()
headfuncs = []
#more names elided
names = ['head',..., 'mouth', 'nose', 'chin', 'hair']
onename = ['head']
for fun in allfuncs:
  for prefix in onename:
    if fun.startswith(prefix):
      headfuncs.append(fun)
print "size =", len(headfuncs)
```
Bug and Debug

- **software 'bug'**
- **Start small**
  - Easier to cope
- **Judicious 'print'**
  - Debugger too

- **Verify the approach being taken, test small, test frequently**
  - How do you 'prove' your code works?

Random and Pseudo-Random

- **'Truly random happens in nature/in the 'wild'**
  - random.org
  - Time-gaps between keystrokes, e.g., used in PGP

- **PRNG: pseudo-random number generator**
  - Uses mathematical and/or algorithmic approaches
  - Good ones are good, bad ones are predictable

- We'll use Python's random library
  - Likely good enough, for 'real science', verify

31 U.S.C. § 5361–5367

- **US Code section 5361 aka UIGEA**
  - Unlawful Internet Gambling Enforcement Act
  - Passed in 2006, took effect June 1, 2010
  - What is legal, what is illegal, what are effects?

Statistical Analysis of Poker Hand

- **How do we represent cards? Deck? Suit? Rank?**
  - What is a card?
  - What can you do with a card?
  - How will we represent a hand?
  - Keep things simple: lists help!

- **How do we 'create' a deck**
  - Number of cards?
  - Code for creating cards?
  - Loop over suits/ranks
  - Debugging assistance!
Coping with cards

- Dealing a deck of cards in Python: Cardtester.py
  - In code below, what is a deck?
  - What is a card?
  - How do nested loops work?

```python
def getDeck():
    d = []
    for rank in range(0,13):
        for suit in range(0,4):
            d.append([rank, suit])
    return d
```

Debugging with Random Numbers

- Every time we run the program it's different!
  - We need to set the seed of the PRNG
  - Same 'randomness' each time, not so random!

- How do we know if the game is 'random'
  - Eyeball or run appropriate statistical tests

- We need a good representation of a card to help
  - What's easier to understand: (0,1) or "ace of hearts"?
  - Why do we use strings? Lists? Tuples?

Solving a problem: do we need code?

- Tubthumping,3:57,Chumbawumba
- Bubbly,3:16,Colbie Caillat
- Viva la Vida,4:01,Coldplay
- Rhapsody in Blue,16:29, Leonard Bernstein
- Brick House,3:46,The Commodores

- What's the total play-time of all tracks in itunes.csv?
  - How do you solve this problem?
  - Changes if we want only tracks less than five minutes?
  - Changes if we want to fill 70-minute play-lists?

- How do we do this in Python?
  - How do represent track-time? Is this a priority?