Compsci 6/101: I  Python

- Techniques for looping
  - Loop over sequences by sequence value
  - Loop by indexing, or by index and value: enumerate
  - While loop: as long as condition holds, e.g., game not over
- Techniques for transforming data
  - One domain leads to solutions, other much harder
  - Identify music with sound-hound/shazaam
  - Encryption: transform data to hide it, but …
  - APT AnagramFree

Loop over sequence with index

- Index useful in accessing elements in order
  - Sometimes need adjacent elements, `i-1`, `i`, and `i+1`
  - Often need both index and element, see enumerate below

```
for i,fr in enumerate(['a','b','c']):
    print i,fr
```

- No more *powerful* than looping over range, why?
  - Idiomatic programming, helps to know vocabulary
    - Syntactic sugar
  - Not necessary, use `for i in range(0,len(seq)):`

Indefinite loop: while … *interactivity*

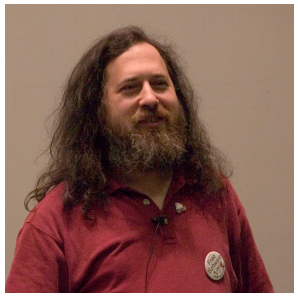
```
wrong = 0
while wrong < max_wrong:
    guess = raw_input()
    if not good_guess(guess):
        wrong += 1
    else:
        #process the guess here
```

- Suppose, for example, play http://www.hangman.no
  - What happens if you loop while True:
  - Break out of loop with break
  - See code in ***`GuessNumber.py`***

From guessing numbers to transforms

- With good-guessing, optimal number of guesses?
  - How do you reason about this?
  - Don't think of the number, but range of possibilities
- How will Watson do in Jeopardy tonight?
  - http://to.pbs.org/fRQz6p
  - How does Watson transform questions so understandable?
- Sometimes changing data leads to solution
  - Transformations depend on problem and solution space
  - If the answer is 'yes', if the answer is 'Waterloo', …

Richard Stallman (b.1953, Hopper '90)



- Transformed programming
  - Free Software Foundation
- "World's Best Programmer"
  - Gnu/Linux: g++, emacs
  - Believes all software should be free, but like "free speech", not "free beer"
  - Won MacArthur award for his efforts and contributions
  - League for Programming Freedom
    - It's about free, not open

Aside: Transform for AnagramFree APT

- How do you know when two words are anagrams?
  - Possible to tell with letter-count fingerprint
  - `"apple"` -> [1,0,0,0,1,0,0,0,0,0,0,1,0,0,0,2,0,0,0,0,0,0,0,0,0,0]
  - Can we create this fingerprint? How?
  - Alternative fingerprint: sort the letters

```
sorted("apple") is … why?
''.join(['a','b','c']) is "abc"
```

- If the data is transformed, still some work to do
  - #Anagrams in ['dgo', 'aet', 'dgo', 'aet', 'aet']?

Simulation and Randomness

- Randomness is essential for games
  - Same game every time, not so exciting
  - Where does randomness come in?
  - How does "random" happen?
- Randomness essential for modeling/simulation
  - To verify simulation, likely sensitive to probabilities
  - So-called stochastic simulations/models
  - We'll start with simple examples to investigate
    - Reinforce concepts in lab this week

Some details of module random

- Want randomness, but want to debug randomness
  - Set seed for debugging, use time or other seed
  - What's done in Las Vegas slot machines?
- See DiceRolling.py and HeadsTails.py
  - Illustrates coding styles and randomness
  - How do we count runs of heads/tails?
  - How do we verify "good" random numbers?
    - Eyeball or run appropriate statistical tests
- Toward game playing
  - Choosing words/items from list at random

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: Cardtester.py

- Dealing a deck of cards in Python: Cardtester.py
  - In code below, what is a deck?
  - What is a card?
    - What's easier to understand: [0,1] or "ace of hearts"
  - How do nested loops work?
  - Why do we use strings? Lists? Tuples?

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