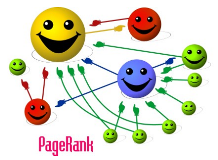
Compsci 6/101: PFTW, Feb 28-March 4

- Algorithms and Data Structures
  - Sets and how they are used in Python (data structure)
  - Algorithms, solving problems, understanding trade-offs
- Transforms and modules
  - Transforming data and then untransforming it
  - How do you send a .jpg via … email, text, copy/paste
  - How do you write programs in more than one .py file?
- Writing code and understanding "costs"
  - Cost of calling a function? It depends!

Algorithm

- What, where, when, who, why?
  - http://en.wikipedia.org/wiki/Algorithm
  - From Euclid to Google?
- Instructions, rules, list
  - task, function, …
  - effective, finite, mechanizable?



- Choose the best website for 'teaching python'
  - How does this work?
- How to search a list of strings…

Programming Equations

- Algorithms + Data Structures = Programs
  - Niklaus Wirth, old view of programming and compsci
  - Different view: functional, object-oriented, extreme/agile
- How old are algorithms?
  - Euclid: find greatest common divisor, GCD (56, 217)
  - Who cares? You do!
- A few basic idioms and algorithms go a long way
  - log one-million much less than one-million (binary search)
  - Don't do the same thing more than once

Tim Peters: Zen of Python

- Beautiful is better than ugly.
- Explicit is better than implicit.
- Simple is better than complex.
- Complex is better than complicated.
- Flat is better than nested.
- Readability counts.
- Special cases aren't special enough to break the rules.
- In the face of ambiguity, refuse the temptation to guess.

http://www.python.org/dev/peps/pep-0020/

Designing Algorithms and Programs

- Designing algorithms to be correct and efficient
  - The most important of these is ______________
  - When do we need to worry about efficiency?
  - Example: finding a number between 1 and 1000
    - High, Low, Correct: how many guesses?
    - Same algorithm can find an element in a sorted list
- Python searching in dictionary, set, list
  - How can we find an element?
  - How long does it take?
  - `if x in collection:`

Comparing Algorithms

- Searching a list of N elements for a specific value
  - Worst case is ....
- Doing binary search (guess a number), sorted list
  - Worst case is ...
- Finding the most frequently occurring element:
  - Strings? ints? does it matter? (toward Python dictionary)
- Where do proteins occur in a genome?
  - Leveraging a previously solved APT

Revisiting cgratio APT

- 'cost' of finding likely sources of protein in DNA

```
def cgratio(strand):
    cg = 0
    for nuc in strand:
        if nuc == 'c' or nuc == 'g':
            cg += 1
    return cg

def maxIndex(strand,windowSize):
    index,max = 0,0
    for i in range(0,len(strand)-windowSize+1):
        cg = cgratio(strand[i:i+windowSize])
        if cg > max:
            max,index = cg,i
    return index
```

Revisiting cgratio APT

- 'cost' of finding likely sources of protein in DNA

```
def runningMax(strand,windowSize):
    gc,counters = 0,[]
    for nuc in strand:
        counters.append(gc)
        if nuc == 'c' or nuc == 'g':
            gc += 1
    counters.append(gc)

    index,max = 0,0
    for i in range(windowSize,len(strand)+1):
        diff = counters[i] - counters[i-windowSize]
        if diff > max:
            max,index = diff,i
    return index-windowSize
```