PFTWeek 9/14-9/18

• **Incremental construction as design pattern**
  - Build programs: start small, add with confidence
  - Build new strings: append(concatenate) values
    - Also use `join` to create a string from a list
  - Build lists: append values, alter existing values
    - Also use `.split()` to create list from a string

• **Compsci 101 specifics: Python -> Course**
  - APT Quiz and ensuring you do well
Software Dreams

- Translating ideas into (Python) code
  - Create interesting “heads”, “totem poles”?
  - Create software for face recognition? Gait?
  - Create "five four" from "four five"?
  - Create "SCUBA" from "self contained underwater breathing apparatus"

- Master the syntax of the language?
  - Organization of program constructs
  - Knowledge of libraries
  - Practice and experience!
Top 10 list for surviving in CompSci 101
Top 10 list for surviving in CompSci 101

10 - Ask Questions
9 - Eat lots of pizza
8 – Learn how to spell Rodger/Astrachan
7 – Read the online textbook
6 – Do the reading quizzes
5 – Check Piazza every day
4 – Visit your professors in their office hours
3 – Learn how to debug your programs
2 – Seek help (one hour rule!)
1 – Start programming assignments early!
Why is this person so important to this course?
Why is this person so important to this course?

- Brad Miller
- Have you donated yet?
Translating Ideas Into Code

Incremental + : numbers and strings

- *Wht vvwls cn y stll rd ths sntnc?*
  - Create a no-vowel version of word
  - Examine each character, if it's not a vowel ...
  - Pattern of building a string

```python
def noVowels(word):
    ret = ""
    for ch in word:
        if not is_vowel(ch):
            ret = ret + ch
    return ret
```
Counting vowels in a string

- Accumulating a count in an int is similar to accumulating characters in a string

```python
def vowelCount(word):
    value = 0
    for ch in word:
        if is_vowel(ch):
            value = value + 1
    return value
```

- Alternative version of adding: `value += 1`
def reverse(s):
    r = ""
    for ch in s:
        r = ch + r
    return r

Create version on the right using disassembler

```
from high- to low-level Python

7      0 LOAD_CONST     1 ("")
3 STORE_FAST     1 (r)
8      6 SETUP_LOOP    24 (to 33)
9 LOAD_FAST      0 (s)
12 GET_ITER
9   13 FOR_ITER      16 (to 32)
16 STORE_FAST     2 (ch)
9      19 LOAD_FAST     2 (ch)
22 LOAD_FAST      1 (r)
25 BINARY_ADD
26 STORE_FAST     1 (r)
29 JUMP_ABSOLUTE 13
9   32 POP_BLOCK
10   33 LOAD_FAST      1 (r)
36 RETURN_VALUE
```
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?
Anatomy of a Python String

- **String is a sequence of characters**
  - Functions apply to sequences: len, slice [:], sorted,
  - Methods applied to strings, specific to strings:
    - st.split(), st.startswith(), st.strip(), st.lower(), st.find(), st.count(), st.join()

- **Strings are immutable sequences**
  - Cannot change a string, can only create new one
    - What does upper do?
  - See resources for functions/methods on strings

- **Iterable**: Can loop over it, **Indexable**: can slice it
Anatomy of a Python List

• Lists are indexable
  ➢ Start with index 0, index with [int], slice too
  ➢ Indexing past end?

• Lists are iterable: for x in [1,2,3]:
  ➢ Confusing boolean use, if 3 in [1,2,3]:

• Lists are mutable
  ➢ Change: lst[0] = 5, can append, can extend

• Lists are heterogenous, can store any type of element, including lists!

• Methods .count(), .append(), .index(), .sort()
Lynn Conway

See Wikipedia and lynnconway.com

- Joined Xerox Parc in 1973
  - Revolutionized VLSI design with Carver Mead

- Joined U. Michigan 1985
  - Professor and Dean, retired '98

- NAE '89, IEEE Pioneer '09

- Helped invent dynamic scheduling early '60s IBM

- Transgender, fired in '68
Standard accumulation idiom

```python
def wcount(collection, word):
    total = 0
    for elt in collection:
        if elt == word:
            total = total + 1
    return total
```

- How do we count 'scarlet' in *Scarlet Letter*?
  - Or dagger in *Hamlet* or *Romeo*?
  - Or friend in *Little Brother*?
  - Or CGAT in a genome?
If we knew all Python's built ins, ...

- Suppose we want to (what are types and values)

```python
f = open("/data/kjv10.txt")
st = f.read()
words = st.split()
angels = wcount(words,"angel")
# can use Python built in too
devils = words.count("devil")
```
Accumulation revisited

def getFirsts(collection, letter):
    total = []
    for elt in collection:
        if elt.startswith(letter):
            total.append(elt)
    return total

- Finding words that start with 't', The Bible?
  ➢ Or words that start with 'U' in The Illiad?
Work Together on Expression Review