PFTW: Sequences aka Strings&Lists

- From Return values to Random-ness [aka two R's]
  - What power does random provide?
  - What is a return value, different from print
  - Examples in Cityscape.py
- Loops, Lists, Strings : FileData.py
  - Loop over sequence: string, file, list, "other"
  - Process each element, sometimes selectively
  - Toward understanding the power of lists
    - List comprehensions: oh my!
- Accumulation as a coding pattern

Motivation: http://bit.ly/sportswords

- How does Google do this? Why do they do this?
  - Search through … and do what?
  - Already know the answer and display it?
- File is comprised of lines
  - Lines composed of "words"
  - Both are strings
- Breaking file into all the words
  - From string to list: both are sequences

Anatomy of a Python String

- String is a sequence of characters
  - Functions we can apply to sequences: len, slice [:], others
  - Methods applied to strings [specific to strings]
    - st.split(), st.startswith(), st.strip(), st.lower(), …
- Strings are *immutable* sequences
  - Characters are actually length-one strings
  - 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

- Create list with brackets (values optional)
  - `s1 = []`
  - `s2 = ["a", "b", "c"]`
  - `s3 = list("123") #from an iterable`
- Lists are *mutable* and *iterable*
  - Append to list, change value stored at index
  - `s2[1] = 5, s2.append(77)`

```
for elem in list:
    #process elem
```

- Functions on lists: len, min, max, sum
  - Operator: in
  - Mutators: .append(x), .extend([..]), .pop(i), …

Indexing a list

- Lists, like strings, start indexing with zero
  - Strings are immutable, lists are mutable
- For some problems, looping by index useful
  - Use range function, range creates open-ended list
  - `range(0,10), range(5,20), range(10,100,5)`
  - Advice/warning: in Python 3 range doesn't create list
- Especially and often useful for two lists
  - Parallel lists: names and GPA, movies and directors, …
  - Toward tuples [sneak preview]

Counting words: accumulation

- Anatomy of assignment and accumulation
  - var = "hello", y = 7
  - What do these do? Memory?
  - Reading assignment statement

- Accumulation

```
var = 0
for x in data:
    if x == "angel":
        var = var + 1
```

- RHS, assign to LHS

Making choices at random

- Why is making random choices useful?
  - How does modeling work? How does simulation work?
  - Random v Pseudo-random, what's used?
  - Online gambling?
- Python random module/library: import random
  - Methods we'll use: `random.random(), random.randint(a,b), random.shuffle(seq), random.choice(seq), random.sample(seq,k), random.seed(x)`
- How do we use a module?

Interlude: Cityscape.py

- How do we make a tower taller?
  - What about the spire?
  - How can we do this with a loop?
  - How can we do this at random?
  - What about making a wider base?

```
   ||
   ||
   ||
+-------+
|       |
|       |
|       |
|       |
|       |
```

- Lessons: why do functions return values
  - Can use them in many contexts, not just printing
  - Horizontal display of multiple towers?

Niklaus Wirth (Turing Award, 1984)

- Designed and implemented several programming languages including Pascal, Modula-2, Oberon
- Wrote the paper that popularized the idea of step-wise refinement
  - Iterative enhancement
  - Grow a working program
- Cranky or tasteful?

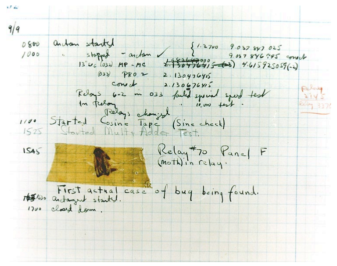
Simple, elegant solutions are more effective, but they are harder to find than complex ones, and they require more time which we too often believe to be unaffordable

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?

Bug and Debug

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



9/9
0800 arctan started
1000 " stopped - arctan ✓ {1.2700 9.037 847 025 / 9.037 846 995 correct}
13"uc (032) MP - MC ~~2.130476415~~ ~~(-3)~~ 4.615925059(-2)
(033) PRO 2 2.130476415
correct 2.130676415
Relays 6-2 in 033 failed special speed test
in Relay " 10,000 test.
Relays changed
1100 Started Cosine Tape (Sine check)
1525 Started Mult + Adder Test.
1545 Relay #70 Panel F
(moth) in relay.
First actual case of bug being found.
1630 arctangent started.
1700 closed down.
Relay 2145
Relay 3370

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

Toward a Deeper Understanding

- What is Python? What is a programming language?
  - How are programs executed? What does that mean?
  - Why do you need to have an understanding of this?
  - What are functions, modules, return values, function calls
- What's an APT and how do you solve them?
  - Why are you writing a function?
  - Who calls the function you write?
- What is a list and what is a list comprehension?
  - How to create, modify, and use lists
  - Why lists will change your life … for the better!

Python (C, Javascript, Java, PHP, …)

- High level programming languages
  - *Translate* to lower-level languages: assembly, bytecode
  - *Executed* by a virtual machine or by a chip/real machine
  - *Compile* the high level language into lower level
  - Python compiler/interpreter written in C or Java (or …)
    - Compilers for platforms: Mac, Windows, Linux, …

- Abstractions: foundation of languages
  - Make it easier to think about problems and avoid details
  - Hide details, which can sometimes have issues
  - What is a loop, a list, an int, a String a function …

From high- to low-level Python

```
def reverse(s):
   r = ""
   for ch in s:
      r = ch + r
   return r
```

```
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
      >>   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
      >>   32 POP_BLOCK

10    >>   33 LOAD_FAST        1 (r)
           36 RETURN_VALUE
```

- Create version on the right using dissassembler
  `dis.dis(code.py)`

High level, low level, abstractions

- Python byte-code is executed by…
  - Platform specific virtual machine/environment
  - Similar to Java
- Javascript code is executed by …
  - Platform specific browser (Firefox, IE, Chrome, Opera, …)
  - Is HTML executed?
- C++ code is executed by …
  - The CPU and the operating system, from compiled code
  - Compiler is platform specific
- Microsoft word is executed by …
  - Platform specific OS, CPU, from compiled executable

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

Debugging APTs: Going green

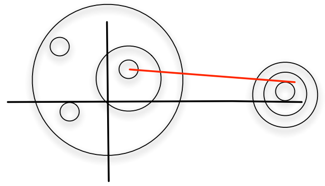
- TxMsg APT: from ideas to code to green
  - What are the main parts of solving this problem?
  - Transform words in original string
    - Abstract that away at first
  - Finding words in original string
    - How do we do this?

```
def getMessage(original):
    ret = ""
    for word in original.split():
        ret = ret + " " + transform(word)
    return ret    #initial space?
```

Debugging APTs: Going green

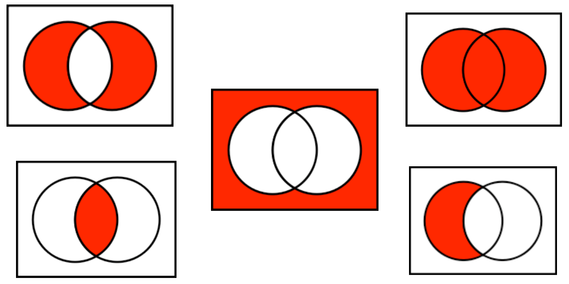
- CirclesCountry APT: from ideas to code to green
  - How do we solve the problem? May not be apparent
  - How do we loop over circles? What is a circle?
    - When is a point inside a circle?

```
x = leastBorder([-3,2,2,0,-4,12,12,12],
[-1,2,3,1,5,1,1,1],[1,3,1,7,1,1,2,3],2,3,13,2)
```



Set, Logic Operations from pictures

- http://en.wikipedia.org/wiki/File:Venn0111.svg



Understanding cgratio APT

- How do you count 'c' and 'g' content of a string?
  - Toward a transformative approach v. modification/mutate

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