Compsci 6/101: PFTW

- 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

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

- Create version on the right using disassembler

```python
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
### Reading and understanding Python

- **When a program executes where does it start?**
  - When you click the 'run' button, what happens?
  - What does it mean to 'execute sequentially'?
  - What happens when one function calls another (e.g., FileFilter.py or OldWoman.py)

- **Simple illustration:**

http://www.kongregate.com/games/Coolio_Niato/light-bot

### 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?

```python
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?

\[
\begin{align*}
    x = \text{leastBorder}([-3, 2, 2, 0, -4, 12, 12], [-1, 2, 3, 1, 5, 1, 1, 1], [1, 3, 1, 7, 1, 1, 2, 3], 2, 3, 13, 2)
\end{align*}
\]
Set, Logic Operations from pictures


---

Revisiting cgratio APT

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

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

def cgcount2(strand):
    cg = [1 for ch in strand if ch == 'c' or ch == 'g']
    return sum(cg)
```

---

List Comprehensions

- **Creating a list from another list, two decisions:**
  - Is new list the same size as original, or smaller?
  - Are elements the same or related by some correspondence?

```python
words = ['bear', 'lion', 'zebra', 'python']
w2 = [w for w in words if some_property(w)]
w3 = [f(w) for w in words]
w4 = [1 for w in words if some_property(w)]
```

- **Once we have list can apply list functions**
  - We have: len, sum, max, min
  - Can "invent" others by writing functions

---

List Comprehensions Again

- **Transformative approach can scale differently**
  - Functional programming: code generates and doesn't modify
  - Basis for (ultra) large scale mapreduce/Google coding

```python
w = [expr for elt in list if bool-expr]
w = [f(w) for w in list if bool_expr(w)]
```

- **Why are abstractions important?**
  - Reason independently of concrete examples
  - Generalize from concrete examples
    - [http://wapo.st/e5ZtkB](http://wapo.st/e5ZtkB)