PFThursday

- Review Organization and Problem-Solving
  - Defining functions, calling functions
  - Return types, print, None

- Incremental construction as design pattern
  - Build programs: start small, add with confidence
  - Build new strings: append/concatenate values
  - Build lists (later, but similar to strings)


- How do you solve this problem?
  - If you have confidence you can solve for any size pan, then start programming
  - If you can't do it by hand …
    - Get some credit for APT, some dancing!

- Sometimes APTs have hard algorithms
  - Translating to code not so bad

- Sometimes APTs have easy algorithms
  - Translating to code is difficult

Three pancakes in a two-cake pan…

- Number of cakes in the system
  - First 5 minutes
  - Second 5 minutes

Three pancakes in a two-cake pan…

- Number of cakes in the system
  - Third 5 minutes

- How many minutes to cook all three pancakes?
Methodically by hand, small values

- Pan has capacity 8, vary #pancakes
  - Can you cook 11 in 15 minutes? Why?
  - Can you cook 13 in 15 minutes? Why?

<table>
<thead>
<tr>
<th>cakes</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>?</td>
<td>10</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
</tbody>
</table>
Pancake Algorithm

- If you have pan of size 17 and 34 pancakes
- If you have pan of size 17 and 43 pancakes
- Pan fits 100 pancakes, but you have 452
- Pan fits N pancakes, but you have P
  - if $P \leq N$ then time needed is ...
  - $X = P/N$, what does this mean for time?
  - $Y = P \% N$, what does this mean for time?

Eclipse Interlude

- Finishing the Pancake problem
  - Translating problem-solving ideas to code
  - Control with if/elif: arithmetic with / and %

Algorithmic Problem/Program Testing

- Complete this form for two more APTs
  

How to teach pancake flipping

- http://www.youtube.com/watch?v=W_gxLKSsSIE
  - For longer, more complex robotic tasks
    - http://www.youtube.com/watch?v=4usoE981e7I

- Do robots matter?
  - Do they dream?
  - Self-driving cars?
  - Machine learning?
Three versions of is_vowel

```python
def is_vowel(ch):
    if ch == 'e':
        return True
    if ch == 'a':
        return True
    if ch == 'i':
        return True
    if ch == 'o':
        return True
    if ch == 'u':
        return True
    return False
```

```python
def is_vowel(ch):
    c = "aeiou".count(ch)
    if c > 0:
        return True
    else:
        return False
```

```python
def is_vowel(ch):
    return "aeiou".count(ch) > 0
```

Python if statements and Booleans

- In python we have if: else: elif:
  - Used to guard or select block of code
  - If guard is True then, else other

- What type of expression used in if/elif tests?
  - ==, <=, <, >, >=, !=, and, or, not, in
  - Value of expression must be either True or False
  - Type == bool, George Boole, Boolean,

- Examples with if
  - String starts with vowel (useful for APT Emphasize)

Eclipse Interlude

- Finishing Emphasize
  - Identifying vowels
  - Helper functions
  - Slicing strings

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!
Building Totem in stages/incrementally

- **What functions do not return values?**
  - They print strings returned by other functions
- **For totem and randompole, which one first?**
  - Don't do both at same time, grow the program
- **Start simple**
  - Next?
  - Add?
  - Questions?

```python
def hair_part():
    return "xxyyzz"
def eye_crossed():
    return "123456"
def totem():
    print hair_part()
    print eye_crossed()
```

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(), ...
    - st.find(), st.count()
- **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**

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

Incremental + : numbers and strings

- **Wht vwls cn y stil rd ths sntnce?**
  - 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

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

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?