

## PFThursday 9/17

- **Design pattern of accumulation**
  - Selective summing to tally a count, `val += 1`
  - Creating strings by concatenating, `s += ch`
  - Appending to a list to grow, `lst.append(elt)`
  - Using join to create string from list in one simple statement, `':' .join(['1', '2', '3'])`
- **Compsci 101 specifics: Python -> Course**
  - APT Quiz and ensuring you do well

Finish reviewing classwork from last time....

<http://bit.ly/101fall15-0915-1>

## Problem Solving

<http://bit.ly/101fall15-0917-2>

## From APT 3 - TxMsg

<http://www.cs.duke.edu/csed/pythonapt/txmsg.html>

### Problem Statement

Strange abbreviations are often used to write text messages on uncomfortable mobile devices. One particular strategy for encoding texts composed of alphabetic characters and spaces is the following:

- Spaces are maintained, and each word is encoded individually. A word is a consecutive string of alphabetic characters.
- If the word is composed only of vowels, it is written exactly as in the original message.
- If the word has at least one consonant, write only the consonants that do not have another consonant immediately before them. Do not write any vowels.
- The letters considered vowels in these rules are 'a', 'e', 'i', 'o' and 'u'. All other letters are considered consonants.

### Specification

```
filename: TxMsg.py
def getMessage(original):
    """
    return String that is 'textized' version
    of String parameter original
    """
    # you write code here
```

## Examples

- Do one by hand?
- Explain to partner?
- Identify Pythonic/programming challenges?

1.	"text message" Returns "tx msg"
2.	"ps i love u" Returns: "p i lv u"
3.	"please please me" Returns: "ps ps m"
4.	"back to the ussr" Returns "bc t t s"
5.	"aeiou bcdghjklmnpqrstvwxyz" Returns: "aeiou b"

Compsci 101.2, Fall 2015

8.5

## 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 = ""  
  
    ret = ret + " " + transform(word)  
    return ret #initial space?
```

Compsci 101.2, Fall 2015

8.6

## 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?
```

Compsci 101.2, Fall 2015

8.7

## Why use helper function 'transform'?

- Structure of code is easier to reason about
  - Harder to develop this way at the beginning
  - Similar to accumulate loop, build on what we know
- We can debug pieces independently
  - What if transform returns "" for every string?
  - Can we test transform independently of getMessage?

Compsci 101.2, Fall 2015

8.8

## Python via Problem Solving

- In the loop for TxMsg we saw:  
`ret = ret + " " + transform(word)`
  - Why does this leave "extra" space at front?
  - Eliminate with `ret.strip()`
- Alternate: collect transform words in list, use join to return
  - Rather than construct string via accumulation and concatenation, construct list with append

## Analyzing/replaying a solved problem

- More than one way to do something?
    - Is this easier to understand? Harder?
    - What does `.join` do? Try it!
- ```
def getMessage(original):  
    trans = []  
    for word in original.split():  
        trans.append(transform(word))  
    return ' '.join(trans)
```

## Understanding TotemCrawler

- Using assignments to understand code
  - Treat some code as a "black box", other times as a clear box worth looking at.
  - APIs and what's under the hood

<http://bit.ly/101fall15-0915-2>