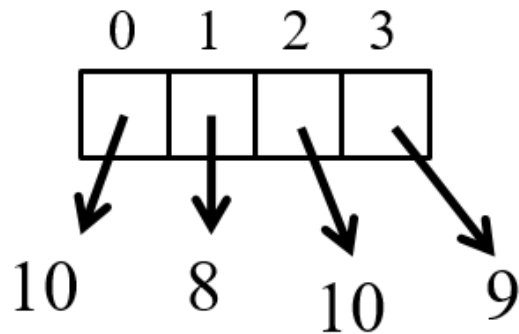


CompSci 101

Introduction to Computer Science

score = [10,8,10,9]



Sep 21, 2017

Prof. Rodger

Announcements

- Reading and RQ8 due next time
- Assignment 3 due tonight
 - Assignment 4 out, due Oct. 3
- APT 3 is due on Tuesday
- APT Quiz 1 take Sunday-Wednesday 11:59pm
 - practice APT quiz available
- Today
 - Breaking apart and putting back together.
 - Thinking about solving assignments, apts

Assignment 4 out today, due Oct 3

- **Transform 1** – PigLatin.

The angry bear climbed the tree.

e-thay angry-way ear-bay imbed-clay
e- thay ee.-tray

→ The angry bear climbed the tree.

- **Transform 2** – Caesar Cipher encryption

The angry bear climbed the tree.

Aol hunyf ilhy jsptilk aol ayll.

→ The angry bear climbed the tree.

Getting help

- Consider a peer tutor – one hour of one on one help a week.
 - Many take advantage of this
 - contact peer tutoring center
- Are you getting too much help?
 - After solving APT
 - Can you solve again with a blank sheet of paper or blank file and no help?
- Are you using 7 step process to solve?

Are you Learning How to Debug?

- Do a little bit at a time, make sure it works!
- Print is your friend!
- Create variables!
- Isolate the problem
 - Comment out sections until you can isolate where the problem is
- Python Tutor – trace
 - Doesn't work with files but comment out file and create variable with sample input

Incremental + : numbers and strings

- Wtht vwls 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

```
def noVowels(word):  
    ret = ""  
    for ch in word:  
        if not isVowel(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

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

- Alternative version of adding:
value += 1

Assignment 3 Questions

bit.ly/101f17-0921-1

Filtering data

- List of all the earthquakes
- **FILTER** – those magnitude 2.0 or greater
 - List of earthquakes 2.0 or greater
- **FILTER** – those earthquakes in Alaska
 - List of earthquakes from Alaska 2.0 or greater
- NOTE you still have a list

String Functions – What is output?

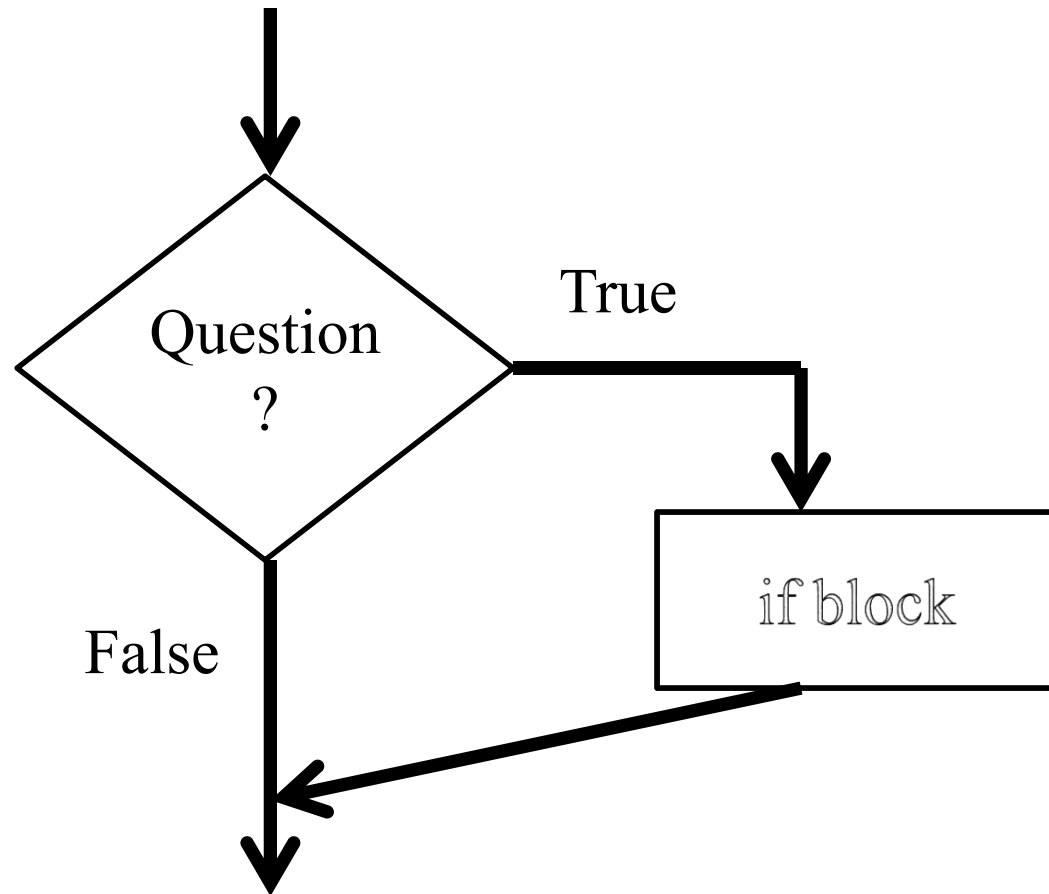
```
name = "VVDarth Vater Darth VaterVVV"  
nm = name.strip("V")
```

```
phrase = "mississippi"  
phrase = phrase.replace("ss", "pp")
```

```
last = "Darth Vater or Darth Vater"  
last = last.replace("a", "o").replace("or", "es")
```

```
b = "the end is near oh dear"  
a = b.endswith('s')
```

Making Decisions



Making Decisions in Python

if condition1:

Block of code to do if condition is true

elif condition2:

Block of code to do if condition1 false, condition2 is true

else:

Block of code to do if other conditions false

- Can have many elifs, leave out elif, leave out else

Making Decisions tools

- Boolean values: True, False
- Boolean operators: and, or, not

X	Y	X and Y	X or Y
True	True	True	True
True	False	False	True
False	True	False	True
False	False	False	False

- Relational operators: <, <=, >, >=
- Equality operators: ==, !=

bit.ly/101f17-0921-2

```
def isVowel(letter):
    answer = False
    if letter == 'a':
        answer = True
    elif letter == 'e':
        answer = True
    elif letter == 'i':
        answer = True
    elif letter == 'o':
        answer = True
    elif letter == 'u':
        answer = True
    return answer
```

```
def isVowel2(letter):
    answer = False
    if letter == 'a':
        answer = True
    if letter == 'e':
        answer = True
    if letter == 'i':
        answer = True
    if letter == 'o':
        answer = True
    if letter == 'u':
        answer = True
    return answer
```

```
def isVowel3(letter):
    if letter == 'a':
        return True
    else:
        return False
    if letter == 'e':
        return True
    else:
        return False
    if letter == 'i':
        return True
    else:
        return False
    if letter == 'o':
        return True
    else:
        return False
    if letter == 'u':
        return True
    else:
        return False
```

```
def isVowel4(letter):
    answer = False
    if letter == 'a':
        answer = True
    else:
        answer = False
    if letter == 'e':
        answer = True
    else:
        answer = False
    if letter == 'i':
        answer = True
    else:
        answer = False
    if letter == 'o':
        answer = True
    else:
        answer = False
    if letter == 'u':
        answer = True
    else:
        answer = False
    return answer
```

Lists

- A list is a collection of objects

```
scores = [99, 78, 91, 84]
```

```
allAboutMe = ["Mo", 25, "934-1234"]
```

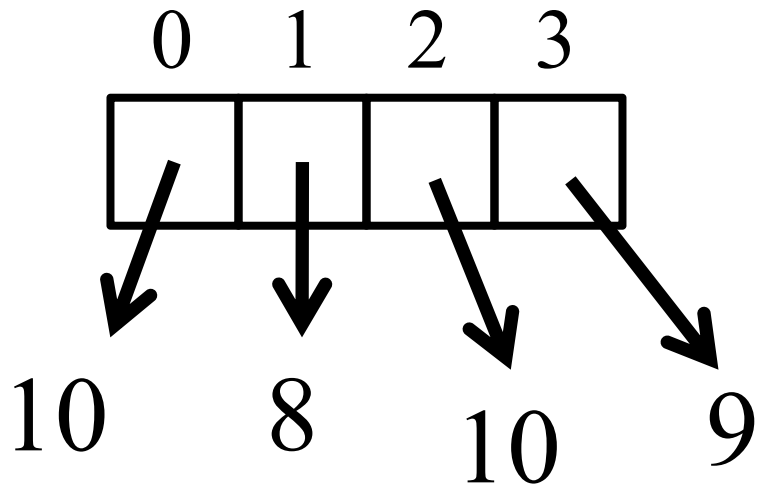
```
club=['Mo', 'Jo', 'Po', 'Flo', 'Bo']
```

- Lists are *mutable* – use [num] to change a value
- Lists are indexed starting at 0, or -1 from the end
- Functions: max, min, len, sum
- Slice lists [:]

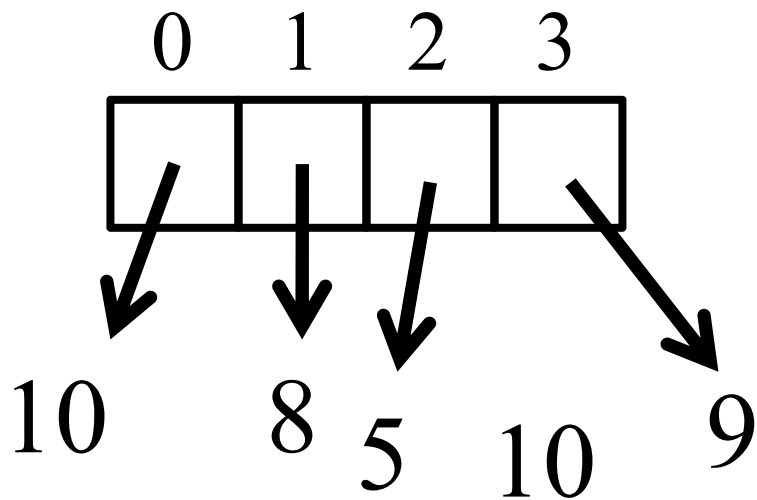
List Examples

```
scores = [10, 8, 10, 9]
print scores
scores[2] = 5
print scores
print max(scores), len(scores)
print sum(scores)
print scores[1:]
print scores[1], scores[-1]
scores.append(4)
scores += [5]
print scores
```

List before/after modification



score = [10,8,10,9]



score [2] = 5

More List Examples

- `phrase = "earthquake, 1.3, 81km SSW of Kobuk, Alaska"`
- `phrase.split(",")` vs `phrase.split()` vs `phrase.split("a")`
- `phrase = "Duke will beat UNC"`
- `alist = phrase.split()`
- `' '.join(alist)` vs `'+'.join(alist)` vs `"YES".join(alist)`
- `append` vs `+= [item]`

Design pattern of accumulation *for item in something*

- Summing to tally a count
value += 1
- Building a new string by concatenating
str += ch
- Building a new list by appending
lst.append(element)
OR
lst += [element]

Processing List Items

- Process all the items in a list, one item at a time
- Format: `for variable in list:`
 `process variable`

- Example:

```
sum = 0
```

```
nums = [6, 7, 3, 1, 2]
```

```
for value in nums:
```

```
    sum = sum + value
```

```
print sum
```

Learn list functions

```
nums = [6, 7, 3, 1, 2]  
print sum(nums)
```

Problem: Sum up even numbers in list of numbers

- Could do it similar to two slides back
- OR Build a list of the correct numbers, then sum

How to build list of evens and sum?

bit.ly/101f17-0921-3

```
def sumUpEven(nums):  
    answer = question1  
    for item in nums:  
        if question2:  
            question3  
    return question4
```

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 bcd fghjklmnpqrstvwxyz"`

Returns: `"aeiou b"`

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?
```

Write helper function *transform*

- How?
- Use seven steps
- Work an example by hand

Transform word - Step 1: work small example by hand

- Word is “please”
- Letter is ‘p’, YES
- answer is “p”
- Letter is ‘l’, NO
- Letter is ‘e’, NO
- Letter is ‘a’, NO
- Letter is ‘s’, YES
- answer is “ps”
- Letter is ‘e’, NO

Step 2: Describe what you did

- Word is “please”, create an empty answer
- Letter is ‘p’, consonant, no letter before, YES
- Add ‘p’ to answer
- Letter is ‘l’, consonant, letter before “p”, NO
- Letter is ‘e’, vowel, letter before ‘l’, NO
- Letter is ‘a’, vowel, letter before ‘e’, NO
- Letter is ‘s’, consonant, letter before ‘a’, YES
- Add ‘s’ to answer
- Letter is ‘e’, vowel, letter before ‘s’, NO
- Answer is “ps”

Step 3: Find Pattern and generalize

Need letter before, pick “a”

answer is empty

for each letter in word

If it is a **consonant**, and the **letter before** is a vowel, then add the letter to the answer

This letter is now the letter before

return answer

Step 4 – Work another example

- Word is message
- Letter is ‘m’, before is ‘a’, add ‘m’ to answer
- Letter is ‘e’, before is ‘m’, NO
- Letter is ‘s’, before is ‘e’, add ‘s’ to answer
- Letter is ‘s’, before is ‘s’, NO
- Letter is ‘a’, before is ‘s’, NO
- Letter is ‘g’, before is ‘a’, add ‘g’ to answer
- Letter is ‘e’, before is ‘g’, NO
- Answer is “msg” **WORKS!!**

Step 5: Translate to Code

Letter before is “a” # start with a vowel

answer is empty

for each letter in word

Step 5: Translate to Code (code)

#If it is a consonant, and the letter before is a
#vowel, then add the letter to the answer

#This letter is now the letter before

return answer

Will our program work for?

- STRING GET SHOULD GET
- green
- apple
- a
- aeiuo
- grrr

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?

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