CompSci 101
Introduction to Computer Science

March 9, 2017
Prof. Rodger
Lecture by Lesia Semenova

Rule Lists

Go for a walk?
• If it’s rainy then no
• Else if it’s sunny then yes
• Else if it’s warm then yes
• ....
• Default no

Announcements

• Reading and RQ due next time
• Assignment 5 due tonight
• APT 6 due Thursday, March 23

• Today:
  – Debugging
  – Which code is better?
Bug and Debug

- software 'bug'
- Start small
  - Easier to cope
  - Simplest input?
- Judicious 'print'
  - Debugger too
- Python tutor
  - Visualizes data
  - step through

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

Debugging Problems

- Today the main focus is on debugging.
- There are several problems. Trace by hand to see if you can figure out if they are correct or not, or what to do to correct them.
- Enter your answers on the google form

Debug 1 – Does it work?
bit.ly/101s17-0309-1

- The function \texttt{sizes} has a parameter named \texttt{words} that is a list of strings. This function returns a list of the sizes of each string. For example, \texttt{sizes(['This', 'is', 'a', 'test'])} should return the list [4, 2, 1, 4]

```python
def sizes(words):
    nums = []
    for w in words:
        nums = len(w)
    return nums
```

Debug 2 – Does it work?
Bit.ly/101s17-0309-2

- The function \texttt{buildword} has a parameter \texttt{words} that is a list of strings. This function returns a string that is made up of the first character from each word in the list. For example, \texttt{buildword(['This', 'is', 'a', 'test'])} returns 'Tiat'

```python
def buildword(words):
    answer = ''
    for w in words:
        answer += w[:1]
    return answer
```
The function *middle* has a parameter *names* that is a list of strings, which each string is in the format "firstname:middlename:lastname". This function returns a list of strings of the middlenames.

For example, the call `middle( "Jo:Mo:Tree", "Mary:Sue:Perez", "Stephen:Lucas:Zhang")` returns `['Mo', 'Sue', 'Lucas']`

---

The function *removeOs* has one string parameter named *names*. This function returns a string equal to *names* but with all the lowercase o's removed. For example, `removeOs('Mo Moo Move Over')` returns ‘M M Mve Over’

---

The function *uniqueDigits* has one int parameter *number*. This function returns the number of unique digits in *number*. For example, the call `uniqueDigits(456655)` should return 3.

---

```python
def middle(names):
    middlelist = []
    for name in names:
        name.split(':
    middlelist.append(name[1])
    return middlelist

def removeOs(word):
    position = word.find("o")
    while position != -1:
        word = word[:position] +
        word[position+1:]
    return word

def uniqueDigits(number)
    digits = [ ]
    while number > 0:
        digits.append(number % 10)
        number = number / 10
    return len(digits)
```
Which code is better?

- For the next two problems, we will look at two examples of code that both work in solving the problem, and think about which code is better.

Problem 6: Which code is better?

- Problem: Given a string parameter named phrase and string named letter, the function findWords returns a list of all the words from phrase that have letter in them.
- Example:
  - findWords("the circus is coming to town with elephants and clowns", "o") would return ['coming', 'to', 'town', 'clowns']

Consider two solutions, which is better? Bit.ly/101s17-0309-6

```python
def findWords(phrase, letter):
    return [phrase.split()[i] for i in range(len(phrase.split())) if letter in phrase.split()[i]]

def findWords2(phrase, letter):
    wordlist = phrase.split()
    answer = []
    for i in range(len(wordlist)):
        if letter in wordlist[i]:
            answer.append(wordlist[i])
    return answer
```

Problem 7 – Which number appears the most times?

- The function most has one parameter nums, a list of integers. This function returns the number that appears the most in the list.
- For example, the call most([3,4,2,2,3,2]) returns 2, as 2 appears more than any other number.
Solution 1

def most(nums):
    maxcnt = 0
    maxnum = -1
    cnts = [0 for n in range(max(nums)+1)]
    for num in nums:
        cnts[num] += 1
        if cnts[num] > maxcnt:
            maxcnt = cnts[num]
            maxnum = num
    return maxnum

Compare with Solution 2

bit.ly/101s17-0309-7

def most2(nums):
    maxcnt = 0
    maxnum = -1
    for num in set(nums):
        cnt = nums.count(num)
        if cnt > maxcnt:
            maxcnt = cnt
            maxnum = num
    return maxnum