CompSci 101
Introduction to Computer Science

Feb 28, 2017
Prof. Rodger
You can see your Exam 1… on Gradescope

• Solutions posted – regrades til March 3
  – Ask for regrade on gradescope

• Try working problem you missed first
  – Then look at solution

• Once you think you understand
  – Get blank sheet of paper – try again

• Understand all solutions
Announcements

• Reading and RQ due next time
• Assignment 5 out today
• APT 4 due today, APT 5 out
• Lab 6 this week
  – Read APT Anagramfree and Assignment 5 before going to lab!

• Today:
  – Focus on problem solving with sets, list comprehensions
Build a list from another list

• Given a list of numbers, create a second list of every number squared.

```python
nums = [8, 3, 5, 4, 1]
sqnums = []
for v in nums:
    sqnums.append(v*v)
print sqnums
```

```
[64, 9, 25, 16, 1]
```
List Comprehension -
Short cut way to build a list

• Take advantage of patterns, make a new list based on per element calculations of another list

• Format:
  
  \[\text{[<expression with variable> for <variable> in <old list>]}\]

• Example:

  \[\text{nums = [8, 3, 5, 4, 1]}\]

  \[\text{sqnums = [v*v for v in nums]}\]
These result in the same list!

```python
ums = [8, 3, 5, 4, 1]

1) sqnums = []
   for v in nums:
       sqnums.append(v*v)

2) sqnums = [v*v for v in nums]
```
Examples of List Comprehensions

bit.ly/101s17-0228-1

```python
nums = [4, 3, 8]
x = [v for v in nums]
x = [2 for v in nums]
x = sum([v*2 for v in nums])
x = [v+5 for v in nums][1]
x = [ nums[len(nums)-i -1] for i in range(len(nums)) ]
```
Creating a list with just the even numbers

```python
nums = [8, 3, 5, 4, 1]
evennums = []
for v in nums:
    if v % 2 == 0:
        evennums.append(v)
print evennums

[8, 4]
```
List Comprehension with Filtering

• Create list and use “if” to filter out elements to the list

• Format:

• [<expression with variable> for <variable> in <old list> if <filter with variable> ]

• Example:  nums = [8, 3, 5, 4, 1]

  evennums =

  [v for v in nums if v%2==0]
names = [“Bo”, “Moe”, “Mary”, “Aaron”, “Joe”]

• What is the list for the following:

1) [w for w in names if w.endswith(“e”)]
2) [w for w in names if w.lower()[0] > ‘c’]
3) [j+1 for j in range(20) if (j%3) == 0]
4) [i*2 for i in [j+1 for j in range(20) if (j%3) == 0] if i*i > 19]
More on List Comprehensions
bit.ly/101s17-0228-3

• Problem: Given a list of strings, return the longest string. If there are more than one of that length, return the first such one.


   Use a list comprehension for this problem
Richard Stallman

- MacArthur Fellowship (Genious grant)
- ACM Grace Murray Hopper award
- Started GNU – Free Software Foundation (1983)
  - GNU Compiler Collection
  - GNU Emacs
Python Sets

• Set – unordered collection of distinct items
  – Unordered – can look at them one at a time, but cannot count on any order
  – Distinct - one copy of each

• Operations on sets:
  – Modify: add, clear, remove
  – Create a new set: difference(-), intersection(&), union (|), symmetric_difference(^)
  – Boolean: issubset <=, issuperset >=

• Can convert list to set, set to list
  – Great to get rid of duplicates in a list
List vs Set

• List
  – Ordered, 3rd item, can have duplicates
  – Example: \( x = [4, 6, 2, 4, 5, 2, 4] \)

• Set
  – No duplicates, no ordering
  – Example: \( y = \text{set}(x) \)

• Both
  – Add, remove elements
  – Iterate over all elements
Summary (from wikibooks)

- set1 = set()                   # A new empty set
- set1.add("cat")                # Add a single member
- set1.update(["dog", "mouse"]))  # Add several members
- set1.remove("cat")             # Remove a member - error if not there

- print set1
- for item in set1:              # Iteration or “for each element"
  print item
- print "Item count:", len(set1) # Length, size, item count
- isempty = len(set1) == 0    # Test for emptiness
- set1 = set(["cat", "dog"]))  # Initialize set from a list
- set3 = set1 & set2            # Intersection
- set4 = set1 | set2              # Union
- set5 = set1 - set3            # Set difference
- set6 = set1 ^ set2             # Symmetric difference (elements in either set but not both)
- issubset = set1 <= set2        # Subset test
- issuperset = set1 >= set2      # Superset test
- set7 = set1.copy()             # A shallow copy (copies the set, not the elements)
- set8.clear()                   # Clear, empty, erase
Creating and changing a set

colorList = ['red', 'blue', 'red', 'red', 'red', 'green']
colorSet = set(colorList)
smallList = list(colorSet)
colorSet.clear()
colorSet.add("yellow")
colorSet.add("red")
colorSet.add("blue")
colorSet.add("yellow")
colorSet.add("purple")
colorSet.remove("yellow")

What is the value of smallList and colorSet after this code executes?
Set Operations

UScolors = set(['red', 'white', 'blue'])
dukeColors = set(['blue', 'white', 'black'])
print dukeColors.union(UScolors)
print dukeColors | UScolors
print dukeColors.intersection(UScolors)
print dukeColors & UScolors
print dukeColors.difference(UScolors)
print dukeColors - UScolors
print UScolors - dukeColors
print dukeColors ^ UScolors
print UScolors ^ dukeColors
Set Examples

bit.ly/101s17-0228-4

poloClub = set(['Mary', 'Laura', 'Dell'])
rugbyClub = set(['Fred', 'Sue', 'Mary'])

Questions:

print [w for w in poloClub.intersection(rugbyClub)]
print poloClub.intersection(rugbyClub)
print [w for w in poloClub.union(rugbyClub)]
print poloClub.union(rugbyClub)
Set Examples (cont)

lista = ['apple', 'pear', 'fig', 'orange', 'strawberry']
listb = ['pear', 'lemon', 'grapefruit', 'orange']
listc = [x for x in lista if x in listb]
listd = list(set(lista)|set(listb))
Assignment 5 - Hangman

• Guess a word given the number of letters.
  – Guess a letter
  – see if it is in the word and where.

• Demo

• Will start in lab
words = ["creation","sentence","reaction","sneak","star","rats","snake"]

Returns: 4

“star” “rats”  → both have letters: a r t s
“snake” “sneak”
“creation” “reaction”
“sentence”