Compsci 101
List Comprehensions, Global, Parallel Lists

Susan Rodger
February 15, 2022
K is for …

• **Kernel**
  - Core of the OS, Core for Machine Learning

• **Keyboard - QWERTY or DVORAK**
  - DVORAK:

    ![DVORAK Keyboard Layout]

• **Key and (Key,Value) pair**
  - Heart of a dictionary
Tiffany Chen

- Duke BS - IDM CS/Biology
- Stanford PhD Biomedical Informatics (CS and Biomedicine)
- Was Director of Informatics, Cytobank
- Now Group Product Manager at Chan Zuckerberg Initiative

“If you are interested in a PhD, I would suggest doing a summer research experience as an undergraduate, but also an internship in industry. You can see how problems are solved in the real world”

“Part of the advantage of being interdisciplinary is that you can see the big picture when no one else can, and you can communicate to everyone else what that big picture is”
Announcements

• Assign 2 – Turtles due tonight!
• APT-3 due Thursday
• Assign 3-Transform out today, due Tuesday, March 1
  • There is a Sakai quiz on Assign3 – Due Feb 28
• Lab 6 Friday - Do prelab before lab
• Exam 2 – one week – in person
• APT Quiz 1 is Feb 24-Feb 27
  • Take during this time
  • Two parts – each part has two APTs
  • Each part is timed
  • More details Thursday
Exam 2 – in person – Tues, Feb 22

- Exam is in class on paper – 10:15am
  - Need pen or pencil
- See materials under 2/22 date
  - Exam 2 Reference sheet - part of exam
- Covers
  - topics /reading through today
  - APTs through APT3
  - Labs through Lab 5, Lab 6 (Part 1 – list comprehensions)
  - Assignments through Assignment 2
  - Concepts from Assign2, No turtles
PFTD

• Pancakes
• Parallel Lists
• List Comprehensions
• Transform Assignment
Pancakes!
APT Pancake

- How do you solve this (or any) problem?
  - 7 Steps!

- Some APTs are hard problems to solve (step 1-4)
  - Translating to code easy

- Some APTs have easy-to-see algorithms (step 5)
  - Translating to code is hard
APT: Pancakes

Problem Statement

You're a short-order cook in a pancake restaurant, so you need to cook pancakes as fast as possible. You have one pan that can fit \texttt{capacity} pancakes at a time. Using this pan you must cook \texttt{numCakes} pancakes. Each pancake must be cooked for five minutes on each side, and once a pancake starts cooking on a side, it has to cook for five minutes on that side. However, you can take a pancake out of the pan when you're ready to flip it after five minutes and put it back in the pan later to cook it on the other side.

Write the method, \texttt{minutesNeeded}, that returns the shortest time needed to cook \texttt{numCakes} pancakes in a pan that holds \texttt{capacity} pancakes at once. See the examples.

<table>
<thead>
<tr>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{filename: Pancakes.py}</td>
</tr>
<tr>
<td>\texttt{def minutesNeeded (numCakes, capacity):}</td>
</tr>
<tr>
<td>\texttt{&quot;&quot;&quot;}</td>
</tr>
<tr>
<td>\texttt{\quad return integer representing time to cook pancakes}</td>
</tr>
<tr>
<td>\texttt{\quad based on integer parameters as described below}</td>
</tr>
<tr>
<td>\texttt{&quot;&quot;&quot;}</td>
</tr>
</tbody>
</table>
Examples

1. numCakes = 0
capacity = 4

Returns: 0

It takes no time to cook 0 pancakes.

2. numCakes = 2
capacity = 2

Returns: 10

You cook both pancakes on one side for five minutes, then flip them over and cook each on the other side for another five minutes.
Step 1: Solve an instance
Three pancakes in a two-cake pan

- First 5 minutes
  - 2 half cooking
  - 1 uncooked

- Second 5 minutes
  - 2 half cooking
  - 1 almost cooked
Step 1: Solve an instance
Three pancakes in a two-cake pan

- Third 5 minutes
  - 1 done
  - 2 almost cooked

- How many minutes to cook all three pancakes?
Step 1: Solve an instance

• What kind of instances? Simple cases that are quickly solved
  • What are these in Pancake problem?

• Don’t solve for N, solve for 5 (generalize is step 3)
  • What do when there are two parameters?
    • Fix one, vary the other one
  • Helps identify cases
WOTO-1 Pancakes

Pancake flipping Video
How to teach pancake Flipping

• [http://www.youtube.com/watch?v=W_gxLKSsSIE](http://www.youtube.com/watch?v=W_gxLKSsSIE)
  • For longer, more complex robotic tasks
    • [http://www.youtube.com/watch?v=4usoE981e7I](http://www.youtube.com/watch?v=4usoE981e7I)
Problem

• Given a file of words, which word occurs the most
• For each word count how many times it occurs
• Determine which word has the highest count
Parallel Lists

- We will use parallel lists to track data
  - Each word is stored in a list named `words`
  - Word’s count is stored in a list named `counts`
  - # occurrences of `words[k]` is in `counts[k]`

```
["apple", "fox", "vacuum", "lime"]
[  5,    2,    25,    15  ]
```
Parallel Lists

- We will use parallel lists to track data
  - Each word is stored in a list named `words`
  - Word’s count is stored in a list named `counts`
  - # occurrences of `words[k]` is in `counts[k]`

```
["apple", "fox", "vacuum", "lime"]
[  5,   2,   25,    15  ]
```

- For example: “apple” has been seen five times
Parallel Lists

• We will use parallel lists to track data
  • Each word is stored in a list named \textit{words}
  • Word’s count is stored in a list named \textit{counts}
  • \# occurrences of \texttt{words[k]} is in \texttt{counts[k]}

\begin{verbatim}
["apple", "fox", "vacuum", "lime"]
[ 5,    2,    25,    15 ]
\end{verbatim}

• For example: “vacuum” has been seen 25 times
Parallel Lists

• We will use parallel lists to track data
  • Each word is stored in a list named `words`
  • Word’s count is stored in a list named `counts`
  • # occurrences of `words[k]` is in `counts[k]`

`["apple", "fox", "vacuum", "lime"]`
`[ 5, 2, 25, 15 ]`

• What happens when we read a word?

Read word “vacuum”?
Parallel Lists

• We will use parallel lists to track data
  • Each word is stored in a list named `words`
  • Word’s count is stored in a list named `counts`
  • # occurrences of `words[k]` is in `counts[k]`

```
["apple", "fox", "vacuum", "lime"]
[  5,    2,    26,    15  ]
```

• What happens when we read a word?

Read word “cat”?
def wordOccursTheMost(fname):
    f = open(fname)
    words = []
    counts = []
    for line in f:
        line = line.strip() #remove newline
        data = line.split()
        for word in data:
            if word not in words:
                words.append(word)
                counts.append(1)
            else: # update word
                pos = words.index(word)
                counts[pos] += 1
    f.close()
WOTO-2 Word Most Often
List Comprehension
Accumulator in one line

```python
def onlyPos(nums):
    ret = []
    for n in nums:
        if n > 0:
            ret.append(n)
    return ret

print(onlyPos([1,2,3,-1,-2,-3]))
```

- List Comprehension
  - We will use a complete, but minimal version of list comprehensions, much more is possible
List Comprehension Syntax

- $V$ is any variable: all list elements in order
- $V_{\text{EXP}}$ is any expression, often use $V$

(ret = [])
for $V$ in LIST:
  ret.append($V_{\text{EXP}}$)

(ret = [V_EXP for V in LIST])

(ret = [])
for $V$ in LIST:
  if BOOL_EXP:
    ret.append($V_{\text{EXP}}$)

(ret = [V_EXP for V in LIST if BOOL_EXP])
List Comprehension Syntax

```
ret = []
for V in LIST:
    ret.append(V_EXP)
```

```
ret = [V_EXP for V in LIST]
```

```
ret = []
for V in LIST:
    if BOOL_EXP:
        ret.append(V_EXP)
```

```
ret = [V_EXP for V in LIST if BOOL_EXP]
```

- if part optional - BOOL_EXP is a Boolean expression usually using V
List Comprehension Examples

print([n*2 for n in range(5)])

print([n for n in range(10) if n % 2 == 1])
List Comprehension Examples

print( [n/2 for n in range(10) if n % 2 == 0] )

lst = ['banana', 'pineapple', 'apple']
print( [c for c in lst if 'n' in c] )
Assignment 3: Transform

• Reading and writing files
  • We've seen how to read, writing is similar
  • Open, read, and close
  • Open, write, and close - .write(...)

• Apply a function to every word in a file
  • Encrypt and decrypt
  • Respect lines, so resulting file has same structure
Encrypting and Decrypting

• We give you:
  • Transform.py
  • Vowelizer.py - Removes vowels, then re-vowelize

• You implement
  • Pig Latin
  • Caesar cipher

• Challenge: Shuffleizer
Concepts in Starter Code

• Global variables
  • Generally avoided, but very useful
  • Accessible in all module functions

• FileDialog and tkinter
  • API and libraries for building UI and UX

• Docstrings for understanding!

Look at code
Transform – Remove Vowels

• First line of twain.txt:

1 The Notorious Jumping Frog of Calaveras County

• Run Transform.py on twain.txt

• Set as:

```python
doTransform("-nvw", Vowelizer.encrypt)
#doTransform("-rvw", Vowelizer.decrypt)
```

• Results in new file: twain-nvw.txt

• First line of twain-nvw.txt is:

1 Th Ntrrs Jmpng Frg f Clvrs Cnty
Transform – Get vowels back?

• First line of twain-nvw.txt:

```
1  Th Ntrs Jmpng Frg f ClvrsCnty
```

• Run Transform.py on twain-nvw.txt

• Set as:

```python
#doTransform("-nvw", Vowelizer.encrypt)
doTransform("-rvw", Vowelizer.decrypt)
```

• Results in new file: twain-nvw-rvw.txt

• First line of twain-nvw-rvw.txt is:

```
1 oath antares jumping fargo fe cleavers county
```
Transform – Vowels summary

• First line in twain.txt

  The Notorious Jumping Frog of Calaveras County

• After removing vowels – “encrypt”

  Th Ntrs Jmpng Frg f Clvrs Cnty

• After trying to re-vowelize – “decrypt”

  oath antares jumping fargo fe cleavers county