**Compsci 101**
*List Comprehensions, Global, Parallel Lists*

Susan Rodger
October 6, 2022

**K** is for ...

- **Kernel**
  - Core of the OS, Core for Machine Learning
- **Keyboard - QWERTY or DVORAK**
  - **DVORAK:**
    ```
    Spacebar
    ~ ! 1 @ # $ % ^ & * () _ + |
    Caps Lock A Z Q W E R T Y U I O P
    Shift < > P O N M L K J I H G F E D C B A
    Ctrl Win Key Alt
    Alt Gr Win Key Menu Ctr
    ```
- **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”

“No 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 next Thursday
- Assign 3-Transform out today, due Thursday, Oct 20
  - Sakai quiz Assign3 – Due Tues, Oct 18 (no grace day)
- No lab Friday - Enjoy Fall Break!
- APT Quiz 1 is Oct 13-Oct 17
  - Take during this time
  - Two parts – each part has two APTs
  - Each part is timed
APT Quiz 1

- Is your own work!
  - No collaboration with others!
  - Use your notes, lecture notes, your code, textbook
  - DO NOT search for answers!
  - Do not talk to others about the quiz until grades are posted
- Post private questions on Ed Discussion
  - We are not on between 9pm and 9am!
  - We are not on all the time, especially weekends
  - Will try to answer questions between 9am – 9pm
    - About typos, cannot help you in solving APTs
- See 101 APT page for tips on debugging APTs
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 `capacity` pancakes at a time. Using this pan you must cook `numCakes` pancakes. Each pancake must be cooked for five minutes on each side, and once a pancake starts cooking on one 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, `minutesNeeded`, that returns the shortest time needed to cook `numCakes` pancakes in a pan that holds `capacity` pancakes at once. See the examples.
Step 1: Solve an instance
Three pancakes in a two-cake pan

1. First 5 minutes
   - 2 half cooking
   - 1 uncooked

2. Second 5 minutes
   - 2 half cooking
   - 1 almost cooked

• Third 5 minutes
  - 1 done
  - 2 almost cooked

How many minutes to cook all three pancakes?

15 minutes!

• 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 to do when there are two parameters?
    • Fix one, vary the other one
    • Helps identify cases
Step 1: Solve an instance

- Pan has capacity 8, vary # pancakes
  - Can you cook 12 in 15 minutes? Why?
  - Can you cook 13 in 15 minutes? Why?

<table>
<thead>
<tr>
<th>cakes</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Step 2: What did we just do?

- $13 - 8 = 5$
- $8/2 = 4$ # Can only take off up to half
- Is 5 <= 4?
  - No, warmer trick won’t work
- 10 minutes for 8 pancakes + 10 minutes for 5 more pancakes = 20 minutes
Step 2: What did we just do?

- $17 - 8 = 9$, $9 - 8 = 1$
- $8/2 = 4$
- Is $1 <= 4$? # Yes, warmer trick will work!
- Total: 25 minutes
  - 10 minutes for 8 pancakes +
  - 5 minutes for 8 pancakes +
  - Take 1 out, start $17^{th}$ pancake
  - 5 minutes finish pancakes 8 to 15 +
  - 5 minutes finish pancake 16 and 17

Step 3: Generalize

- Pan has capacity 8, Generalize to algorithm?

<table>
<thead>
<tr>
<th>cakes</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>25</td>
<td>25</td>
</tr>
</tbody>
</table>

Step 4: Test Steps

- Remove as many as can with panCapacity
- Will the remainder fit in half the pan?
  - No, warmer trick won’t work
  - 10 minutes for 8 pancakes + 10 minutes for 5 more pancakes = 20 minutes
- Remove as many as can with panCapacity
- Will the remainder fit in half the pan?
  - Yes, use warmer
    - 5 minutes instead of 10 for last batch
  - No, don’t use warmer
    - 10 minutes for all sets of panCapacity
- Case 1:
  - cap 17, cook 34
Step 4: Test Steps

- Remove as many as can with panCapacity
- Will the remainder fit in half the pan?
  - Yes, use warmer
    - 5 minutes instead of 10 for last batch
  - No, don’t use warmer
    - 10 minutes for all sets of panCapacity
- Case 1:
  - cap 17, cook 34
  - remainder = 0
  - Edge case! No need for warmer
  - Total: 20 minutes
- Case 2:
  - cap 17, cook 42
- Will the remainder fit in half the pan?
  - Yes, use warmer
    - 5 minutes instead of 10 for last batch
  - No, don’t use warmer
    - 10 minutes for all sets of panCapacity
- Case 1:
  - cap 17, cook 34
  - remainder = 0
  - Edge case! No need for warmer
  - Total: 20 minutes
- Case 2:
  - cap 17, cook 42
  - remainder = 8
  - Yes, use warmer
  - Total: 25 minutes

Step 5: Code

- N pancakes
- How many panCapacity can remove?
  - N // panCapacity
- remainder
  - N % panCapacity
- Half of pan?
  - panCapacity / 2

Let’s code it up!

```python
def minutesNeeded(numCakes, capacity):
    full = numCakes // capacity
    left = numCakes % capacity
    minutes = 10 * full
    if left > capacity/2:
        minutes += 10
    else:
        minutes += 5
    return minutes
```
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]`

```python
["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 `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”?

```
Read word “vacuum”?
```

- What happens when we read a word?

Read word “vacuum”? Add 1 to `counts[2]`
Parallel Lists

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

\[
\text{["apple", "fox", "vacuum", "lime"]}
\quad \text{[5, 2, 26, 15]}
\]

• What happens when we read a word?
  
  Read word “cat”?  

Parallel Lists

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

\[
\text{["apple", "fox", "vacuum", "lime", "cat"]}
\quad \text{[5, 2, 26, 15, 1]}
\]

• What happens when we read a word?
  
  Read word “cat”?  

Calculate word most often in file

```python
6 def wordOccursTheMost(fname):
7     f = open(fname)
8     words = []
9     counts = []
10    for line in f:
11        line = line.strip() #remove newline
12        data = line.split()
13        for word in data:
14            if word not in words:
15                words.append(word)
16                counts.append(1)
17            else: # update word
18                pos = words.index(word)
19                counts[pos] += 1
20     f.close()
```
Calculate word most often in file

```python
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()
```

How do you finish the function?

WOTO-2 Word Most Often

Calculate word most often in file

- words is list of all the words from the file
- counts is the count of each word in the file
- Find the largest count value: `maxcount = max(counts)`
- Find index location of largest count value: `maxpos = counts.index(maxcount)`
- Return word in same location: `return words[maxpos]`

Complete function:
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

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

- V is any variable: all list elements in order
- V_EXP is any expression, often use V

List Comprehension Examples

```python
print([n*2 for n in range(6)])
print([n for n in range(10) if n % 2 == 1])
```

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

print( [n*2 for n in range(6)] )

[0, 2, 4, 6, 8, 10]

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

[1, 3, 5, 7, 9]

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

[0, 1, 2, 3, 4]

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

['banana', 'pineapple']

WOTO-3 List Comprehension Examples

WOTO-3 List Comprehension Example

```
words = ['giraffe', 'zebra', 'ant', 'lion', 'elephant']
x = [2*x for x in [len(w) for w in words if len(w)>3] if x%2== 0]

words = ['giraffe', 'zebra', 'ant', 'lion', 'elephant']
y = [len(w) for w in words if len(w) > 3]  # Break it up to two list comprehensions
x = [2*x for x in y if x%2== 0]
```

```
y is [7, 5, 4, 8]  # Difficult to debug!!!
x is [8, 16]
```

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:
  The Notorious Jumping Frog of Calaveras County

• Run Transform.py on twain.txt

• Set as:
  doTransform("-nvw", Vowelizer.encrypt)
  #doTransform("-rvw", Vowelizer.decrypt)

• Results in new file: twain-nvw.txt
• First line of twain-nvw.txt is:
  Th Ntrs Jmpng Frg f Clvrs Cnty

Transform – Get vowels back?

• First line of twain-nvw.txt:
  Th Ntrs Jmpng Frg f Clvrs Cnty

• Run Transform.py on twain-nvw.txt

• Set as:
  #doTransform("-nvw", Vowelizer.encrypt)
  doTransform("-rvw", Vowelizer.decrypt)

• Results in new file: twain-nvw-rvw.txt
• First line of twain-nvw-rvw.txt is:
  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