Compsci 101
List Comprehensions, Parallel Lists

Susan Rodger
Feb 21, 2023

K is for …

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

• Keyboard - QWERTY or DVORAK
  • DVORAK:
    ![DVORAK Keyboard]
  • 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”

Announcements

• APT-3 due Thursday
• Assign 3-Transform out today, due Thursday, March 2
  • Sakai quiz Assign3 – Due Tues, Feb 28 (no grace day)
• Lab 6 Friday - Prelab out today
• APT Quiz 1 is Thurs Feb 23-Mon Feb 27
  • Take during this time
  • Two parts – each part has two APTs
  • Each part is timed

“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”
APT Quiz 1

Is your own work!
- No collaboration with others!
- Use your notes, lecture notes, your code, textbook
- DO NOT search for answers! No Chat-GPT!
- 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 Quiz 1 Feb 23-27

- Opens 2/23 1pm
- Closes at 11pm 2/27 – must finish all by this time
- There are two parts based on APTs 1-3
  - Each part has two APT problems
  - Each part is 2 hours – more if you get accommodations
  - Each part starts in Sakai under tests and quizzes
  - Sakai is a starting point with countdown timer that sends you to a new apt page just for each part
  - Could do each part on different day or same days
- Old APT Quiz so you can practice (not for credit) – on APT Page
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.

```python
filename: Pancakes.py

def minutesNeeded (numCakes, capacity):
    *** return integer representing time to cook pancakes based on integer parameters as described below ***
```
### 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

- Third 5 minutes
  - 1 done
  - 2 almost cooked

How many minutes to cook all three pancakes?

15 minutes!

### 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.

- 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>?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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 \(1 \leq 4\)? # Yes, warmer trick will work!
- Total: 25 minutes
  - 10 minutes for 8 pancakes +
  - 5 minutes for 8 pancakes +
  - Take 1 out, start 17th 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>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>25</td>
<td>25</td>
<td>25</td>
<td>25</td>
</tr>
</tbody>
</table>

Step 3: Generalize

- \(13 - 8 = 5\)
- \(8/2 = 4\)
- Is \(5 \leq 4\)?
  - No, warmer trick won’t work
- 10 minutes for 8 pancakes + 10 minutes for 5 more pancakes = 20 minutes

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

- 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
- remainder = 8
- Yes, use warmer
- Total: 25 minutes

Step 5: Code

- 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

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

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

Let’s code it up!

Very Close!
Has a slight bug in it!
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

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

- What happens when we read a word?

  - Read word “vacuum”?

  - 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”?

Add into words

Expand counts

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", "cat"]
[  5,    2,    26,    15,     1  ]

• What happens when we read a word?

Read word “cat”?

Calculate word most often in file

```python
def wordOcursTheMost(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()
```
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**


Complete function:

```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
    maxcount = max(counts)
    maxpos = counts.index(maxcount)
    return words[maxpos]
```
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]))
```

```
return [n for n in nums if n > 0]
```

- **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_EXP** is any expression, often use **V**

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

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

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

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

**List Comprehension Examples**

```
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 Example

```python
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]
x = [2*x for x in y if x%2== 0]
```

Don’t do this!!!

Break it up to two list comprehensions

```
y is  [7, 5, 4, 8]
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`:
  
  1. 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:
  
  1. Th Ntrs Jmpng Frg f Clvrs Cnty

Transform – Get vowels back?

- First line of `twain-nvw.txt`:

  1. 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:

  1. oath antares jumping fargo fe cleavers county

Transform – Vowels summary

- First line in `twain.txt`

  1. The Notorious Jumping Frog of Calaveras County

- After removing vowels – “encrypt”

  1. Th Ntrs Jmpng Frg f Clvrs Cnty

- After trying to re-vowelize – “decrypt”

  1. oath antares jumping fargo fe cleavers county