CS134:
Nested Lists & Comprehensions
Announcements & Logistics

- **Homework 4** is out on GLOW, due Monday at 10 pm
- **Lab 4** will be released today: has two parts!
  - Part 1 is due Wed/Thur (Oct 5/6) at 10 pm
  - Part 2 is due the following Wed/Thur (Oct 12/13) at 10 pm (after reading days!)
- **Final exam**: **Friday Dec 16 at 9:30 am**
- **Midterm exam**: **Thur Oct 20** evening exam (more details forthcoming regarding format)
  - Time Option 1: **6 pm - 7:30 pm in Wege (TCL 123)**
  - Time Option 2: **8 pm - 9:30 pm in Wege (TCL 123)**
  - TCL 206 for reduced distractions/extra time
  - Let us know asap if you have any class conflicts or need additional accommodations
  - Extra time accommodations should plan to start at 6pm if possible
Last Time

• Discussed **file reading** using lists and strings
  • Used string methods `.strip()`, `.split()`
  • Used list methods `.append()`, `.extend()`, `.count()`
• Learned about **ranges** (another sequence in Python)

```python
# simple for loop that prints numbers 1-10
for i in range(1, 11):
    print(i)
```
Today’s Plan

• Learn about list comprehensions as a way to simplify list accumulations
  • Leads to simpler, more succinct code
• Begin exploring lists of lists
• Use our knowledge about lists and loops to analyze interesting properties of our student data
  • Help prepare for Lab 4
List Comprehensions
List Patterns: Map & Filter

• When using lists and loops, there are common patterns that appear

• **Mapping:** Iterate over a list and return a new list that results from *performing an operation on each element* of original list
  • E.g., take a list of integers `numList` and return a new list which contains the square of each number in `numList`

• **Filtering:** Iterate over a list and return a new list that results from *keeping only elements of the original list that satisfy some condition*
  • E.g., take a list of integers `numList` and return a new list which contains only the even numbers in `numList`

• Python allows us to implement these patterns succinctly using *list comprehensions*
List Comprehensions

**Mapping List Comprehension** (perform operation on each element)

```
newList = [expression for item in sequence]
```

**Filtering List Comprehension** (only keep some elements)

```
newList = [item for item in sequence if conditional]
```

- Important points:
  - List comprehensions always start with an **expression** (even a variable name like “item” is an expression!)
  - We **never use append( )** inside of list comprehensions
  - We can **combine mapping and filtering** into a single list comprehension:
Dissecting List Comprehensions

Task: Extract even numbers from a range and create a list of their squares.

Using a list comprehension:

```python
result = [n**2 for n in range(10) if n%2 == 0]
```

All list comprehensions can be rewritten using a for loop!
Using List Comprehensions

- **List comprehensions** are convenient when working with files
- Recall our list of student names from before

```
students
['RJ Acosta',
 'Jackson C. Adelman',
 'Harris Agha',
 'Nick R. Alcock',
```
- Example: How can we find the list of student names that begin with a vowel? (Hint: we'll use our `isVowel()` function again)
  - Idea:
    - Iterate over students (list of strings)
    - For each name in list, check if first letter is a vowel
    - If it is, add name to result list
Using List Comprehensions

- **List comprehensions** are convenient when working with files
- Recall our list of student names from before

```python
students

['RJ Acosta',
 'Jackson C. Adelman',
 'Harris Agha',
 'Nick R. Alcock',

vowelNames = []
for name in students:
    if isVowel(name[0]):
        vowelNames.append(name)
```
Using List Comprehensions

- **List comprehensions** are convenient when working with files
- Recall our list of student names from before

```
students
[
'RJ Acosta',
'Jackson C. Adelman',
'Harris Agha',
'Nick R. Alcock',
]
```

- Example: How can we find the list of student names that begin with a vowel? (Hint: we'll use our `isVowel()` function again)

```python
vowelNames = []
for name in students:
    if isVowel(name[0]):
        vowelNames.append(name)
```
Using List Comprehensions

- **List comprehensions** are convenient when working with files.
- Recall our list of student names from before:

  ```
  students
  ['RJ Acosta',
   'Jackson C. Adelman',
   'Harris Agha',
   'Nick R. Alcock',
  ]
  ```

- Example: How can we find the list of student names that begin with a vowel? (Hint: we'll use our `isVowel()` function again):

  ```
  vowelNames = []
  for name in students:
      if isVowel(name[0]):
          vowelNames.append(name)
  ```

  ```
  vowelNames = [name for name in students if isVowel(name[0])]
  vowelNames
  ```
Using List Comprehensions

- **List comprehensions** are convenient when working with files
- Recall our list of student names from before

```python
students = ['RJ Acosta', 'Jackson C. Adelman', 'Harris Agha', 'Nick R. Alcock',
            'Emir C. Atli', 'Anjali K. Bhatia', 'Alex W. Choi', 'Ethan Cooper',
            'Edith N. Edwards-Mizel', 'Amir H. Estejab', 'Arden N. Fluehr',
            'Arden N. Fluehr']

vowelNames = [name for name in students if isVowel(name[0])]
print(vowelNames)
```

- Example: How can we find the list of student names that begin with a vowel? (Hint: we’ll use our `isVowel()` function again)
Lists of Lists
Lists of Lists!

- We have already seen lists of strings
- We can also have **lists of lists** (sometimes called a two-dimensional list)!
- Often arise when using list comprehensions
- Suppose we have a **list of lists of strings** called `myList`
- `word = myList[a][b]` (# word is a string)
  - `a` is index into “outer” list (identifies **which inner list** we want)
  - `b` is index into “inner” list (identifies **which element** within the inner list)

```python
def main():
    myList = [['cat', 'frog'], ['dog', 'toad'], ['cow', 'duck']]
    word = myList[1][0]  # 'dog'
```

myList = [['cat', 'frog'], ['dog', 'toad'], ['cow', 'duck']]
We Don’t Talk About Bruno Data Types

• Python is a loosely typed programming language

• We don’t explicitly declare data types of variables

• But like Bruno, the creepy uncle in *Encanto* who lurks behind the walls and predicts the future, data types are always there

• It’s important to make sure we pay attention to what a function expects, especially with lists and strings! (remember this in Lab 4)

• **Lists of lists of strings versus list of strings:**

```python
myList = [ ['cat', 'frog'], [ 'dog', 'toad'], [ 'cow', 'duck'] ]
myList[1][0] is 'dog'

myList = [ 'cat', 'frog', ['dog', 'toad'], ['cow', 'duck'] ]
myList[1][0] is 'f'
```
Lists of Lists and Comprehensions

- Suppose we want to create a list of lists of strings using our student data.

```python
filename = "classnames.csv"
allStudents = []
with open(filename) as roster:
    for student in roster:
        allStudents.append(student.strip().split(','))
```

Acosta,RJ,26,rja3
Agha,Harris,25,hha1
Alcock,Nick R.,25,nra2
Atli,Emir C.,26,eca2
Chang,Daniel Y.,25,dy3c1
Durham,Keelan S.,25,khd2
Felten,Timothy E.,26,tf2
Gwilt,Kyle E.,25,kc1
Hartman,Sarah A.,25,sah4
Howard–Sarin,Brij C.,26,bch6
Jiang,Weiran,26,wj4
Joy,Matt L.,26,mlj2
Keyes,Mikey A.,26,mak5
Kubomiya,Reona,26,rk20
Lee,Gabe,26,gjl1
Lee,Yuri J.,26,yjl1
Nguyen,Trung Nguyen T.,26,ttn2
Lists of Lists and Comprehensions

- Suppose we want to create a list of lists of strings using our student data.

```python
filename = 'csv/classnames.csv'
with open(filename) as roster:
    for student in roster:
        allStudents.append(student.strip().split(','))

# with a list comprehension!
filename = 'csv/classnames.csv'
with open(filename) as roster:
    allStudents = [student.strip().split(',') for student in roster]
```

```
allStudents # list of lists of strings
[['Acosta', 'RJ', '26', 'rja3'],
 ['Agha', 'Harris', '25', 'hha1'],
 ['Alcock', 'Nick R.', '25', 'nra2'],
 ['Atli', 'Emir C.', '26', 'eca2'],
 ['Chang', 'Daniel Y.', '25', 'dyc1'],
 ['Durham', 'Keelan S.', '25', 'ksd2'],
 ['Felten', 'Timothy E.', '26', 'tef2'],
 ['Gwilt', 'Kyle E.', '25', 'kg15'],
 ['Hartman', 'Sarah A.', '25', 'sah4']]
```

classnames.csv

list of lists of strings
More List Comprehensions

```
allStudents:
[['Acosta', 'RJ', '26', 'rja3'],
 ['Agha', 'Harris', '25', 'hha1'],
 ['Alcock', 'Nick R.', '25', 'nra2'],

• Generate list of only last names using allStudents

```python
# generate list of only student last names
lastNames = [s[0] for s in allStudents]
lastNames

['Acosta',
 'Agha',
 'Alcock',
 'Atli',
 'Chang',

• Generate list of only first names

```python
# List comprehension to generate a list of first names
# (without middle initial)
firstNames = [s[1].split()[0] for s in allStudents]
firstNames

['RJ',
 'Harris',
 'Nick',
 'Emir',
 'Daniel',

split() first name, return first element (effectively removes middle initial)```
Exercise:
Student Fun Facts
Exercise: Student Fun Facts!

• Write a function `characterList` which takes in two arguments `rosterList` (list of lists of strings) and `character` (a string) and returns the list of students in the class whose first name starts with character.

• Can we do this with a list comprehension?

```python
def characterList(rosterList, character):
    """Takes the student info as a list of lists and a string character and returns a list of students whose first name starts with character""
```
Exercise: Student Fun Facts!

• Write a function `characterList` which takes in two arguments `rosterList` (list of lists of strings) and `character` (a string) and returns the list of students in the class whose first name starts with character.

• Can we do this with a list comprehension?

```python
def characterList(rosterList, character):
    """ Takes the student info as a list of lists and a string character and returns a list of students whose first name starts with character""
    return [name[1] for name in rosterList if name[1][0] == character]
```

call = characterList(allStudents, 'B')

['Brij C.', 'Betsy']
Exercise: Student Fun Facts!

- Write a function `yearList` which takes in two arguments, `rosterList` (list of lists of strings) and `year` (int) and returns the list of students in the class with that graduating year.

```python
def yearList(rosterList, year):
    """Takes the student info as a list of lists and a year (22-26) and returns a list of students graduating that year""
    return [name[1]+" "+name[0] for name in rosterList if name[2] == str(year)]

seniors = yearList(allStudents, 23)
seniors
['Min Kyu Park',
'Matthew L. Phang',
'Jennifer R. Sarmiento',
'Patrick Izidro',
'Sameer Jain',
'Tiffany J. Park',
'Matt Wisotsky',
'Grace A. Clarke',
'Ethan Cooper']
```
Exercise: Student Fun Facts!

- Write a function `mostVowels` that can be used to compute the list of students with the most vowels in their first name. (Hint: use `countVowels()`)

```python
def mostVowels(wordList):
    ''' Takes a list of strings wordList and returns a list of strings from wordList that contain the most # vowels'''
```

- General strategy for finding max in list of lists?
  - Initialize a max value BEFORE the loop to a very small number
  - If you see a value bigger than max, update max
Exercise: Student Fun Facts!

- Write a function `mostVowels` that can be used to compute the list of students with the most vowels in their first name. (Hint: use `countVowels()`.)

```python
def mostVowels(wordList):
    ''' Takes a list of strings wordList and returns a list of strings from wordList that contain the most # vowels '''

    maxSoFar = 0  # initialize counter
    result = []

    for word in wordList:
        count = countVowels(word)
        if count > maxSoFar:
            # update: found a better word
            maxSoFar = count
            result = [word]

        elif count == maxSoFar:
            result.append(word)

    return result

# which student(s) has most vowels in their name?
mostVowelNames = mostVowels(firstNames)
mostVowelNames

['Genevieve', 'Maximilian']
```
Exercise: Student Fun Facts!

- Write a function `leastVowels` that can be used to compute the list of students with the least vowels in their first name. (Hint: use `countVowels()`.)

```python
def leastVowels(wordList):
    ''' Takes a list of strings wordList and returns a list
    of strings in wordList that contain the least number of vowels''
    minSoFar = len(wordList[0])  # initialize counter
    result = []
    for word in wordList:
        count = countVowels(word)
        if count < minSoFar:
            # update: found a better word
            minSoFar = count
            result = [word]
        elif count == minSoFar:
            result.append(word)
    return result

leastVowels(firstNames)

['RJ', 'C.J.', 'M']
The end!

CS134: Nested Lists & Comprehensions