IbraFSG™ 3 - Week 7; List Comprehensions

Ibrahim Chehab

UTM RGASC

February 28, 2024

Table of Contents

Welcome back to IbraFSGs™

- Welcome back to IbraFSGs™! Hello to new people and welcome back to tenured members.
- This week we will be discussing *list comprehensions*.
- List Comprehensions are a way to create lists in Python using very little code. They are a very powerful tool that you will use in the future (A2)

Quick House Keeping Notes: Would you rather IbraFSGs™ continue to provide harder, "worst-case-scenario" midterm-style problems, or would you like to see more "real-world" problems?
Today I've tried to mix the two together, but I'd like to know your thoughts.

A Recap of the UltraSheet™

- An *UltraSheet™* is a "cheat sheet" that you compile for **yourself** to review course materials
  - Sharing UltraSheets™ is **counter-productive** and **will not help you learn the material**
  - However, reviewing content in a group and simultaneously updating your UltraSheets™ is a good idea
- It acts like your own personalized textbook chapter
  - It allows you to **regurgitate all the course information in a contigous, organized manner** and helps you **find gaps in your knowledge**
- UltraSheets™ help with type 1 and 2 questions
  - Can you remember what type 1 and 2 questions are?
  - Can you remember what a *type 3* question is? :troll:

Midterm Debrief

How did the midterm go! In parciular:

- Which course content did the midterm emphasize?
- Which types of questions did you notice were emphasized? (Type I, II, III)
- Which studying tactics worked? Which ones didn't?
- What will you do differently for next time?

Fun fact: In *CSC236* you prove the correctness of functions using *induction.* Aka: You don't need to make test cases, you know for a fact it works no matter what.

Key Terms

- **List Comprehension:** A way to create lists in Python using very little code.
- **Expression:** A piece of code that returns a value.
- **Iterable:** An object that can be iterated over.

...Yeah that's about it for this week. **Note:** Your UltraSheets™ should be filled with examples and explanations of these terms, and specifically how they relate to List Comprehensions.

A Trick to Understanding List Comprehensions

List Comprehensions may seem scary at first, but they are actually quite simple. The trick to understanding them is to relate them to MAT102: Example:

$\{x^2 : x \in \{0, 1, 2, 3, 4, 5\}\}$ $\Longleftrightarrow$

```
[x**2 for x in range(6)]
```

Fun Fact: You can include conditionals in list comprehensions, and items to be added if the condition is not met! Example: What would the following list comprehension return?

```
[x**2 if x % 2 == 0 else x**3 for x in range(6)]
```

Cool Application for List Comprehensions

List Comprehensions are a very quick way to return *counts* of something, and to filter for a certain attribute. For instance (fill in the blanks):

```
# Count the number of songs I've liked:
____[song for song in songs _______]

# filter for songs i've liked
[song for song in songs __________]
```

Practice Problem I: Loop2LstComp

Nugget is working on clearing out their Spotify Playlist. They're working on making some list comprehensions to filter out some songs. Help them convert the following for-loops into list comprehensions:

```
# Q1:
lst = []
for i in songs: # songs is an arbitrary list of Song
    objects; implementation is irrelevant for this
    question
  lst.append(i.title)
# Q2:
filter = []
for song in songs:
    if song.is_liked():
        filter.append(song)
```

Practice Problem I: Loop2LstComp (hard mode)

```
# Q3 (Blame Nugget for this, not me):
filter = []
for song in songs:
  if song.genre == "Rock":
    filter.append("Rock on!")
  elif song.genre == "Pop":
    filter.append("Pop on!")
  elif song.genre == "Love":
    filter.append("No Swifties Allowed")
  else:
    filter.append("euuuh, brother euuuh")
```

Hint: You can't use `elif` in a List Comprehension, but you can use `if` and `else`. How can we replicate the `elif` functionality using only `if` and `else`?
Hint 2: Start by eliminating `elif` in the original problem
Hint 3: Think of the chain rule from MAT135

Practice Problem II: This ain't Build A ForLoop

Bella Poarch is working on her next hit single, *Build A ForLoop*. She's trying to create some list comprehensions, but since making list comprehensions isn't smiling on TikTok she's struggling. Help her fill in the following list comprehensions:

- **Q1:** Create a list comprehension that returns the first 10 even numbers
- **Q2:** Create a list comprehension that returns the first 10 odd numbers
- **Q3:** Create a list comprehension that flattens a 2D list

Example: `[[1, 2, 3], [4, 5, 6], [7, 8, 9]]` $\rightarrow$ `[1, 2, 3, 4, 5, 6, 7, 8, 9]`
Hint: An Iterable can contain an expression :thonk:

A final challenge...

Recall from last week our challenge problem:
Sharon Goodwin delves into Python in her free time. She's trying to create a series of recursive functions that mutually recurse over each other to determine whether a positive integer is even or odd. Help her create these two functions.

RESTRICTIONS:

- You are **NOT** allowed to use **ANY** of Python's integer operations **EXCEPT** subtraction.
- You may **NOT** use **Modulo**.
- Each function must have **EXACTLY** one base-case.
- You **MUST** use mutual recursion.
- You **MAY NOT** use any helper methods.

A final challenge...

Continue solving this problem today:

```
def is_even(num: int) -> bool:
    """
    Method which uses mutual recursion to determine
    whether an integer is even or odd.
    """
    # TODO: Implement this method with a recursive
    call to is_odd


def is_odd(num: int) -> bool:
    """
    Method which uses mutual recursion to determine
    whether an integer is even or odd.
    """
      # TODO: Implement this method with a recursive
    call to is_even
```