Introduction to Computer Science I - New SME Assignment Solution

Niranjana Srinivasa Ragavan

March 2025

In this document, I have written the explanations for the questions and solutions provided in the given assignment document. I have used PythonTex package to execute the python code written within the Latex. My explanations are based on the assumption of prerequisite knowledge specified in the assignment.

KP1. Understanding Nested List Comprehensions

G1.1 (E) Predicting the Output of a Nested List Comprehension

Question

Consider the following code snippet

```
nested_list = [[(i + j) ** 2 for j in range(1, 4)] for i in range(1, 4)]
print(nested_list)
```

What will be printed?

Solution

```
Output: [[4, 9, 16], [9, 16, 25], [16, 25, 36]]
```

Explanation

Recall that nested list comprehension consists of list comprehensions within a list comprehension. Expression inside `[]` is called list comprehension.

- In this code, the outer list comprehension `for i in range(1, 4)` which generates values $i = 1, 2, 3$.
- For each value of `i`, the inner list comprehension `for j in range(1, 4)` generates values $j = 1, 2, 3$.
- For each pair of `i` and `j`, the expression `(i + j) ** 2` is computed.
- For each iteration of `i`, the computed result is stored as a sublist and the final result is the list of all these sublists.
- **Step by step evaluation**

```
i j (i+j)**2
1 1 4
1 2 9
1 3 16
Sublist for i = 1: [4, 9, 16]
i j (i+j)**2
2 1 9
2 2 16
2 3 25
Sublist for i = 2: [9, 16, 25]
i j (i+j)**2
3 1 16
3 2 25
3 3 36
Sublist for i = 3: [16, 25, 36]
```

- **Scope of the variables:**
  - In python, in the nested `for` loop, the inner loop has the same scope as that of the enclosing scope. Thus, outer loop variables defined before the inner loop can be directly referenced by the inner loop.

- While in nested list comprehension, each list comprehension creates its own local scope.
- Still, the inner list comprehension can access the variables local to the outer comprehension. This is due to the property called **Closures**. The inner list comprehension can access the variables from the enclosing scope.
- In this code, the non-local variable `i` which is in the scope of outer comprehension, can be accessed as a free variable by inner list comprehension. This means, `i` is not bound to the inner comprehension, but is stored in a closure. This repeats for each iteration of `i`.
- This can be evident from this line of byte-level code `LOAD_CLOSURE 1 (i)`.

G1.2 (M) Converting a `for` Loop to a Nested List Comprehension

Question

Rewrite the following `for` loop as a nested list comprehension:

```
matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
new_matrix = []
for row in matrix:
    new_row = []
    for num in row:
        new_row.append(num + 2)
    new_matrix.append(new_row)
```

Solution

```
new_matrix = [[num + 2 for num in row] for row in matrix]
```

Explanation

- **Explanation of nested `for` loops:** The outer `for` loop iterates through each row in the matrix. The inner loop iterates through each number in the row and creates a new row by appending 2 to each number in the original row. The new rows are appended to create the new matrix.
- **Creating list comprehension:** Replace the outer loop with `[..for row in matrix]`. In the inner loop, for each `num` in row, `num + 2` expression should be executed. Replace the inner loop with `[num + 2 for num in row]`.
- Finally, all the modified rows are appended to create `new_matrix`. The list comprehension implicitly builds the list and there is no need for `append` function.

KP2. Using Nested List Comprehensions with Strings

G2.1 (E) Predicting Output for Uppercasing Nested Words

Question

What will be printed when the following code is run?

```
sentence = "I am happy"
words = [[char.upper() for char in word] for word in sentence.split()]
print(words)
```

Solution

```
Output: [['I'], ['A', 'M'], ['H', 'A', 'P', 'P', 'Y']]
```

Explanation

- Method `sentence.split()` splits the sentence and creates a **list** of individual words.
- Methods `char.upper()` turns lowercase character into an uppercase character.
- **Outer comprehension:** It iterates through each `word` in the list of individual words created by `.split()` method.
- **Inner comprehension:** It iterates through each `char` in the `word` and converts the character into uppercase using `.upper()` method. If the `char` is already in uppercase, it does nothing. For each word, the list of its uppercase characters is created.
- The final list is the collection of sublists of uppercase letters of individual words.
- **Step by step evaluation**

Words List: ['I', 'am', 'happy']
word char upper
I I I
Sublist 'I': ['I']
word char upper
am a A
am m M
Sublist 'am': ['A', 'M']
word char upper
happy h H
happy a A
happy p P
happy p P
happy y Y
Sublist 'happy': ['H', 'A', 'P', 'P', 'Y']

KP3. Using Nested List Comprehensions with Conditions

G3.1 (E) Creating a Nested List of Filtered Words

Question

What is the output of the following code snippet?

```
words = [["cat", "elephant"], ["dog", "tiger"], ["fox", "giraffe"]]
long_words = [[word for word in row if len(word) > 3] for row in words]
print(long_words)
```

Solution

```
Output: [['elephant'], ['tiger'], ['giraffe']]
```

Explanation

KP4. Writing Code Using Nested List Comprehensions

G4.1 Writing Code for Creating Given Nested List

Question

Write a Python program using a nested list comprehension to create a 3 x 4 grid filled with zeros and print it.

Solution

```
grid = [[0 for _ in range(4)] for _ in range(3)]
print(grid)
```

Explanation

G4.2 Writing Code for Creating Given Nested List from Strings

Question

Write a nested list comprehension that extracts only vowels from each word in a sentence, storing them in nested lists. For `sentence = "Python Is Amazing"`, the output must be `[['o'], ['I'], ['A', 'a', 'i']]`

Solution

```
sentence = "Python Is Amazing"
vowels = "aeiouAEIOU"
vowel_list = [[char for char in word if char in vowels] for word in sentence.split()]
print(vowel_list)
```

Explanation

G4.3 Writing Code for Creating Given Nested List with Conditionals

Question

Write a nested list comprehension that creates a 5×5 grid, but fills it with 1 if the sum of row and column indices is even, and 0 otherwise, and print it.

Solution

```
grid = [[1 if (i + j) % 2 == 0 else 0 for j in range(5)] for i in range(5)]
print(grid)
```

Explanation