Programming with Python Feb 21 2025

Brief Summary of contents discussed.

List: A list in Python is a comma separated, **ordered**, **mutable** collection of elements. Lists allow storing multiple data types in a single variable.

List Comprehension: List comprehension is a concise way to create lists in Python using a single line of code.

```
[expression for item in iterable if condition]
```

Nested list (lists within list):

```
lst = [[2, 3, 11], [5, 13], [7, 23, 29, 17], [19]]

print(lst[1])
print(list[-1])
print(lst[0][1])
print(lst[2][3])
print(list[3][1]) #Error, index out of range
```

Example 1: Write a python function to flatten a nested list with and without extend.

Approach 1:

```
def flatten_list(nested_list):
    flat_list = []
    for sublist in nested_list:
        for element in sublist:
            flat_list.append(element)
    return flat_list
```

Approach 2 (using list extend function):

```
def flatten_list(nested_list):
    flat_list = []
    for sublist in nested_list:
        flat_list.extend(sublist)
    return flat_list
```

Approach 3: Use List Comprehension (#TODO)

Example 2: Write a function that takes a sentence as input, splits it into words, reverses the order of words, and returns the reversed sentence.

```
def rev_sentence(sentence):
    words = sentence.split()
    words.reverse()  # Reverse the list in-place
    return " ".join(words)
```

Note: For reversing try other approaches as well as discussed in class (such string reversal using `[::-1]` using slicing syntax or string concatenation)

Example 3: Write a function that extracts hashtags from a given post.

```
    def extract_hashtags(post):
        words = post.split()
        hashtags = []
        for word in words:
            if word[0] == '#':
                hashtags.append(word[1:])
        print(hastags)
#Note: Also try this using List Comprehension.
```

List Comprehension Example:

1. **Squares**

```
squares = [x**2 for x in range(1, 6)]
print(squares)
```

2. **Words to lowercase**

```
words = ["hello", "world", "python"]

uppercase_words = [word.lower() for word in words]

print(uppercase_words)
```

3. **Flatten a nested_lis**

```
nested_list = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
flat_list = [element for lst in nested_list for
element in lst]
print(flat_list)
```

Exercise

1. Write a function that converts a given sentence into an acronym (taking the first letter of each word).

```
sentence = "machine learning engineer"
print(acronym(sentence)) # Output: "MLE"
```

2. (Append vs Extend) Write a function that takes a list of words and creates two new lists: (Ask for output)

- One list where each word is appended one by one.
- Another list where each word is extended as a list of characters.

```
Example List of words: 'apple', 'banana'
Output 1: ['apple', 'banana']
Output 2: ['a', 'p', 'p', 'l', 'e', 'b', 'a', 'n',
'a', 'n', 'a']
```

3. `(Use List Comprehension)` Write a function that extracts hashtags from a given post.

4. Given a list of email addresses that might have extra spaces or uppercase letters, write a function to clean them.

5. Write a function that merges two lists of names, removes duplicates, and returns a sorted list.Use Case: Merging contact lists while filtering duplicates.
6. Flatten lists such as `[[1, 2], 3, [4, 5, [6, 7]], 8]`

Commonly used List functions:

Function	Description	Example	Output
`len(list)`	Returns the number of elements in the list	`len([1, 2, 3, 4])`	`4`
`min(list)`	Returns the smallest element in the list	`min([3, 1, 4, 2])`	`1`
`max(list)`	Returns the largest element in the list	`max([3, 1, 4, 2])`	`4`
`append(x)`	Adds x to the end of the list	`lst = [1, 2]; lst.append(3)`	`[1, 2, 3]`
`extend(iterable)`	Adds elements from another iterable to the list	`lst = [1, 2]; lst.extend([3, 4])`	`[1, 2, 3, 4]`
`count(x)`	Returns the number of times x appears	`[1, 2, 2, 3].count(2)`	`2`
`remove(x)`	Removes the first occurrence of x	`lst = [1, 2, 3]; lst.remove(2)`	`[1, 3]`
`index(x)`	Returns the index of the first occurrence of x	`[10, 20, 30].index(20)`	`1`
`pop(i)`	Removes and returns the element at index i (default last)	`lst = [1, 2, 3]; lst.pop(1)`	`2 (List: [1, 3])`
`insert(i, x)`	Inserts x at index i	`lst = [1, 3]; lst.insert(1, 2)`	`[1, 2, 3]`
`reverse()`	Reverses the list in place	`lst = [1, 2, 3]; lst.reverse()`	`[3, 2, 1]`
`sort()`	Sorts the list in ascending order	`lst = [3, 1, 2]; lst.sort()`	`[1, 2, 3]`