LU02.L05 - Listen

Schritt 1: Verwenden von List Comprehension oder "map"-Funktion

Wir können diese Aufgabe mit einer Schleife, einer List Comprehension oder der map-Funktion lösen. Hier sind Beispiele für beide Ansätze:

Mit Schleife:

```
def increment_numbers(numbers):
    new_numbers = [] # create an empty list to store the new numbers
    for number in numbers:
        new_numbers.append(number + 1) # append the incremented value to the
new list
    return new_numbers
```

Diese Techniken werden wir im Verlauf des Kurses genauer kennenlernen.

Mit List Comprehension:

```
def increment_numbers(numbers):
    return [number + 1 for number in numbers]
```

Mit map-Funktion:

```
def increment_numbers(numbers):
    return list(map(lambda x: x + 1,
numbers))
```

Schritt 2: Rückgabe der neuen Liste

Die Funktion gibt die neue Liste zurück, wie in den vorherigen Schritten gezeigt.

Schritt 3: Testen der Funktion

```
if __name__ == '__main__':
    new_numbers = increment_numbers(numbers)
    print('Original numbers:', numbers)
    print('Incremented numbers:', new_numbers)
```

Schritt 4: Testen der Funktion

Lassen Sie die Pytests laufen.

Ausgabe

```
Original numbers: [1, 2, 3, 4, 5]
Incremented numbers: [2, 3, 4, 5, 6]
```

Zusammenfassung

Die Musterlösung zeigt zwei verschiedene Möglichkeiten, wie eine Funktion eine neue Liste erstellen kann, in der alle Elemente der Eingabeliste um eins erhöht wurden, ohne die ursprüngliche Liste zu verändern. Dies ist ein wichtiger Aspekt der funktionalen Programmierung und hilft dabei, den Code besser verständlich und vorhersagbar zu machen.
