Question 1

Build this list [0,0] two separate ways.

Answer for Question 1

Method 1:

list1 = [0, 0]

Method 2:

list2 = [0] * 2

print(list1) print(list2)

Question 2

Reassign 'hello' in this nested list to say 'goodbye' instead: list3 = [1,2,[3,4,'hello']]

In [7]:
```
# Answer for Question 2

list3 = [1,2,[3,4,'hello']]
list3[2][2] = 'goodbye'
print(list3)
```

```
[1, 2, [3, 4, 'goodbye']]
```

Question 3

Sort the list below: list4 = [5,3,4,6,1]

In [8]:
```
# Answer for Question 3

list4 = [5,3,4,6,1]
list4.sort()
print(list4)
```

```
[1, 3, 4, 5, 6]
```

Question 4 (Exercise 6.1)

Write a function called nested_sum that takes a list of lists of integers and adds up the elements from all of the nested lists. For example: t = [[1, 2], [3], [4, 5, 6]] nested_sum(t)

21

In [9]:
```
# Answer for Question 4

def nested_sum(t):
    return sum(sum(sublist) for sublist in t)

# Test the function
t = [[1, 2], [3], [4, 5, 6]]
result = nested_sum(t)
print(result)  # Should print 21
```

```
21
```

Question 5 (Exercise 6.2)

Write a function called cumsum that takes a list of numbers and returns the cumulative sum; that is, a new list where the ith element is the sum of the first i + 1 elements from the original list. For example: t = [1, 2, 3] cumsum(t)

[1, 3, 6]

In [11]:
```
# Answer for Question 5

def cumsum(t):
    result = []
    total = 0
    for num in t:
        total += num
        result.append(total)
    return result

# Test the function
t = [1, 2, 3]
result = cumsum(t)
print(result)  # Should print [1, 3, 6]
```

```
[1, 3, 6]
```

Question 6.4

Write a function called chop that takes a list, modifies it by removing the first and last elements, and returns None. For example:

t = [1, 2, 3, 4] chop(t) t [2, 3]

In [12]:
```
# Answer for Question 6.4

def chop(t):
    if len(t) >= 2:
        del t[0]
        del t[-1]
    else:
        t.clear()
    return None

# Test the function
t = [1, 2, 3, 4]
result = chop(t)
print(result)  # Should print None
print(t)       # Should print [2, 3]
```

```
None
[2, 3]
```

Question 6.5

Write a function called is_sorted that takes a list as a parameter and returns True if the list is sorted in ascending order and False otherwise. For example:

is_sorted([1, 2, 2]) True is_sorted(['b', 'a']) False

In [14]:
```
# Answer for Question 6.5

def is_sorted(t):
    return all(t[i] <= t[i+1] for i in range(len(t)-1))

# Test the function
print(is_sorted([1, 2, 2]))  # Should print True
print(is_sorted(['b', 'a']))  # Should print False
```

```
True
False
```

Question 1 (Exercise 6.6)

Two words are anagrams if you can rearrange the letters from one to spell the other. Write a function called is_anagram that takes two strings and returns True if they are anagrams.

In [15]:
```
# Answer for Question 1

def is_anagram(str1, str2):
    # Remove spaces and convert to lowercase
    str1 = str1.replace(" ", "").lower()
    str2 = str2.replace(" ", "").lower()

    # Check if the sorted strings are equal
    return sorted(str1) == sorted(str2)

# Test the function
print(is_anagram("listen", "silent"))  # Should return True
print(is_anagram("hello", "world"))    # Should return False
```

```
True
False
```

Question 2 (Exercise 6.7)

Write a function called has_duplicates that takes a list and returns True if there is any element that appears more than once. It should not modify the original list.

```
In [16]: # Answer for Question 2

def has_duplicates(lst):
    return len(lst) != len(set(lst))

# Test the function
print(has_duplicates([1, 2, 3, 4, 5]))  # Should return False
print(has_duplicates([1, 2, 3, 2, 4]))  # Should return True
```

```
False
True
```