x\_coord = [23, 53, 2, -12, 95, 103, 14, -5]

y\_coord = [677, 233, 405, 433, 905, 376, 432, 445]

z\_coord = [4, 16, -6, -42, 3, -6, 23, -1]

labels = ["F", "J", "A", "Q", "Y", "B", "W", "X"]

points = []

**for** point **in** zip(labels, x\_coord, y\_coord, z\_coord):

points.append("{}: {}, {}, {}".format(\*point))

**for** point **in** points:

print(point)

Notice here, the tuple was unpacked using \* in the format method. This can help make your code cleaner!

F: 23, 677, 4

J: 53, 233, 16

A: 2, 405, -6

Q: -12, 433, -42

Y: 95, 905, 3

B: 103, 376, -6

W: 14, 432, 23

X: -5, 445, -1

**Quiz Solution: Zip Lists to a Dictionary**

cast\_names = ["Barney", "Robin", "Ted", "Lily", "Marshall"]

cast\_heights = [72, 68, 72, 66, 76]

cast = dict(zip(cast\_names, cast\_heights))

print(cast)

The order of elements in this output may vary since dictionaries are unordered.

{'Lily': 66, 'Barney': 72, 'Marshall': 76, 'Ted': 72, 'Robin': 68}

**Quiz Solution: Unzip Tuples**

cast = (("Barney", 72), ("Robin", 68), ("Ted", 72), ("Lily", 66), ("Marshall", 76))

names, heights = zip(\*cast)

print(names)

print(heights)

('Barney', 'Robin', 'Ted', 'Lily', 'Marshall')

(72, 68, 72, 66, 76)

**Quiz Solution: Transpose with Zip**

data = ((0, 1, 2), (3, 4, 5), (6, 7, 8), (9, 10, 11))

data\_transpose = tuple(zip(\*data))

print(data\_transpose)

((0, 3, 6, 9), (1, 4, 7, 10), (2, 5, 8, 11))

**Quiz Solution: Enumerate**

cast = ["Barney Stinson", "Robin Scherbatsky", "Ted Mosby", "Lily Aldrin", "Marshall Eriksen"]

heights = [72, 68, 72, 66, 76]

**for** i, character **in** enumerate(cast):

cast[i] = character + " " + str(heights[i])

print(cast)

['Barney Stinson 72', 'Robin Scherbatsky 68', 'Ted Mosby 72', 'Lily Aldrin 66', 'Marshall Eriksen 76']

List Comprehensions - create lists really quickly and concisely with list comprehensions.

capitalized\_cities = []

**for** city **in** cities:

capitalized\_cities.append(city.title())

can be reduced to:

capitalized\_cities = [city.title() **for** city **in** cities]

List comprehensions allow us to create a list using a for loop in one step.

You create a list comprehension with brackets [], including an expression to evaluate for each element in an iterable. This list comprehension above calls city.title() for each element city in cities, to create each element in the new list, capitalized\_cities.

### Conditionals in List Comprehensions

You can also add conditionals to list comprehensions (listcomps). After the iterable, you can use the if keyword to check a condition in each iteration.

squares = [x\*\*2 **for** x **in** range(9) **if** x % 2 == 0]

The code above sets squares equal to the list [0, 4, 16, 36, 64], as x to the power of 2 is only evaluated if x is even. If you would like to add else, you have to move the conditionals to the beginning of the listcomp, right after the expression, like this.

squares = [x\*\*2 **if** x % 2 == 0 **else** x + 3 **for** x **in** range(9)]

**A. Create a dictionary that includes the count of Oscar nominations for each director in the nominations list.**

**B. Provide a dictionary with the count of Oscar wins for each director in the winners list.**

##### Here's the logic for my solution:

1. To solve this question, I use the .items method for dictionaries. Remember, the key in our nominated dictionary is a list of nominated directors. Think Compound Data Structures!
2. I know I need to create a dictionary where the key is a director and the value is the number of nominations.
3. But to get each director as a key, I will have to use two for loops.
4. First, to iterate through the nominated dictionary's value (which here is a list of nominations) .
5. But I have do to this again to iterate through each element (what I'm trying to get to - a nominated director) in the nominated list.
6. After that, if the director isn't yet in our dictionary, we give that director a count of one. If the director is in the dictionary, we increment that director's count by one.

nom\_count\_dict = {}

**for** year, list\_dir **in** nominated.items():

**for** director **in** list\_dir:

**if** director **not** **in** nom\_count\_dict:

nom\_count\_dict[director] = 1

**else**:

nom\_count\_dict[director] += 1

Question 1b

Provide a dictionary with the count of wins for each director.

##### Essentially, it is the same logic as above, with the other dictionary.

We could use the same approach as in question 1a and it would work fine, but I've provided a shorter alternative here. Instead of the last 4 lines as above, I've just written 1 line, by using the .get method. In this line, we find the director in the win\_count\_dict dictionary and return the value for that director (the number of times they've won). If they aren't in the dictionary, get returns 0 for that director. Then we increment that director's count by one.

win\_count\_dict = {}

**for** year, winnerlist **in** winners.items():

**for** winner **in** winnerlist:

win\_count\_dict[winner] = win\_count\_dict.**get**(winner, 0) + 1

**Provide a list with the name(s) of the director(s) with the most Oscar wins. We are asking for a list because there could be more than 1 director tied for the most Oscar wins.**

## Question 2

Provide a list with the name(s) of the director(s) with the most Oscar wins.

##### Here's the logic for my solution:

* To address this question, I will need to first create a dictionary with the number of wins by each winning director. For that I can use the code we wrote for Question 1b above.

*#FIRST PART OF SOLUTION*

win\_count\_dict = {}

**for** year, winnerlist **in** winners.items():

**for** winner **in** winnerlist:

win\_count\_dict[winner] = win\_count\_dict.**get**(winner, 0) + 1

1. This win\_count\_dict dictionary provides a dictionary with the win count for the directors. We will need this to then identify which key (here, director name) has the highest value (here, win count).
2. To perform this task, we use a variable highest\_count to keep track of the highest count of wins.
3. We iterate through the dictionary to see if the value for a key (i.e., wins for a director) is more than the highest count.
4. If it is, we replace that value as the highest\_count.
5. Plus we add that key (here, director name) to our list tracking the most\_win\_director.
6. Every time we come upon a value higher than the current highest\_count, we replace highest\_count with the new higher value, empty the most\_win\_director and replace it with the new key (i.e., director's name).

*#SECOND PART OF SOLUTION*

highest\_count = 0

most\_win\_director = []

**for** key, value **in** win\_count\_dict.items():

**if** value > highest\_count:

highest\_count = value

most\_win\_director.clear()

most\_win\_director.append(key)

**elif** value == highest\_count:

most\_win\_director.append(key)

**else**:

continue

**Here is an alternative compact solution** to replace the 12 lines above for the second part of the solution, using the built-in function max(), and a list comprehension with a condition:

*#ALTERNATIVE SECOND PART OF SOLUTION*

highest\_count = max(win\_count\_dict.values())

most\_win\_director = [key **for** key, value **in** win\_count\_dict.items() **if** value == highest\_count]