More Dictionaries and Lists
CS51 - Fall 2026

Practice Problem 1 - Dictionaries

Write a function `get_counts` that given a list called `data` of (possibly duplicate) items, it creates a dictionary where the key-value pairs are the list items and the frequencies with which they appear in the list.

Practice Problem 2 - Dictionaries and tuples

Write a function `get_most_frequent` that given a dictionary called `counts` where the key-value pairs are a list's items and the frequencies with which they appeared in that list, it returns an item-frequency tuple, where item is the one with the highest frequency in the list.

Practice Problem 3 - Nested lists

Given the following lists `bizarre_list = [47, 'cs51', ['Feta','Cheddar', 'Edam', 'Gouda']]` and `matrix = [[ 1, 2, 3], [ 4, 5, 6], [ 7, 8, 9], [10, 11, 12]]`

What will the following expressions evaluate to?

- `len(bizarre_list)`
- `'Feta' in bizarre_list`
- `['Feta','Cheddar', 'Edam', 'Gouda'] in bizarre_list`
- `matrix[1]`
- `matrix[2][1]`

Practice Problem 4 - More nested lists

What would the following code print?

```
a_list = [[4, [True, False], 6, 8], [888, 999]]
if a_list[0][1][0]:
    print(a_list[1][0])
else:
    print(a_list[1][1])
```

Practice Problem 5 - Writing code with nested lists

Define a function `nested_total` that takes a list of lists of integers and returns the sum of all the values. For example,

```
lst = [[1, 2], [3], [4, 5, 6]]
sum = nested_total(lst)
print(sum)
```

would print 21.

Practice Problem 6 - More code with nested lists

Define a function `nested_avg` that takes a list of lists of integers and returns a list with each sublist averaged. For example,

```
list = [[1, 2], [3], [4, 5, 6]]
lst_avg = nested_avg(lst)
print(lst_avg)
```

would print `[1.5, 3.0, 5.0]`.

Practice Problem 7 - List comprehension

What will the following code print?

```
courses = ['csci051', 'phys042', 'csci054', 'engl019']
cs_courses = [course for course in courses if 'csci' in course]
print(cs_courses)
```

Practice Problem 8 - More list comprehension

Consider the following code. What will `new_lst` evaluate to?

```
new_lst = []
for x in range(10):
    if x % 2 == 0:
        new_lst.append(x**2)
```

Use list comprehension to shorten it