

In 61B, we'll be learning the Java programming language, with an emphasis on the usage and creation of data structures. Let's kick off the class by reviewing how lists and maps (called dictionaries in Python) work. Below, write the requested Python functions. Use loops. **Do not use any list or dictionary comprehensions (we won't learn them in 61B).**

(a) `def evens(list_of_ints):`
 `"""Returns a copy of the list but keeping only the even numbers."""`
 `return_list = []`

(b) `def count_words(list_of_words):`
 `"""Returns a map from each word to its count."""`
 `counts = {}`

- (c) Below, we see the Java solution to these problems. Discuss with your group what interesting features you observe in the Java code. If you have any Java veterans in your group, grill them about the weirdness, or of course feel free to ask your TA.

```
public static List<Integer> evens(List<Integer> L) {
    List<Integer> list = new ArrayList<>();
    for (int num : L) {
        if (num % 2 == 0) {
            list.add(num);
        }
    }
    return list;
}

public static Map<String, Integer> countWords(List<String> words) {
    Map<String, Integer> map = new TreeMap<>();
    for (String word : words) {
        if (map.containsKey(word)) {
            map.put(word, map.get(word) + 1);
        } else {
            map.put(word, 1);
        }
    }
    return map;
}
```

- (a) Create a **Dog** class in python. A dog should have two properties: a name and a size. The dog class should have a method called **grow** that increases the dog's size by 1. If the user prints out a dog, it should print the name followed by "the size ", followed by the size, followed by the word " dog" For example, the code below should print "maya the size 1000 dog".

```
dogs = [Dog("maya", 1000), Dog("yipster", 5), Dog("scott", 25)]
print(dogs[0])

class Dog():
```

- (b) Below, we show the Java solution. As before, discuss with your group what you observe about the code.

```
import java.util.List;

public class Dog {
    public String name;
    public int size;

    public Dog(String n, int s) {
        name = n;
        size = s;
    }

    public void grow() {
        size += 1;
    }

    @Override
    public String toString() {
        return name + " the size " + size + " dog";
    }

    public static void main(String[] args) {
        List<Dog> dogs = List.of(new Dog("maya", 1000),
                                new Dog("yipster", 5),
                                new Dog("scott", 25));
        System.out.println(dogs.get(0));
    }
}
```

- (a) Using the example code from earlier in this worksheet, try to write a Java function below which returns the difference between the maximum and minimum item in a `List<Integer>`. You may assume the list has length at least 1. You can get the *i*th item of a `List` called `L` by calling `L.get(0)`. You can get the size of a list with `L.size()`. There are solutions that don't use either of these functions.

```
public static int maxMinDiff(List<Integer> L) {
```

- (b) Extra problem: Write a Java function that takes a `List<String>` and returns a map from each String to the list of Strings that immediately follow it (i.e. come right after it). For example, if the input list is `["I", "love", "java", "but", "I", "love", "python", "more"]`, then the output should be:

```
{
    "I": ["love", "love"],
    "love": ["java", "python"],
    "java": ["but"],
    "but": ["I"],
    "python": ["more"]
}
```

```
public static Map<String, List<String>> listOfFollowers(List<String> x) {
```