HOMEWORK 9

1. Using dictionaries as a counter

Problem 1. Write a function called word_count(sentence) that takes a sentence as input. The function should return a dictionary where the keys are the words in the sentence and the values represent the count of how many times each word appears. For this problem, we do not treat uppercase and lowercase as identical. For example

```
sentence = "Whoever has learned how to listen to trees no longer
   wants to be a tree"
word_count(sentence)
```

should return

```
{"Whoever": 1,
    "has": 1,
    "learned": 1,
    "how": 1,
    "to": 3,
    "listen": 1,
    "trees": 1,
    "no": 1,
    "longer": 1,
    "wants": 1,
    "be": 1,
    "a": 1,
    "tree": 1
}
```

Problem 2. Write a function named mail_count(alist) that takes a list of email strings as input. Each email is a string with the following format:

```
"From stephen.marquard@uct.ac.za Sat Jan 7"
```

The function should return a dictionary where the keys are days of the week, and the values represent the number of emails sent on each day. For example

```
emails = [
    "From stephen.marquard@uct.ac.za Sat Jan 7",
    "From louis@media.berkeley.edu Fri Jan 5",
    "From zqian@umich.edu Fri Jan 5",
```

```
    "From rjlowe@iupui.edu Thu Jan 4",
    "From cwen@iupui.edu Sat Jan 7"
]
print(mail_count(emails))
```

should return

```
{"Sat": 2, "Fri": 2, "Thu": 1}
```

For this problem, you should use the split method to get the day of the week from an email.

Problem 3. Write a function called major_count(d) that takes a dictionary as input. In this dictionary, the keys represent student names, and the values indicate their respective majors. The function should return a new dictionary where the keys are the majors and the values are the corresponding counts of students enrolled in each major. For example

```
d = {
    "Alice": "Biology",
    "Bob": "Mathematics",
    "Charlie": "Biology",
    "David": "Computer Science",
    "Eva": "Mathematics",
    "Frank": "Computer Science",
}
major_count(d)
```

should return

```
{
    "Biology": 2,
    "Mathematics": 2,
    "Computer Science": 2
}
```

Problem 4. Write a function named most_frequent(alist) that accepts a list of integers as input and returns the number that occurs most frequently. If there are ties for the most frequent number, return the largest one among them.

```
alist = [1, 3, 2, 3, 4, 2, 5, 3, 2]
most_frequent(alist)
```

should return 3. Note that both 3 and 2 appear 3 times. We pick 3 because it is the bigger number.

Problem 5. Write a function named election_result(votes) that takes a list of candidate names as input and returns the winner of the election. If a candidate receives at least 50% of the votes, that candidate is declared the winner. If no candidate meets this requirement, the function should indicate that a reelection is necessary. For example

```
votes = ["Alice", "Bob", "Alice", "Charlie", "Bob", "Alice", "
   Alice"]
election_result(votes)
```

should return "Alice". On the other hand

```
votes = ["Alice", "Bob", "Alice", "Charlie", "Bob", "Alice", "
   Charlie"]
election_result(votes)
```

should return

`"Reelection"`.

Problem 6. Write a function named count_digits(n) that takes an integer n as input and returns a dictionary where the keys are the digits (0-9) and the values are the counts of how many times each digit appears in n. For example

```
n = 1122334455
count_digit(n)
```

should return

```
{
    1: 2,
    2: 2,
    3: 2,
    4: 2,
    5: 2
}
```

For this problem, you might want to convert an integer to a string and vice versa. This is not absolutely necessary but it will make the problem a bit easier.

2. Class

Problem 7. Write a class Rectangle that represents a rectangle. Each Rectangle object should have the following attributes.

- length: the length of the rectangle (a float).
- width: the width of the rectangle (a float).

Add the following methods to your class

- area(): Returns the area of the rectangle.
- perimeter(): Returns the perimeter of the rectangle.
- resize(new_length, new_width): Updates the rectangle's dimensions to the new length and width.

Problem 8. Write a class College that represents a college. Each College object should have the following attributes:

- name: the name of the college (a string).
- location: the location of the college (a string).
- established_year: the year the college was established (an integer).
- total_students: the number of students currently enrolled (an integer).

Add the following methods to your class:

- `details()`: Returns a string in the following format:

  `"College Name: [name], Location: [location], Total Students: [total_students]"`
- `enroll(students)`: Increases the total number of students by the specified number of students enrolled.
- `graduate(students)`: Decreases the total number of students by the specified number of students graduated. If the number of graduates exceeds the current total number of students, return an error message:

  `"Error: Cannot graduate more students than currently enrolled."`
- `college_age(current_year)`: Returns the age of the college by subtracting the established year from the current year.