Topics:
String methods, Lists.
----

1. Maximum Occurrence
Write a function that gets a string as an argument and returns the letter with the maximum occurrence in it.
Ex:
```
>>> s = 'Astana'
>>> print(most_used(s))
'a'
```

----

2. Max & Min
Write a function that finds the biggest and the smallest number in a given list.
(Try to avoid using the built-in `max`, `min` and `sort` functions).
Ex:
```
>>> t = [23, 7, 55, -2]
>>> maxmin(t)
Max: 55
min: -2
```

----

3. List Sum
a)
Write a `list_sum` function that calculates the sum of numbers in a given list.
(You are not allowed to use the built-in `sum` function).
Ex:
```
>>> t = [1,2,3,4,5]
>>> list_sum(t)
Sum: 15
```

b)
Will your above solution work with nested lists?
Ex:
```
>>> t2 = [[1],[2,3],[4,5]]
>>> nested_sum(t2)
15
```

Write another function called `nested_sum` that takes a list of lists of integers and adds up the elements from all of the nested lists. (Exercise 10.1 in the textbook)

----

4. Mixed Sum
Generalize your solution for the previous problem (List Sum) so that it works for different cases.
You may need to use recursion principle.
Ex:
```
>>> t3 = [1,[2,3],[4,[5]]]
>>> mixed_sum(t3)
15
```

5. Cumulative Sum

(Exercise 10.1 in the textbook)
Write a function called `cumsum` that takes a list of numbers and returns the cumulative sum; that is, a new list where the `i-th` element is the sum of the first `i + 1` elements from the original list.

Ex:

```
>>> t = [1, 2, 3]
>>> cumsum(t)
[1, 3, 6]
```

----

6. Word Sort

Write a program that accepts a comma separated sequence of words as input, and prints the words in a comma-separated sequence after sorting them alphabetically.

Ex:

```
>>> word_sort()
Enter your words separated by comma(no spaces):
without,hello,bag,world
Your words sorted out:
Bag,hello,without,world
```

----

7. Anagrams

(Exercise 10.6 in the textbook)
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.

Ex:

```
>>> is_anagram(‘return’,’turner’)
True
>>> is_anagram(‘free’,’refer’)
False
```

----

8. Duplicates

a)

(Exercise 10.7 in the textbook)
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.

Ex:

```
>>> has_duplicates([1, 2,3,2,4,5])
True
>>> has_duplicates([‘s’,’d’,’u’,2,0])
False
```

b)

Write another function called `remove_duplicates` that takes a list and returns a new list that does not contain duplicate values. You may use `has_duplicates` to check the list.

Ex:

```
>>> t1 = [1,2,3,2,1,3]
>>> t2 = remove_duplicates(t1)
>>> t2
[1,2,3]
```

----

9. Birthdays

(Exercise 10.8 in the textbook)
This exercise pertains to the so-called Birthday Paradox, which you can read about at http://en.wikipedia.org/wiki/Birthday_paradox.
If there are 23 students in your class, what are the chances that two of you have the same birthday? You can estimate this probability by generating random samples of 23 birthdays and checking for matches.
Hint: you can generate random birthdays with the `randint` function in the `random` module.

----

10. Word Count

a)
Write a function that calculates the number of words in a given text file.
Ex:

words.txt
This file has 2 lines.
There are 14 words, because numbers are also counted.

```
>>> word_count('words.txt')
Word count: 14
```

b)
Modify your code so that numbers are not counted as words.
Ex:

```
>>> word_count2('words.txt')
Word count: 12
```