Asignment 4

Q1. Write a function ends() that takes in a string from the user and prints just the first two and the last two characters of the string. You may assume that any input will be at least 2 characters long.

Enter a string >>> spring 3

spng

Q2 Write a function mix() that takes two strings a and b and prints the two strings concatenated, but with the first two characters of each word swapped with the other word's first two characters. You may assume that any input will be at least two characters long.

For example:

String a >>> german

String b >>> english

Output -> enrman geglish

Example 2 :

String a >>> dog

3 String b >>> dinner

4 dig donner

Q3 Write a function split which divides a string into two halves. If the length is even, the front and back halves are the same length. If the length is odd, we'll say that the extra character goes in the front. For example, in ’abcde’, the front half is ’abc’ and the back half is ’de’. Write a function that takes in two strings a and b and prints a-front + b-front + a-back + b-back.

Eg1

String a >>> abcd

String b >>> efghi

abefgcdhi

Eg 2

String a >>> this dinner is

String b >>> what am i doing1

this diwhat am nner isi doing1

Q4. Any fraction can be written as the division of two integers. You could express this in Python as a tuple – (numerator, denominator). Write functions for each of the following. They must use the tuple representation to return fractions.

1. Given two fractions as tuples, multiply them.
2. Given two fractions as tuples, divide them.
3. Given a list of fractions as a tuple, return the one that is smallest in value.

Also write a small command-line interface such that the user running your script sees something like this:

```
 == Multiplication and Division ==
Enter a fraction >>> 5/3
Enter a fraction >>> 10/3
Multiplication of the fractions: 50/9
```

Division of the first by the second: 5/10

Smallest fraction

Enter a fraction >>> 1/3

Enter a fraction >>> 10/3

Enter a fraction >>> 6/4

Enter a fraction >>> stop

Output Smallest fraction: 1/3

Q5. Take in numbers as input until “stop” is entered. Then split the numbers into three lists: one containing all the numbers, one containing all even values, and one containing all odd. Print out all three lists, as well as each list’s sum and average. Assume all input values are integers.

Q6. Take in numbers as input until “stop” is entered. As you take in each number, insert it into a list so that the list is sorted in ascending order. That is, look through the list until you find the place where the new element belongs. If the number is 21 is already in the list, do not add it again. After “stop” is entered, print out the list. Do not use any of Python’s built-in sorting functions.

Example :

1 Input a number >>> 12

2 List contains [12.0]

3 Input a number >>> 5.2

4 List contains [5.2, 12.0]

5 Input a number >>> 73

6 List contains [5.2, 12.0, 73.0]

```
Input a number >>> 45
List contains [5.2, 12.0, 45.0, 73.0]
Input a number >>> 100
List contains [5.2, 12.0, 45.0, 73.0, 100.0]
Input a number >>> -5
List contains [-5.0, 5.2, 12.0, 45.0, 73.0, 100.0]
Input a number >>> 2.3
List contains [-5.0, 2.3, 5.2, 12.0, 45.0, 73.0, 100.0]
Input a number >>> stop
[-5.0, 2.3, 5.2, 12.0, 45.0, 73.0, 100.0]
```

Q7. Write the following functions:

overlap() Given two lists, find a list of the elements common to both lists and return it.

join() Given two lists, join them together to be one list without duplicate elements and return that list.

Q8. Create a list, squares, that consists of the squares of the numbers 1 to 10. A list comprehension could be useful here.

***Use filter() and a lambda expression to print out only the squares that are between 30 and 70 (inclusive).

Q9. Define a function called count that has two arguments called sequence and item. Return the number of times the item occurs in the list.For example: count([1,2,1,1], 1) should return 3 (because 1 appears 3 times in the list).

Q10. Use a list comprehension to create a list, `cubes_by_four. The comprehension should consist of the cubes of the numbers 1 through 10 only if the cube is evenly divisible by four. Finally, print that list to the console. Note that in this case, the cubed number should be evenly divisible by 4, not the original number.

Q11. Create a list, to_21, that's just the numbers from 1 to 21, inclusive. Create a second list, odds, that contains only the odd numbers in the to_21 list (1, 3, 5, and so on). Use list slicing for this one instead of a list comprehension. Finally, create a third list, middle_third, that's equal to the middle third of `to_21, from 8 to 14, inclusive.

Q12. The string

garbled = "!XeXgXaXsXsXeXmX XtXeXrXcXeXsX XeXhXtX XmXaX XI"`

is garbled in two ways:

First, our message is backwards;

Second, the letter we want is every alternate letter.

Use lambda and filter to extract the message and save it to a variable called message. Use list slicing to extract the message and save it to a variable called message.

Q13 Q Write a python program to read three numbers (a,b,c) and check how many numbers between ‘a’ and ‘b’ are divisible by ‘c’

Q14. Q Pig Latin is a language game, where you move the first letter of the word to the end and add "ay." So "Python" becomes "ythonpay." To write a Pig Latin translator in Python, here are the steps we'll need to take:

Ask the user to input a word in English.

Make sure the user entered a valid word.

Convert the word from English to Pig Latin.

Display the translation result.

Q15. Q Write a program that generates a random number (0-10) and ask you to guess it. You have three asserts.

Define a random_number with randint between 0-10.

Initialize guesses_left to 3.

Use a while loop to let the user keep guessing so long as guesses_left is greater than zero.

Ask the user for their guess.

If they guess correctly, print 'You win!' and break. Decrement guesses_left by one.

Use an else: case after your while loop to print:You lose.