Lab 4

Loops and sequences

Contents

Sum of digits

Write a program, which prompts the user for an integer N then computes and prints the sum of its digits.

- Validate the input.
- Store the input as a string and iterate through the characters to extract the digits

Sample run:

```
This program computes the sum of digits of an integer
Please enter an integer: 58742
The sum of digits of (58742) = 26
```

Palindrome

Write a program that prompts for a string (of arbitrary length), and prints whether the string entered is a palindrome. A palindrome is a string that reads the same from left to right and from right to left.

You can use a guess and check approach or compare the string against its reverse.

```
enter palindrome: HannaH
'HannaH' is a palindrome
```

```
enter palindrome: reem
'reem' is not a palindrome
```

String transformation

Write a program which prompts the user for a string then prints a modified version of the input string where every space is replaced by the underscore character and every non-alphabetic character is removed.

Hint: Strings do not support individual element modification. They are ***immutable***. Use an accumulator to create the new string

Standard deviation

Write a program that computes the average of a user-prompted set of floating point numbers. Prompt the user for each number separately. The user needs to enter the string 'q' or 'Q', instead of a number, to signal the end of the sequence of numbers. Any non-numerical entry should be ignored in calculations.

The standard deviation is computed as

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$$

Where μ is the mean (average)

Here is a sample run:

```
This program computes the average of a set of numbers
Please enter the one number at each prompt. Enter q or Q to stop prompting
and get the result
? 2.1
? 3.6
? hello
You entered a non numerical value. This entry will be ignored
? 4.6
? 2
? q
The average of this set of values is: 3.075
The standard deviation of this set of values is: 1.0848386976873565
```

Write a second version of the code where the numbers are all entered at once, space-separated. Validation of the float type is still required.

String difference

Write a program that prompts the user for two, space-separated words (words do not include spaces but can include any other printable character i.e. alphabetic, numeric, punctuation, etc …).

- If the user enters less than two words, the program prints a message and exits.
- If the user enters more than two words, the additional words are ignored

The program then prints the "difference" of both word. For this problem, the "difference" of two words is defined as the sequence of characters from the longest word left after removing all case-sensitive occurrences of all the characters of the shortest word.

- If the two words have the same length, the first is considered the longest

Hints:

- The string method `find` may be useful
- There are two possible approaches:
    - removing from the long string, characters that appear in the short string
    - accumulate the characters from the long string that do not appear in the short string

Sample runs:

User enters less than two words:

```
Please enter two space-separated words: abcdfegh

You entered only one word.
```

User enters more than two words

```
Please enter two space-separated words: HelloEveryone oe test

The difference of HelloEveryone and oe is HllEvryn
```

First word is shorter that the second word:

```
Please enter two space-separated words: ace abcdefgh

The difference of abcdefgh and ace is bdfgh
```

Longest word does not contain any character from the shortest:

```
Please enter two space-separated words: qwertyuuiop asdfgh

The difference of qwertyuuiop and asdfgh is qwertyuuiop
```

The two words have the same length:

```
Please enter two space-separated words: qawsedrf edrftgyh

The difference of qawsedrf and edrftgyh is qaws
```

Anagrams

Write a program which prompts the user two strings then finds out whether they are anagrams. Anagrams are words (more generally strings) composed of the same characters only in a different order e.g. “listen” and “silent”.

There are multiple algorithms to solve this problem. Below are a couple suggestions, but you can use others:

- A simple (but inefficient) one of them is to count the number of occurrences of each character in the first string and check that the second have the same counts. Then, count the number of occurrences of each character of the second string and check that the first string have the same count. If at any point counts do not match, the strings are not anagrams.
- Another simple approach would be to “find and remove” from the second string, a match of each character of the first string. If a match is not found, the strings are not anagrams. If at the end of the matching, the second string is not empty, the strings are not anagrams.

Make sure to use string operators and methods to simplify your implementation.

Here are some sample runs:

```
This program finds out if two strings are anagrams

Please enter the first string: silent

Please enter the second string: listen

silent and listen are anagrams
```

```
This program finds out if two strings are anagrams

Please enter the first string: silents

Please enter the second string: listen

silents and listen are not anagrams
```

List rotation

Write a program that prompts the use for a set of space-separated integers and a number of rotations n > 0, then rotates the list n times.

Rotating the list one time consists in:

- the second element becoming the first
- the third element becoming the second
- the fourth element becoming the third
- etc ….
- And the first element becoming the last

Here are some sample runs:

```
Please enter a set of space-separated integers: 1 2 3 4 5 6 7 8 9 10

Please enter the number of rotations: 1

initial list: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

list rotated 1 times: [2, 3, 4, 5, 6, 7, 8, 9, 10, 1]
```

```
Please enter a set of space-separated integers: 1 2 3 4 5 6 7 8 9 10

Please enter the number of rotations: 4

initial list: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

list rotated 4 times: [5, 6, 7, 8, 9, 10, 1, 2, 3, 4]
```

```
Please enter a set of space-separated integers: 1 2 3 4 5 6 7 8 9 10

Please enter the number of rotations: 10

initial list: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

list rotated 10 times: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Inverted Floyd Triangle

Write a program that prints an inverted Floyd Triangle where the sequence of numbers is decrementing.

Sample run:

```
This program prints an inverted Floyd triangle
Please enter a positive integer: 6
21    20    19    18    17    16
15    14    13    12    11
10    9     8     7
6     5     4
3     2
1
```

Print a pattern

Write a program which prompt the user for an even, strictly positive integer N then prints a symmetric design. The design is composed of N/2 lines of N characters each, where the line k, $k\ in$ [0..(N/2)−1], contains 2k stars at the center of the line, preceded by an opening parenthesis and a number of dashes and followed by a closing parentheses and the same number of dashes. Here are some sample outputs for corresponding input values:

N = 2	N = 4	N = 6
() 	-()- (**)	--()-- -(**)- (****)
N = 8	**N = 10**	**N = 5**
---()--- --(**)-- -(****)- (******)	----()---- ---(**)--- --(****)-- -(******)- (********)	only works with even numbers

List stats:

Write a program that prompts the user for a set of integers $a_0, a_1, \ldots, a_{n-1}$. Use only one prompt for space separated numbers. Print a message and exit if any entry is not an integer.
For each number in the provided set, the program should count how many other numbers are strictly smaller. The program should then print a list of these counts (order corresponds to input)

Sample run:

```
Please enter a set of space-separated integers: 3 6 8 9 2 6 1

[2, 3, 5, 6, 1, 3, 0]
```

Explanation:

Here we have:
2 numbers smaller than 3 that are 1 and 2
3 numbers smaller than 6 that are 1, 2 and 3
5 numbers smaller than 8 that are 1, 2, 3, 6 and 6
6 numbers smaller than 9 that are 1, 2, 3, 6, 6 and 8
1 number smaller than 2 that is 1
3 numbers smaller than 6 that are 1, 2 and 3
0 numbers smaller than 1

Remove duplicates

Write a program, which prompts the user for a sequence of space-separated integers. The sequence may contain multiple occurrences of any integer.

Then, it reads it into a list while ignoring non-integer entries. The program should, then, generate and print the list containing a single occurrence of each of the user's integer entries.

Sample run 1:

```
This program removes duplicates from a given sequence of integers

Please enter a space-separated sequence of integers: 1 2 3 1 a 4 2 b 5 3 c 6 7 8 8 9

[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Sample run 2:

```
This program removes duplicates from a given sequence of integers

Please enter a space-separated sequence of integers: 4 4 4 4 4 4

[4]
```

Sample run 3:

```
This program removes duplicates from a given sequence of integers

Please enter a space-separated sequence of integers: 1 2 3 4

[1, 2, 3, 4]
```

Sample run 4:

```
This program removes duplicates from a given sequence of integers

Please enter a space-separated sequence of integers: a b c d

[]
```

Unique elements

Write a program, which prompts the user for a sequence of space-separated integers. The sequence may contain multiple occurrences of any integer.

Then, it reads it into a list while ignoring non-integer entries. The program should, then, generate and print the list of the user's integer entries that are unique i.e. has been entered only once in the input sequence.

Sample run 1:

```
This program finds unique integers in a given sequence of integers

Please enter a space-separated sequence of integers: 1 2 3 1 a 4 2 b 5 3 c 6 7 8 8 9

[4, 5, 6, 7, 9]
```

Sample run 2:

```
This program finds unique integers in a given sequence of integers

Please enter a space-separated sequence of integers: 4 4 4 4

[]
```

Sample run 3:

```
This program finds unique integers in a given sequence of integers

Please enter a space-separated sequence of integers: 1 2 3 4

[1, 2, 3, 4]
```

Sample run 4:

```
This program finds unique integers in a given sequence of integers

Please enter a space-separated sequence of integers: a b c d

[]
```

String pattern 2

Write a program that prompts the user for a string. Then prints out a pattern as follows:

- The pattern contains n lines where n is the number of characters in the input string
- Line i contains the i^{th} character of the string repeated i times, where i = 1…n, then one space, then the i^{th}-last character of the string repeated n-i+1 times.
    - i=1: one repetition of the first character, one space, and n repetitions of the last character
    - i=2: two repetitions of the second character, one space, and n-1 repetitions of the second last character
    - …
    - i=n: n repetitions of the last character, one space and one repetition of the last character

Sample run:

This program prints a pattern based on your input

```
Please enter a string: string

s ggggggg

tt nnnnn

rrr iiii

iiii rrr

nnnnn tt

ggggggg s
```

String pattern 3

Write a program that prompts the user for a string. Then prints out a pattern as follows:

- The pattern contains n-1 lines where n is the number of characters in the input string
- Line i contains the input string with one additional space character separating the i^{th} last character from the previous one.
    - i=1: space separates the last character from the second last
    - i=2: space separates the second last character from the third last character
    - …
    - i=n-1: space separates second character from the first character

Sample run:

```
This program prints a pattern based on your input

Please enter a string: string

strin g

stri ng

str ing

st ring

s tring
```

Plot logarithm

Write a program which prompts the user for a strictly positive integer n, then prints the shape of the binary logarithm function where:

- The first line contains a '*' character, after n dashes
- The second line contains a '*' character, after n//2 dashes
- The third line contains a '*' character, after n//4 dashes
- The third line contains a '*' character, after n//8 dashes
- …
- The

The program should prompt repeatedly, until the user enters valid input.

Here are some sample runs:

```
Please enter a strictly positive integer: 1
-*
```

```
Please enter a strictly positive integer: 2
--*
-*
```

```
Please enter a strictly positive integer: 5
-----*
--*
-*
```

```
Please enter a strictly positive integer: 13
-------------*
------*
---*
-*
```

```
Please enter a strictly positive integer: 22
----------------------*
-----------*
-----*
--*
-*
```

```
Please enter a strictly positive integer: 38
-------------------------------------*
```

```
--------------------*
---------*
----*
--*
-*
```

```
Please enter a strictly positive integer: 65
----------------------------------------------------------------*
---------------------------------*
-----------------*
--------*
----*
--*
-*
```

Is prime?

Write a program that checks whether a user-prompted positive integer smaller than 1000 is a prime number. The program **keeps prompting the user until the input is in the right type and range.**

Algorithm:

To determine that, the number should be divided by the prime numbers smaller than its square root. If the number divides evenly by any of those prime numbers, it is not prime.

The prime numbers smaller than square root of 1000 are 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, and 31.

Implementation:

- The best approach to use is “Guess and Check”.
- The problem may be solved without loops but the code would be shorter and more elegant if it uses a loop.

Sample program output:

```
This program find out whether a positive integer smaller than1000 is prime
Please enter a positive integer smaller than 1000: abc
Not an integer
Please enter a positive integer smaller than 1000: -22
Out of range entry
Please enter a positive integer smaller than 1000: 2584
Out of range entry
Please enter a positive integer smaller than 1000: 100
100 is not prime
```

Another sample:

```
This program find out whether a positive integer smaller than1000 is prime
Please enter a positive integer smaller than 1000: 383
383 is prime
```

Another sample:

```
This program find out whether a positive integer smaller than1000 is prime
Please enter a positive integer smaller than 1000: 23
23 is prime
```

Primes up to N

Write a program that **keeps prompting until the user enters a positive integer number N.**
The program then finds and prints the list of **all the prime numbers** less or equal to N, by adding the numbers to list one number at a time, in order.

Algorithm:

To begin, the list of prime numbers contains only the first prime i.e. 2. The program then should check if the numbers between 3 and N are prime numbers. When a prime is found, it is added to the current list of primes. A number is prime, if it does not evenly divide by any prime number identified so far (can also check only primes smaller than its square root).

Implementation:

This algorithm requires two nested loops:

- One loop that runs through the numbers from 3 to N, checking one number at each iteration and eventually appending it to the list of primes
- One loop that, given the number selected by the outer loop, divides it by the elements of the list of primes.

Notice that identifying a given number as prime would use a "Guess and Check" approach, similarly to the previous exercise.

Sample program output:

```
This program finds the prime numbers up to a given upper bound.
Please enter a positive integer for the upper bound on prime numbers: abc
Not an integer
Please enter a positive integer for the upper bound on prime numbers: -30
Number is negative
Please enter a positive integer for the upper bound on prime numbers: 100
Prime numbers smaller than 100: [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37,
41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97]
```

Sieve of Eratosthenes

Write a program that keeps prompting until the user enters a positive integer number N.
The program then finds all the prime numbers less or equal to N, using the sieve of Eratosthenes algorithm.

Algorithm:

- Start by creating a list of the numbers from 2 to N. The objective is to remove all non-prime numbers from the list and leave only the primes.
- Obviously, the first number on the list, 2, is prime, whereas all multiples of 2 are non-prime. So, in the first step, loop through the whole list removing all multiples of 2.
- The second number in the remaining list is 3 which is also prime, while its multiples are not. So, the second step consists in removing all multiples of 3 from the list.
- At this point, the next (third) number in the remaining list is 5 since 4 has been removed due to being a multiple of 2. 5 is a prime number; it should be kept in the list but all its multiples have to be removed and that would be the third step of the algorithm.
- It should continue in the same way i.e. select/consider the next number in the list and remove its multiples until selection reaches the last number left in the list.

Implementation:

- To generate the list of integers `[i, i+1, i+2, …, j]` use the syntax `[x for x in range(i, j+1)]`
- To remove an element `x` of a list `ls`, use `ls.remove(x)`. Note this raises an exception, if `x` is not in `ls`
- There is an arbitrary number of “steps” depending on the user input. Thus, these steps are iterations of an outer loop that selects the next prime number, while the inner loop performs removal (from the list) of the multiples of the selected prime number.

Multiplication chart

Write a program which displays the *multiplication chart*. You should prompt the user for a **single positive integer** as the size for the multiplication chart displayed (i.e., 10x10, 15x15, 18x18 etc.). The first row and the first column must be denoted for the chart's multipliers. At the end, your output should look like this:

```
Please specify the size of the multiplication chart: 10

   |   1   2   3   4   5   6   7   8   9   10
---------------------------------------------------
1  |   1   2   3   4   5   6   7   8   9   10
2  |   2   4   6   8  10  12  14  16  18   20
3  |   3   6   9  12  15  18  21  24  27   30
4  |   4   8  12  16  20  24  28  32  36   40
5  |   5  10  15  20  25  30  35  40  45   50
6  |   6  12  18  24  30  36  42  48  54   60
7  |   7  14  21  28  35  42  49  56  63   70
8  |   8  16  24  32  40  48  56  64  72   80
9  |   9  18  27  36  45  54  63  72  81   90
10 |  10  20  30  40  50  60  70  80  90  100
```

Besides nested loops, you will also need to use some string formatting methods to set the correct *width* and *alignment* for the chart's contents. https://docs.python.org/3/library/string.html#format-specification-mini-language

Reduce a list to 0

Write a program that prompts the user for a set of space-separated positive integers $a_0, a_1, \ldots, a_{n-1}$. The program then reads and stores only positive integers and ignores any invalid entry.

Then, your program should count the steps to reduce each number a_i to 0 using these rules:

- If the number is even, divide it by 2
- If the number is odd, decrement it by 1

For instance, to reduce the number 10:

- 10 is even, divide 10 by 2, it becomes 5
- 5 is odd, decrement 5 by 1, it becomes 4
- 4 is even, divide 4 by 2, it becomes 2
- 2 is even, divide 2 by 2, it becomes 1
- 1 is odd, decrement 1 by 1, it becomes 0

So, 5 steps are needed to reduce 10 to 0.

Then, the program prints the output as a **list of pairs**: **(input number, count of steps)**.

Sample run:

```
Please enter a set of space-separated positive integers: 3 45 st 59 16 32 89

[(3, 3), (45, 9), (59, 10), (16, 5), (32, 6), (89, 10)]
```

Intersection

Write a program that prompts the user for two sequences of space-separated values. The sequences may appear to contain mixed-type values, but can be considered/handled as sequences of strings. The sequences can also contain multiple copies of any value.

The program then generates a list of all the elements of the intersection of the two sequences (in any order), taking into account the replication of the values. This means that if a value v appears twice in the first sequence and three times in the second sequence, then the intersection should contain two instances of v (but they can be stored at any position in the list).

The program finally, prints the elements of the intersection, **strictly following the format** provided in the sample runs below.

Sample runs:

```
This program finds the intersection of two sets

Please enter the space-separated elements of the first set: a b c d

Please enter the space-separated elements of the second set: e f g h i j k

The intersection of these two sets is {}
```

```
This program finds the intersection of two sets

Please enter the space-separated elements of the first set: 12 k e 34 1.5 12
hi 12 0.2

Please enter the space-separated elements of the second set: 1.5 hi 12 0.1 54
12 hi hi hi

The intersection of these two sets is {12, 12, 1.5, hi}
```

Matrix input and print

Write a program which prompts the user for matrix dimensions then its **integer** elements one by one (input validation required). Prompting for elements should stop once all elements have been entered. Once all data is entered, the program prints the matrix in left-aligned columns. Columns should be of width 10 characters.

Hints:

- We can use the string format method or printf-style formatting to write a number is a pre-specified width. This will allow us to obtain aligned columns. https://docs.python.org/3/library/string.html#format-specification-mini-language
- Note that one cannot know in advance the width needed to fit any element of the matrix. So if some element of the matrix has more digits than the pre-specified width of 10 characters, the output will not be aligned and separated as expected. This is an acceptable limitation, though it can be overcome.

Sample run:

```
This program prints a matrix in aligned rows and columns
What is the number of rows: 3
What is the number of columns: 3
Please enter the elements one by one at the prompts
? 1
? 234
? -5
? 67
? 8
? 911
? 23
? 4
? -56
1         234       -5
67        8         911
23        4         -56
```

Matrix input and sum

Write a program that asks the user to provide two matrices of the same dimensions and prints their sum. Follow these guidelines:

1. Prompt for matrix dimensions
2. Prompt for all elements of the first matrix at once (space-separated) and make sure they are integers and that you have enough elements (extra numbers should be ignored, discarded)
3. Store the matrix as follows: each row is a list of integers, the matrix is a list of rows i.e. a list of lists of integers
4. Repeat 2. and 3. to get the second matrix stored in the same type of data structure
5. Compute the sum of the matrices and store it in the same type of data structure
6. Display the sum matrix in left-aligned columns. Columns should be of width 10 characters.

Pascal' triangle

Write a program that displays a Pascal's triangle with n rows, where n is a user-prompted strictly positive integer. You will find information about Pascal's triangle in https://www.mathsisfun.com/pascals-triangle.html

Algorithm:

You must use a list of n lists to generate and store the triangle. For instance, the triangle with:

- 3 rows is stored as [[1], [1, 1], [1, 2, 1]]
- the triangle with 4 rows is stored as [[1], [1, 1], [1, 2, 1], [1, 3, 3, 1]]
- etc …

To generate the i^{th} row, you must use the sum property i.e. a number is the sum of the two that are on top of it, as shown in the referenced page.

Print the triangle with the columns left aligned of fixed width.

Sample run:

```
This program prints a Pascal's triangle with n line

Please provide the number of lines n: 1.2

n must be an integer

Please provide the number of lines n: -5

n must be at least 1

Please provide the number of lines n: 7

Pascal’s triangle with 7 lines:

1

1     1

1     2     1

1     3     3     1

1     4     6     4     1

1     5     10    10    5     1

1     6     15    20    15    6     1
```

(For a challenge, print it with all lines centered)

Largest water container

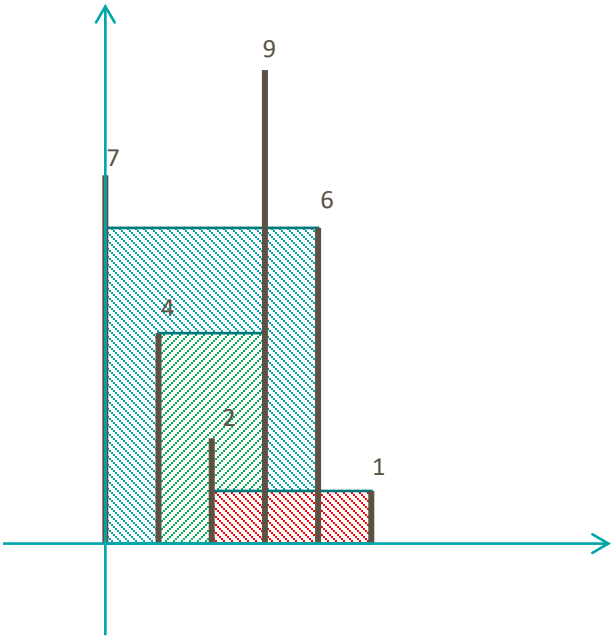
Write a program that prompts the user for a set of strictly positive integers $a_0, a_1, \ldots, a_{n-1}, n \geq 2$. Use only one prompt for space separated numbers. Print a message and exit if any entry is not a positive integer or if there are not enough numbers.

In the Cartesian plane, consider vertical lines starting at points $(i, 0)$ and finishing at points (i, a_i), for $0 \leq i \leq n-1$. Each pair of lines can be imagined as a representation (projection) of a water container. Your program should find and print out the pairs (i, a_i) and (j, a_j) that correspond to the container able to hold the most water. The algorithm is based on a guess and check approach, where the maximum water amount is the quantity to guess and amend. This quantity is amended if when selecting a pair of points and computing the amount of water contained by their corresponding container, it turns to be larger than the current guess. Use two nested loops to systematically select all possible pairs.

Hint: a container can hold water up to its shortest side

Sample run:

```
Please enter a set of space-separated integers: 7 4 2 9 6 1

The container with the most water is: [(0, 7), (4, 6)]
```



This is the illustration of the above sample run. The solution is the blue container defined by the points (0,7) and (4,6). The other two are only examples of other containers, potential solutions, that the Guess and check would discard.

Balanced string

Write a program that keeps prompting the user for a balanced string until getting a valid input. For the purpose of this program, a balanced string is defined as having an equal number of characters 'R' (capital) and characters 'L'(capital), regardless of the presence of other characters.

The program then separates (divides, splits) the balanced input string into as many balanced substrings as possible and generates a list of them.

The program finally prints to the screen the list of balanced substrings obtained.

Sample runs:

```
Please enter a string containing an equal number of 'R' and 'L': RLRLRLRLRL

['RL', 'RL', 'RL', 'RL', 'RL']
```

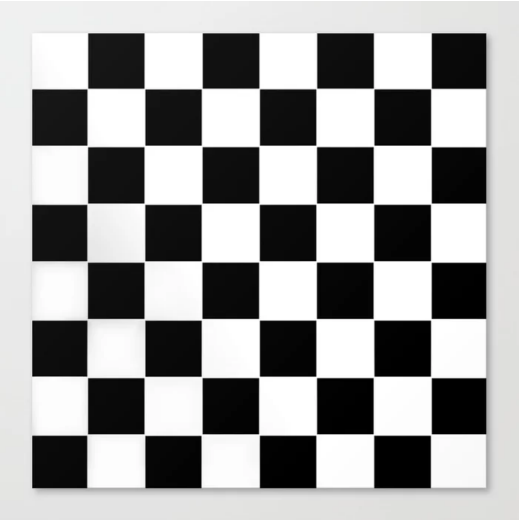
```
Please enter a string containing an equal number of 'R' and 'L': LLRLLLRRRRLR

['LLRLLLRRRR', 'LR']
```

```
Please enter a string containing an equal number of 'R' and 'L':
ABCREFLGLHLRIJKR

['ABCREFL', 'GLHLRIJKR']
```

```
Please enter a string containing an equal number of 'R' and 'L': RRRMLL

This string does not contain equal number of 'R' and 'L'

Please enter a string containing an equal number of 'R' and 'L': RRRMLLLL

This string does not contain equal number of 'R' and 'L'

Please enter a string containing an equal number of 'R' and 'L': RRRMLLL

['RRRMLLL']
```

Checkerboard

In this exercise, you will try to generate a data structure that abstracts (represents) a checkerboard. A checkerboard can be thought of as a matrix.



A checkerboard pattern.

A checkerboard image is a black and white image, which can be represented as a matrix of pixels (light bulbs). Each pixel can either be turned on (white)or off (black). Thus, an "on" pixel can be represented by a matrix element equal to 1 and an "off" pixel can be represented by a matrix element equal to 0.

Write a program which prompts the user for the width of a checkerboard cell (black or white square) w and the number of these cells along the board's side n. The width should be a positive integer and the number of cells a positive **even** integer. Then, the program generates and prints to the screen either of the following data structures:

1. A matrix (list of lists) M of 0's and 1's, where 0's correspond to white pixels and 1's correspond to black pixels. Examples of these matrices,
    - For w = 2 and n =2, `M = [ [ 0, 0, 1, 1], [0, 0, 1, 1], [1, 1, 0, 0], [1, 1, 0, 0]]`
    - For w=2 and n = 4, `M = [ [ 0, 0, 1, 1, 0, 0, 1, 1], [ 0, 0, 1, 1, 0, 0, 1, 1], [1, 1, 0, 0, 1, 1, 0, 0], [1, 1, 0, 0, 1, 1, 0, 0], [ 0, 0, 1, 1, 0, 0, 1, 1], [ 0, 0, 1, 1, 0, 0, 1, 1], [1, 1, 0, 0, 1, 1, 0, 0], [1, 1, 0, 0, 1, 1, 0, 0] ]`
    - For w = 4 and n = 2, `M = [ [0, 0, 0, 0, 1, 1, 1, 1], [0, 0, 0, 0, 1, 1, 1, 1], [0, 0, 0, 0, 1, 1, 1, 1], [0, 0, 0, 0, 1, 1, 1, 1], [1, 1, 1, 1, 0, 0, 0, 0], [1, 1, 1, 1, 0, 0, 0, 0], [1, 1, 1, 1, 0, 0, 0, 0], [1, 1, 1, 1, 0, 0, 0, 0] ]`

    This matrix should be printed with the rows and columns aligned and elements spaced by one (or two) space. So, when printed, the matrix corresponding to w = 4 and n = 4 looks like:

    ```
    0  0  0  0   1  1  1  1  0  0  0  0   1  1  1  1
    ```

```
0  0  0  0   1  1  1  1  0  0  0  0   1  1  1  1
0  0  0  0   1  1  1  1  0  0  0  0   1  1  1  1
0  0  0  0   1  1  1  1  0  0  0  0   1  1  1  1
1  1  1  1   0  0  0  0  1  1  1  1   0  0  0  0
1  1  1  1   0  0  0  0  1  1  1  1   0  0  0  0
1  1  1  1   0  0  0  0  1  1  1  1   0  0  0  0
1  1  1  1   0  0  0  0  1  1  1  1   0  0  0  0
0  0  0  0   1  1  1  1  0  0  0  0   1  1  1  1
0  0  0  0   1  1  1  1  0  0  0  0   1  1  1  1
0  0  0  0   1  1  1  1  0  0  0  0   1  1  1  1
0  0  0  0   1  1  1  1  0  0  0  0   1  1  1  1
1  1  1  1   0  0  0  0  1  1  1  1   0  0  0  0
1  1  1  1   0  0  0  0  1  1  1  1   0  0  0  0
1  1  1  1   0  0  0  0  1  1  1  1   0  0  0  0
1  1  1  1   0  0  0  0  1  1  1  1   0  0  0  0
```

2. A multi-line string S of spaces and hashes (#), where spaces correspond to white pixels and hashes correspond to black pixels. Examples of these strings are:
   - For w = 2 and n =2, S = “ ##\n ##\n## \n## \n”
   - For w=2 and n = 4, S = “ ## ##\n ## ##\n## ## \n## ## \n ## ##\n ## ##\n## ## \n## ## \n”
   - For w = 4 and n = 2, S = “ ####\n ####\n ####\n ####\n####\n #### \n #### \n #### \n #### \n”

   Printing the string corresponding to w = 4 and n = 4 gives the following pattern on the screen:

```
    ####    ####
    ####    ####
    ####    ####
    ####    ####
####    ####
####    ####
####    ####
####    ####
    ####    ####
    ####    ####
    ####    ####
    ####    ####
####    ####
####    ####
####    ####
####    ####
```

Composing words

You are given a Python script, where two objects have been defined:

- A list of strings: **`words`**
- A string of characters: **`letters`**

In this file, write a code that finds out which of the strings from the list **`words`**, can be formed using the characters of the string **`letters`**, when using each occurrence of each character once.

The output of the program should be:

```
Out of:  ['PICTORIALLY', 'UNDISPUTED', 'TEARING', 'CIDERS', 'PREMIERE',
'BOUTIQUE', 'PLETHORA', 'MAGNIFICATION', 'REGURGITATING', 'TWENTIES']

Using the characters:  PCOILYNIPTDERNCDRPEIRBUIUPEHRMGIIAINEUGTTNTETE

We can form:  ['TEARING', 'PREMIERE', 'PLETHORA']
```

Note that:

- The word PICTORIALLY cannot be formed because there is only one L in the string **`letters`**
- The code should be written independently from the size or contents/values of the given objects (input instance); i.e. shall we change the hardcoded values of these objects, the code should work correctly.
- You can use the other possible input instances, which are commented, to further test your code.

App with menu

Write a program, which repeatedly offers the user to perform multiple operations on a (initially empty) list of floating point values. Indeed, the program presents the user with a menu, then based on the user entry, performs one of the following operations:

User entry	operation
a	Append a number to the list
d	Delete a number from the list (position to be provided later)
p	Print the tab-separated list elements (see sample run for formatting)
q	To quit and terminate the program

Note that the menu choices are case sensitive (i.e. A, D, P and Q are not valid menu choices).

Sample run:

```
This program allows you to create and manipulate a set of floating point
values
==============
a - add new element
d - delete element
p - print data
q - quit
>> p
==============

==============
a - add new element
d - delete element
p - print data
q - quit
>> a
==============
please enter a number to append: a
Not a float.
==============
a - add new element
d - delete element
p - print data
q - quit
>> a
==============
please enter a number to append: 2.4
==============
a - add new element
d - delete element
p - print data
q - quit
>> p
```

```
==============
2.4
==============
a - add new element
d - delete element
p - print data
q - quit
>> a
==============
please enter a number to append: 8
==============
a - add new element
d - delete element
p - print data
q - quit
>> a
==============
please enter a number to append: 47.23
==============
a - add new element
d - delete element
p - print data
q - quit
>> a
==============
please enter a number to append: 5
==============
a - add new element
d - delete element
p - print data
q - quit
>> p
==============
2.4   8.0   47.23 5.0
==============
a - add new element
d - delete element
p - print data
q - quit
>> d
==============
please enter index: 5
Index out of range.
==============
a - add new element
d - delete element
p - print data
q - quit
>> d
==============
```

```
please enter index: -3
Index out of range.
==============
a - add new element
d - delete element
p - print data
q - quit
>> 3
==============
Not a valid choice
==============
a - add new element
d - delete element
p - print data
q - quit
>> d
==============
please enter index: 3
==============
a - add new element
d - delete element
p - print data
q - quit
>> p
==============
2.4   8.0   47.23
==============
a - add new element
d - delete element
p - print data
q - quit
>> q
```

Histogram

Write a program, which prompts the user for a set of student scores then produces a histogram of those scores based on their range.

Input:

The program prompts for space-separated integer scores in the range `[0 .. 100]`, reads and ignores any invalid value (wrong type or out-of-range). At this point, the valid scores should be stored as a list of integers.

Computation:

The program then counts how many scores are in each of the ranges `[0, 9]`, `[10, 19]`, `[20, 29]`, `[30, 39]`, `[40, 49]`, `[50, 59]`, `[60, 69]`, `[70, 79]`, `[80, 89]`, `[90, 99]` and `[100, 100]`. To compute the count of scores in each range, you can create a list (say `histogram`) of 11 integers, all initialized to zero. Then, loop through the list of scores, incrementing `histogram` elements, such that `histogram[i]` corresponds to the count of scores in the range `[10*i, 10*i+9]`. For example, if the score is 84 then `histogram[8]` should be incremented.

Output:

The printout consists of 11 lines each one starts with a range and then a number of dashes corresponding to the number of scores within that range. Refer to the sample run for illustration of the output format.

This program produces a histogram of a set of scores (all between 0 and 100).

```
Please provide all the scores space-separated (invalid scores will be
ignored): 25 99 45 100 92 3 28 100 a 74 42 55 62 98 100 93 101 81 72 58 63.5
94 71 96 93
[ 0,  9]: -
[10, 19]:
[20, 29]: --
[30, 39]:
[40, 49]: --
[50, 59]: --
[60, 69]: -
[70, 79]: ---
[80, 89]: -
[90, 99]: -------
[   100]: ---
```