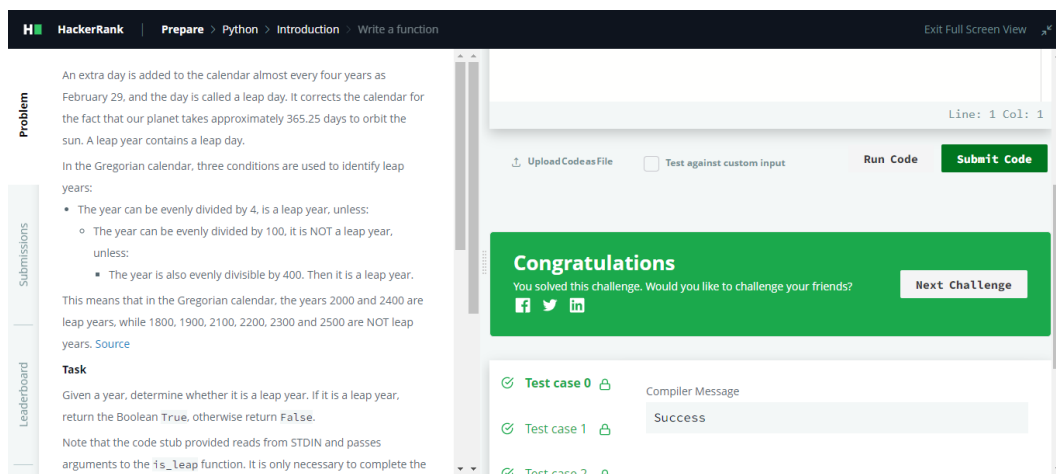


Assignment No 3
Muhammad Adnan Kundi
360998
Artificial Intelligence
Dr Yasar Ayaz

1 write an function



Problem

An extra day is added to the calendar almost every four years as February 29, and the day is called a leap day. It corrects the calendar for the fact that our planet takes approximately 365.25 days to orbit the sun. A leap year contains a leap day.

In the Gregorian calendar, three conditions are used to identify leap years:

- The year can be evenly divided by 4, is a leap year, unless:
 - The year can be evenly divided by 100, it is NOT a leap year, unless:
 - The year is also evenly divisible by 400. Then it is a leap year.

This means that in the Gregorian calendar, the years 2000 and 2400 are leap years, while 1800, 1900, 2100, 2200, 2300 and 2500 are NOT leap years. [Source](#)

Task

Given a year, determine whether it is a leap year. If it is a leap year, return the Boolean True, otherwise return False.

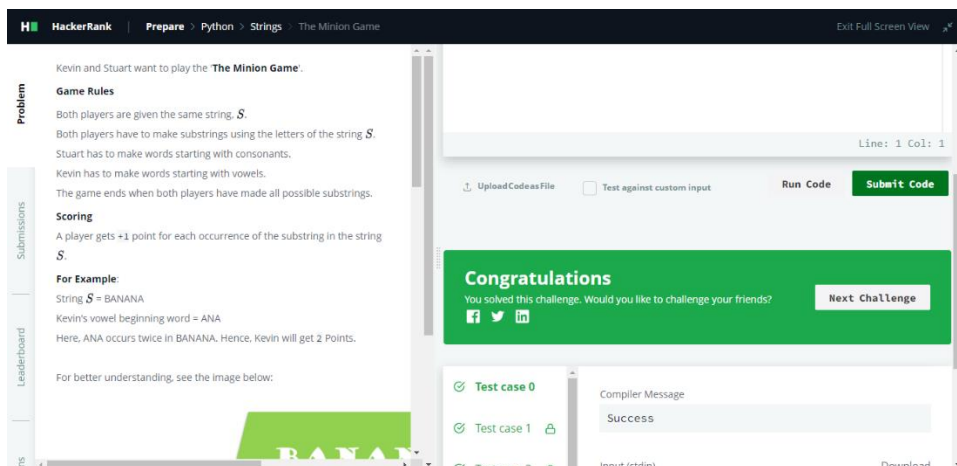
Note that the code stub provided reads from STDIN and passes arguments to the `is_leap` function. It is only necessary to complete the

Congratulations
You solved this challenge. Would you like to challenge your friends?
[f](#) [t](#) [in](#) [Next Challenge](#)

Test case 0 [🔒](#) Compiler Message
Test case 1 [🔒](#) Success
Test case 2 [🔒](#)

```
def is_leap(year):  
    leap = False  
  
    # Write your logic here  
    leap = (year % 400 == 0) or (year % 4 == 0 and year % 100 !=  
0)  
    return leap  
  
year = int(input())  
print(is_leap(year))
```

2 The Minion Game



Kevin and Stuart want to play the 'The Minion Game'.

Game Rules

Both players are given the same string S .
Both players have to make substrings using the letters of the string S .
Stuart has to make words starting with consonants.
Kevin has to make words starting with vowels.
The game ends when both players have made all possible substrings.


Scoring

A player gets +1 point for each occurrence of the substring in the string S .

For Example:

String S = BANANA
Kevin's vowel beginning word = ANA
Here, ANA occurs twice in BANANA. Hence, Kevin will get 2 Points.

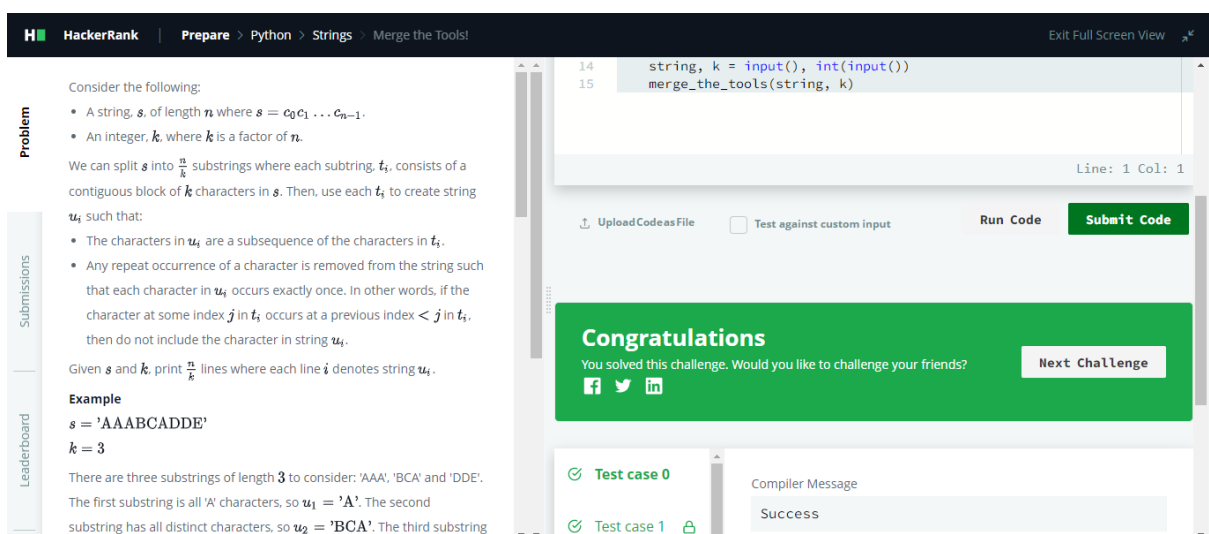
For better understanding, see the image below:



```
def minion_game(string):
    # your code goes here
    vowel = 'aeiou'.upper()
    strl = len(string)
    kevin = sum(strl -
i for i in range(strl) if string[i] in vowel)
    stuart = strl * (strl + 1) / 2 - kevin
    if kevin == stuart:
        print ('Draw')
    elif kevin > stuart:
        print ('Kevin %d' % kevin)
    else:
        print ('Stuart %d' % stuart)
if __name__ == '__main__':
```

```
def minion_game(string):
    # your code goes here
    vowel = 'aeiou'.upper()
    strl = len(string)
    kevin = sum(strl -
i for i in range(strl) if string[i] in vowel)
    stuart = strl * (strl + 1) / 2 - kevin
    if kevin == stuart:
        print ('Draw')
    elif kevin > stuart:
        print ('Kevin %d' % kevin)
    else:
        print ('Stuart %d' % stuart)
if __name__ == '__main__':
```

3 Merge The Tools



Consider the following:

- A string, s , of length n where $s = c_0c_1 \dots c_{n-1}$.
- An integer, k , where k is a factor of n .

We can split s into $\frac{n}{k}$ substrings where each subtring, t_i , consists of a contiguous block of k characters in s . Then, use each t_i to create string u_i such that:

- The characters in u_i are a subsequence of the characters in t_i .
- Any repeat occurrence of a character is removed from the string such that each character in u_i occurs exactly once. In other words, if the character at some index j in t_i occurs at a previous index $< j$ in t_i , then do not include the character in string u_i .

Given s and k , print $\frac{n}{k}$ lines where each line i denotes string u_i .

Example

s = 'AAABCADDE'
 k = 3

There are three substrings of length 3 to consider: 'AAA', 'BCA' and 'DDE'.
The first substring is all 'A' characters, so u_1 = 'A'. The second substring has all distinct characters, so u_2 = 'BCA'. The third substring

```
14 string, k = input(), int(input())
15 merge_the_tools(string, k)
```

```
def merge_the_tools(string, k):
    # your code goes here
    temp = []
    len_temp = 0
    for item in string:
        len_temp += 1
        if item not in temp:
            temp.append(item)
        if len_temp == k:
            print (''.join(temp))
            temp = []
            len_temp = 0
if __name__ == '__main__':
    string, k = input(), int(input())
    merge_the_tools(string, k)
```

4 Time Delta

Problem

When users post an update on social media, such as a URL, image, status update etc., other users in their network are able to view this new post on their news feed. Users can also see exactly when the post was published, i.e. how many hours, minutes or seconds ago.

Since sometimes posts are published and viewed in different time zones, this can be confusing. You are given two timestamps of one such post that a user can see on his newsfeed in the following format:

Day dd Mon yyyy hh:mm:ss +xxxx

Here +xxxx represents the time zone. Your task is to print the absolute difference (in seconds) between them.

Input Format

The first line contains T , the number of testcases. Each testcase contains 2 lines, representing time t_1 and time t_2 .

Constraints

- Input contains only valid timestamps
- $year \leq 3000$.

Output Format

Print the absolute difference ($t_1 - t_2$) in seconds.

```
13 time_format = '%a %d %b %Y %H:%M:%S %z'
14 t1 = datetime.strptime(t1, time_format)
15 t2 = datetime.strptime(t2, time_format)
16 return str(int(abs((t1-t2).total_seconds())))
17
18 if __name__ == '__main__':
19     fptr = open(os.environ['OUTPUT_PATH'], 'w')
20
21     t = int(input())
```

Line: 33 Col: 1

Upload Code as File ☐ Test against custom input **Run Code** **Submit Code**

Congratulations
You solved this challenge. Would you like to challenge your friends?

Next Challenge

Test case 0 Compiler Message
Success

Test case 1 Success

```
#!/bin/python3
```

```
import math
import os
import random
import re
import sys
```

```
# Complete the time_delta function below.
```

```
from datetime import datetime
def time_delta(t1, t2):
```

```

time_format = '%a %d %b %Y %H:%M:%S %z'
t1 = datetime.strptime(t1, time_format)
t2 = datetime.strptime(t2, time_format)
return str(int(abs((t1-t2).total_seconds())))

if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')

    t = int(input())

    for t_itr in range(t):
        t1 = input()

        t2 = input()

        delta = time_delta(t1, t2)

        fptr.write(delta + '\n')

    fptr.close()

```

5 Find Angle MBC

HackerRank

Prepare > Python > Math

Find Angle MBC

Exit Full Screen View

Problem

ABC is a right triangle, 90° at B .
Therefore, $\angle ABC = 90^\circ$.

Point M is the midpoint of hypotenuse AC .

You are given the lengths AB and BC .

Your task is to find $\angle MBC$ (angle θ° , as shown in the figure) in degrees.

Input Format

The first line contains the length of side AB

Submissions

Leaderboard

ans

Line: 11 Col: 1

Upload Code as File

☐ Test against custom input

Run Code

Submit Code

Congratulations

You solved this challenge. Would you like to challenge your friends?

Next Challenge

Test case 0

Compiler Message

Test case 1

Success

Test case 2

Input (stdin)

Download

```

# Enter your code here. Read input from STDIN. Print output to ST
DOUT
import math
ab=int(input())
bc=int(input())
ca=math.hypot(ab,bc)
mc=ca/2

```

```

bca=math.asin(1*ab/ca)
bm=math.sqrt((bc**2+mc**2)-(2*bc*mc*math.cos(bca)))
mbc=math.asin(math.sin(bca)*mc/bm)
print(int(round(math.degrees(mbc),0)),'\u00B0',sep='')

```

6 No Idea

The screenshot shows the HackerRank interface for the 'No Idea' problem. On the left, the problem description states: 'There is an array of n integers. There are also 2 disjoint sets, A and B , each containing m integers. You like all the integers in set A and dislike all the integers in set B . Your initial happiness is 0. For each i integer in the array, if $i \in A$, you add 1 to your happiness. If $i \in B$, you add -1 to your happiness. Otherwise, your happiness does not change. Output your final happiness at the end.' It also includes constraints: $1 \leq n \leq 10^5$, $1 \leq m \leq 10^5$, and $1 \leq \text{Any integer in the input} \leq 10^9$. The input format specifies three lines: n and m , the array elements, and the elements of sets A and B . The output format is a single integer representing total happiness. The main area shows a green 'Congratulations' banner with social media links and a 'Next Challenge' button. Below this, test cases 0 and 1 are marked as successful, and the compiler message says 'Success'.

Enter your code here. Read input from STDIN. Print output to STDOUT

```

if __name__ == "__main__":
    happiness = 0
    n, m = map(int, input().strip().split(' '))
    arr = list(map(int, input().strip().split(' ')))

    good = set(map(int, input().strip().split(' ')))
    bad = set(map(int, input().strip().split(' ')))

    for i in arr:
        if i in good:
            happiness += 1
        elif i in bad:
            happiness -= 1
    print(happiness)

```

7 Word Order

Enter your code here. Read input from STDIN. Print output to STDOUT

```

from collections import Counter
N = int(input())
LIST = []

```

```

for i in range(N):
    LIST.append(input().strip())
COUNT = Counter(LIST)
print(len(COUNT))
print(*COUNT.values())

```

HackerRank | Prepare > Python > Collections > Word Order

Problem

You are given n words. Some words may repeat. For each word, output its number of occurrences. The output order should correspond with the input order of appearance of the word. See the sample input/output for clarification.

Note: Each input line ends with a "\n" character.

Constraints:

- $1 \leq n \leq 10^5$
- The sum of the lengths of all the words do not exceed 10^6
- All the words are composed of lowercase English letters only.

Input Format

The first line contains the integer, n .
The next n lines each contain a word.

Output Format

Output 2 lines.
On the first line, output the number of distinct words from the input.
On the second line, output the number of occurrences for each distinct word according to their appearance in the input.

Sample Input

```

4
a
a
b
b

```

Line: 10 Col: 1

Upload Code as File ☐ Test against custom input **Run Code** **Submit Code**

Congratulations
You solved this challenge. Would you like to challenge your friends?

Test case 0 ☒ Test case 1 ☒ Test case 2 ☒

Compiler Message
Success

Input (stdin)
Download

8 Compress The String

HackerRank | Prepare > Python > Itertools > Compress the String!

Problem

In this task, we would like for you to appreciate the usefulness of the groupby() function of itertools. To read more about this function, [Check this out](#).

You are given a string S . Suppose a character 'c' occurs consecutively X times in the string. Replace these consecutive occurrences of the character 'c' with (X, c) in the string.

For a better understanding of the problem, check the explanation.

Input Format

A single line of input consisting of the string S .

Output Format

A single line of output consisting of the modified string.

Constraints

- All the characters of S denote integers between 0 and 9.
- $1 \leq |S| \leq 10^4$

Sample Input

```

1222311

```

Sample Output

```

(1,1)(2,2)3(1,1)

```

Line: 5 Col: 1

Upload Code as File ☐ Test against custom input **Run Code** **Submit Code**

Congratulations
You solved this challenge. Would you like to challenge your friends?

Test case 0 ☒ Test case 1 ☒ Test case 2 ☒

Compiler Message
Success

Input (stdin)
Download

```

# Enter your code here. Read input from STDIN. Print output to ST
DOUT
from itertools import groupby
for k, c in groupby(input()):

```

```
print("(%d, %d)" % (len(list(c)), int(k)), end=' ')
```

9 Company Logo

HackerRank

Prepare > Python > Collections > Company Logo

Exit Full Screen View

Problem

A newly opened multinational brand has decided to base their company logo on the three most common characters in the company name. They are now trying out various combinations of company names and logos based on this condition. Given a string s , which is the company name in lowercase letters, your task is to find the top three most common characters in the string.

- Print the three most common characters along with their occurrence count.
- Sort in descending order of occurrence count.
- If the occurrence count is the same, sort the characters in alphabetical order.

For example, according to the conditions described above, **GOOGLE** would have it's logo with the letters **G, O, E**.

Input Format

A single line of input containing the string S .

Constraints

- $3 < \text{len}(S) \leq 10^4$
- S has at least 3 distinct characters

Output Format

Submissions

Leaderboard

2/15

Line: 9 Col: 1

Upload Code as File

☐ Test against custom input

Run Code

Submit Code

Congratulations

You solved this challenge. Would you like to challenge your friends?

Next Challenge

Test case 0

Compiler Message

Success

Test case 1

Input (stdin)

Download

```
from collections import Counter
```

```
S = input()
S = sorted(S)
FREQUENCY = Counter(list(S))
```

```
for k, v in FREQUENCY.most_common(3):
    print(k, v)
```

10 Pilling Up

HackerRank

Prepare > Python > Collections > Pilling Up!

Exit Full Screen View

Problem

There is a horizontal row of n cubes. The length of each cube is given. You need to create a new vertical pile of cubes. The new pile should follow these directions: if $\text{cube}[i]$ is on top of $\text{cube}[j]$ then $\text{sideLength}[j] \geq \text{sideLength}[i]$.

When stacking the cubes, you can only pick up either the leftmost or the rightmost cube each time. Print Yes if it is possible to stack the cubes. Otherwise, print No.

Example

$\text{blocks} = [1, 2, 3, 8, 7]$

Result: No

After choosing the rightmost element, 7, choose the leftmost element, 1. After that, the choices are 2 and 8. These are both larger than the top block of size 1.

$\text{blocks} = [1, 2, 3, 7, 8]$

Result: Yes

Choose blocks from right to left in order to successfully stack the blocks.

Input Format

The first line contains a single integer T , the number of test cases. For each test case, there are 2 lines.

Submissions

Leaderboard

2/15

Line: 22 Col: 1

Upload Code as File

☐ Test against custom input

Run Code

Submit Code

Congratulations

You solved this challenge. Would you like to challenge your friends?

Next Challenge

Test case 0

Compiler Message

Success

Test case 1

Input (stdin)

Download

```
# Enter your code here. Read input from STDIN. Print output to ST
DOUT
ANS = []
T = int(input())
for _ in range(T):
    n = int(input())
    sl = list(map(int, input().split()))
    for _ in range(n-1):
        if sl[0] >= sl[len(sl)-1]:
            a = sl[0]
            sl.pop(0)
        elif sl[0] < sl[len(sl)-1]:
            a = sl[len(sl)-1]
            sl.pop(len(sl)-1)
        else:
            pass
    if len(sl) == 1:
        ANS.append("Yes")
    if ((sl[0] > a) or (sl[len(sl)-1] > a)):
        ANS.append("No")
        break
print("\n".join(ANS))
```

11 Triabgle Quest 2

```
for i in range(1,int(input())+1): #More than 2 lines will result
in 0 score. Do not leave a blank line also
    print(((10**i)//9)**2)
```

12 Iterable And Iterators

HackerRank | Prepare > Python > Itertools > Iterables and Iterators Exit Full Screen View

Problem

The itertools module standardizes a core set of fast, memory efficient tools that are useful by themselves or in combination. Together, they form an iterator algebra making it possible to construct specialized tools succinctly and efficiently in pure Python.

To read more about the functions in this module, check out their [documentation here](#).

You are given a list of N lowercase English letters. For a given integer K , you can select any K indices (assume 1-based indexing) with a uniform probability from the list.

Find the probability that at least one of the K indices selected will contain the letter: 'a'.

Input Format

The input consists of three lines. The first line contains the integer N , denoting the length of the list. The next line consists of N space-separated lowercase English letters, denoting the elements of the list.

The third and the last line of input contains the integer K , denoting the number of indices to be selected.

Output Format

Output a single line consisting of the probability that at least one of the K indices selected contains the letter 'a'.

Line: 10 Col: 1

[Upload Code as File](#) ☐ Test against custom input [Run Code](#) [Submit Code](#)

Congratulations
You solved this challenge. Would you like to challenge your friends?
[f](#) [t](#) [in](#) [Next Challenge](#)

Test case 0 ☒ [Test case 1](#) ☒ [Test case 2](#) ☒ [Test case 3](#) ☒ [Test case 4](#) ☒ [Test case 5](#) ☒ [Test case 6](#) ☒ [Test case 7](#) ☒ [Test case 8](#) ☒ [Test case 9](#) ☒ [Test case 10](#) ☒ [Test case 11](#) ☒ [Test case 12](#) ☒ [Test case 13](#) ☒ [Test case 14](#) ☒ [Test case 15](#) ☒ [Test case 16](#) ☒ [Test case 17](#) ☒ [Test case 18](#) ☒ [Test case 19](#) ☒ [Test case 20](#) ☒ [Test case 21](#) ☒ [Test case 22](#) ☒ [Test case 23](#) ☒ [Test case 24](#) ☒ [Test case 25](#) ☒ [Test case 26](#) ☒ [Test case 27](#) ☒ [Test case 28](#) ☒ [Test case 29](#) ☒ [Test case 30](#) ☒ [Test case 31](#) ☒ [Test case 32](#) ☒ [Test case 33](#) ☒ [Test case 34](#) ☒ [Test case 35](#) ☒ [Test case 36](#) ☒ [Test case 37](#) ☒ [Test case 38](#) ☒ [Test case 39](#) ☒ [Test case 40](#) ☒ [Test case 41](#) ☒ [Test case 42](#) ☒ [Test case 43](#) ☒ [Test case 44](#) ☒ [Test case 45](#) ☒ [Test case 46](#) ☒ [Test case 47](#) ☒ [Test case 48](#) ☒ [Test case 49](#) ☒ [Test case 50](#) ☒ [Test case 51](#) ☒ [Test case 52](#) ☒ [Test case 53](#) ☒ [Test case 54](#) ☒ [Test case 55](#) ☒ [Test case 56](#) ☒ [Test case 57](#) ☒ [Test case 58](#) ☒ [Test case 59](#) ☒ [Test case 60](#) ☒ [Test case 61](#) ☒ [Test case 62](#) ☒ [Test case 63](#) ☒ [Test case 64](#) ☒ [Test case 65](#) ☒ [Test case 66](#) ☒ [Test case 67](#) ☒ [Test case 68](#) ☒ [Test case 69](#) ☒ [Test case 70](#) ☒ [Test case 71](#) ☒ [Test case 72](#) ☒ [Test case 73](#) ☒ [Test case 74](#) ☒ [Test case 75](#) ☒ [Test case 76](#) ☒ [Test case 77](#) ☒ [Test case 78](#) ☒ [Test case 79](#) ☒ [Test case 80](#) ☒ [Test case 81](#) ☒ [Test case 82](#) ☒ [Test case 83](#) ☒ [Test case 84](#) ☒ [Test case 85](#) ☒ [Test case 86](#) ☒ [Test case 87](#) ☒ [Test case 88](#) ☒ [Test case 89](#) ☒ [Test case 90](#) ☒ [Test case 91](#) ☒ [Test case 92](#) ☒ [Test case 93](#) ☒ [Test case 94](#) ☒ [Test case 95](#) ☒ [Test case 96](#) ☒ [Test case 97](#) ☒ [Test case 98](#) ☒ [Test case 99](#) ☒ [Test case 100](#) ☒ [Test case 101](#) ☒ [Test case 102](#) ☒ [Test case 103](#) ☒ [Test case 104](#) ☒ [Test case 105](#) ☒ [Test case 106](#) ☒ [Test case 107](#) ☒ [Test case 108](#) ☒ [Test case 109](#) ☒ [Test case 110](#) ☒ [Test case 111](#) ☒ [Test case 112](#) ☒ [Test case 113](#) ☒ [Test case 114](#) ☒ [Test case 115](#) ☒ [Test case 116](#) ☒ [Test case 117](#) ☒ [Test case 118](#) ☒ [Test case 119](#) ☒ [Test case 120](#) ☒ [Test case 121](#) ☒ [Test case 122](#) ☒ [Test case 123](#) ☒ [Test case 124](#) ☒ [Test case 125](#) ☒ [Test case 126](#) ☒ [Test case 127](#) ☒ [Test case 128](#) ☒ [Test case 129](#) ☒ [Test case 130](#) ☒ [Test case 131](#) ☒ [Test case 132](#) ☒ [Test case 133](#) ☒ [Test case 134](#) ☒ [Test case 135](#) ☒ [Test case 136](#) ☒ [Test case 137](#) ☒ [Test case 138](#) ☒ [Test case 139](#) ☒ [Test case 140](#) ☒ [Test case 141](#) ☒ [Test case 142](#) ☒ [Test case 143](#) ☒ [Test case 144](#) ☒ [Test case 145](#) ☒ [Test case 146](#) ☒ [Test case 147](#) ☒ [Test case 148](#) ☒ [Test case 149](#) ☒ [Test case 150](#) ☒ [Test case 151](#) ☒ [Test case 152](#) ☒ [Test case 153](#) ☒ [Test case 154](#) ☒ [Test case 155](#) ☒ [Test case 156](#) ☒ [Test case 157](#) ☒ [Test case 158](#) ☒ [Test case 159](#) ☒ [Test case 160](#) ☒ [Test case 161](#) ☒ [Test case 162](#) ☒ [Test case 163](#) ☒ [Test case 164](#) ☒ [Test case 165](#) ☒ [Test case 166](#) ☒ [Test case 167](#) ☒ [Test case 168](#) ☒ [Test case 169](#) ☒ [Test case 170](#) ☒ [Test case 171](#) ☒ [Test case 172](#) ☒ [Test case 173](#) ☒ [Test case 174](#) ☒ [Test case 175](#) ☒ [Test case 176](#) ☒ [Test case 177](#) ☒ [Test case 178](#) ☒ [Test case 179](#) ☒ [Test case 180](#) ☒ [Test case 181](#) ☒ [Test case 182](#) ☒ [Test case 183](#) ☒ [Test case 184](#) ☒ [Test case 185](#) ☒ [Test case 186](#) ☒ [Test case 187](#) ☒ [Test case 188](#) ☒ [Test case 189](#) ☒ [Test case 190](#) ☒ [Test case 191](#) ☒ [Test case 192](#) ☒ [Test case 193](#) ☒ [Test case 194](#) ☒ [Test case 195](#) ☒ [Test case 196](#) ☒ [Test case 197](#) ☒ [Test case 198](#) ☒ [Test case 199](#) ☒ [Test case 200](#) ☒ [Test case 201](#) ☒ [Test case 202](#) ☒ [Test case 203](#) ☒ [Test case 204](#) ☒ [Test case 205](#) ☒ [Test case 206](#) ☒ [Test case 207](#) ☒ [Test case 208](#) ☒ [Test case 209](#) ☒ [Test case 210](#) ☒ [Test case 211](#) ☒ [Test case 212](#) ☒ [Test case 213](#) ☒ [Test case 214](#) ☒ [Test case 215](#) ☒ [Test case 216](#) ☒ [Test case 217](#) ☒ [Test case 218](#) ☒ [Test case 219](#) ☒ [Test case 220](#) ☒ [Test case 221](#) ☒ [Test case 222](#) ☒ [Test case 223](#) ☒ [Test case 224](#) ☒ [Test case 225](#) ☒ [Test case 226](#) ☒ [Test case 227](#) ☒ [Test case 228](#) ☒ [Test case 229](#) ☒ [Test case 230](#) ☒ [Test case 231](#) ☒ [Test case 232](#) ☒ [Test case 233](#) ☒ [Test case 234](#) ☒ [Test case 235](#) ☒ [Test case 236](#) ☒ [Test case 237](#) ☒ [Test case 238](#) ☒ [Test case 239](#) ☒ [Test case 240](#) ☒ [Test case 241](#) ☒ [Test case 242](#) ☒ [Test case 243](#) ☒ [Test case 244](#) ☒ [Test case 245](#) ☒ [Test case 246](#) ☒ [Test case 247](#) ☒ [Test case 248](#) ☒ [Test case 249](#) ☒ [Test case 250](#) ☒ [Test case 251](#) ☒ [Test case 252](#) ☒ [Test case 253](#) ☒ [Test case 254](#) ☒ [Test case 255](#) ☒ [Test case 256](#) ☒ [Test case 257](#) ☒ [Test case 258](#) ☒ [Test case 259](#) ☒ [Test case 260](#) ☒ [Test case 261](#) ☒ [Test case 262](#) ☒ [Test case 263](#) ☒ [Test case 264](#) ☒ [Test case 265](#) ☒ [Test case 266](#) ☒ [Test case 267](#) ☒ [Test case 268](#) ☒ [Test case 269](#) ☒ [Test case 270](#) ☒ [Test case 271](#) ☒ [Test case 272](#) ☒ [Test case 273](#) ☒ [Test case 274](#) ☒ [Test case 275](#) ☒ [Test case 276](#) ☒ [Test case 277](#) ☒ [Test case 278](#) ☒ [Test case 279](#) ☒ [Test case 280](#) ☒ [Test case 281](#) ☒ [Test case 282](#) ☒ [Test case 283](#) ☒ [Test case 284](#) ☒ [Test case 285](#) ☒ [Test case 286](#) ☒ [Test case 287](#) ☒ [Test case 288](#) ☒ [Test case 289](#) ☒ [Test case 290](#) ☒ [Test case 291](#) ☒ [Test case 292](#) ☒ [Test case 293](#) ☒ [Test case 294](#) ☒ [Test case 295](#) ☒ [Test case 296](#) ☒ [Test case 297](#) ☒ [Test case 298](#) ☒ [Test case 299](#) ☒ [Test case 300](#) ☒ [Test case 301](#) ☒ [Test case 302](#) ☒ [Test case 303](#) ☒ [Test case 304](#) ☒ [Test case 305](#) ☒ [Test case 306](#) ☒ [Test case 307](#) ☒ [Test case 308](#) ☒ [Test case 309](#) ☒ [Test case 310](#) ☒ [Test case 311](#) ☒ [Test case 312](#) ☒ [Test case 313](#) ☒ [Test case 314](#) ☒ [Test case 315](#) ☒ [Test case 316](#) ☒ [Test case 317](#) ☒ [Test case 318](#) ☒ [Test case 319](#) ☒ [Test case 320](#) ☒ [Test case 321](#) ☒ [Test case 322](#) ☒ [Test case 323](#) ☒ [Test case 324](#) ☒ [Test case 325](#) ☒ [Test case 326](#) ☒ [Test case 327](#) ☒ [Test case 328](#) ☒ [Test case 329](#) ☒ [Test case 330](#) ☒ [Test case 331](#) ☒ [Test case 332](#) ☒ [Test case 333](#) ☒ [Test case 334](#) ☒ [Test case 335](#) ☒ [Test case 336](#) ☒ [Test case 337](#) ☒ [Test case 338](#) ☒ [Test case 339](#) ☒ [Test case 340](#) ☒ [Test case 341](#) ☒ [Test case 342](#) ☒ [Test case 343](#) ☒ [Test case 344](#) ☒ [Test case 345](#) ☒ [Test case 346](#) ☒ [Test case 347](#) ☒ [Test case 348](#) ☒ [Test case 349](#) ☒ [Test case 350](#) ☒ [Test case 351](#) ☒ [Test case 352](#) ☒ [Test case 353](#) ☒ [Test case 354](#) ☒ [Test case 355](#) ☒ [Test case 356](#) ☒ [Test case 357](#) ☒ [Test case 358](#) ☒ [Test case 359](#) ☒ [Test case 360](#) ☒ [Test case 361](#) ☒ [Test case 362](#) ☒ [Test case 363](#) ☒ [Test case 364](#) ☒ [Test case 365](#) ☒ [Test case 366](#) ☒ [Test case 367](#) ☒ [Test case 368](#) ☒ [Test case 369](#) ☒ [Test case 370](#) ☒ [Test case 371](#) ☒ [Test case 372](#) ☒ [Test case 373](#) ☒ [Test case 374](#) ☒ [Test case 375](#) ☒ [Test case 376](#) ☒ [Test case 377](#) ☒ [Test case 378](#) ☒ [Test case 379](#) ☒ [Test case 380](#) ☒ [Test case 381](#) ☒ [Test case 382](#) ☒ [Test case 383](#) ☒ [Test case 384](#) ☒ [Test case 385](#) ☒ [Test case 386](#) ☒ [Test case 387](#) ☒ [Test case 388](#) ☒ [Test case 389](#) ☒ [Test case 390](#) ☒ [Test case 391](#) ☒ [Test case 392](#) ☒ [Test case 393](#) ☒ [Test case 394](#) ☒ [Test case 395](#) ☒ [Test case 396](#) ☒ [Test case 397](#) ☒ [Test case 398](#) ☒ [Test case 399](#) ☒ [Test case 400](#) ☒ [Test case 401](#) ☒ [Test case 402](#) ☒ [Test case 403](#) ☒ [Test case 404](#) ☒ [Test case 405](#) ☒ [Test case 406](#) ☒ [Test case 407](#) ☒ [Test case 408](#) ☒ [Test case 409](#) ☒ [Test case 410](#) ☒ [Test case 411](#) ☒ [Test case 412](#) ☒ [Test case 413](#) ☒ [Test case 414](#) ☒ [Test case 415](#) ☒ [Test case 416](#) ☒ [Test case 417](#) ☒ [Test case 418](#) ☒ [Test case 419](#) ☒ [Test case 420](#) ☒ [Test case 421](#) ☒ [Test case 422](#) ☒ [Test case 423](#) ☒ [Test case 424](#) ☒ [Test case 425](#) ☒ [Test case 426](#) ☒ [Test case 427](#) ☒ [Test case 428](#) ☒ [Test case 429](#) ☒ [Test case 430](#) ☒ [Test case 431](#) ☒ [Test case 432](#) ☒ [Test case 433](#) ☒ [Test case 434](#) ☒ [Test case 435](#) ☒ [Test case 436](#) ☒ [Test case 437](#) ☒ [Test case 438](#) ☒ [Test case 439](#) ☒ [Test case 440](#) ☒ [Test case 441](#) ☒ [Test case 442](#) ☒ [Test case 443](#) ☒ [Test case 444](#) ☒ [Test case 445](#) ☒ [Test case 446](#) ☒ [Test case 447](#) ☒ [Test case 448](#) ☒ [Test case 449](#) ☒ [Test case 450](#) ☒ [Test case 451](#) ☒ [Test case 452](#) ☒ [Test case 453](#) ☒ [Test case 454](#) ☒ [Test case 455](#) ☒ [Test case 456](#) ☒ [Test case 457](#) ☒ [Test case 458](#) ☒ [Test case 459](#) ☒ [Test case 460](#) ☒ [Test case 461](#) ☒ [Test case 462](#) ☒ [Test case 463](#) ☒ [Test case 464](#) ☒ [Test case 465](#) ☒ [Test case 466](#) ☒ [Test case 467](#) ☒ [Test case 468](#) ☒ [Test case 469](#) ☒ [Test case 470](#) ☒ [Test case 471](#) ☒ [Test case 472](#) ☒ [Test case 473](#) ☒ [Test case 474](#) ☒ [Test case 475](#) ☒ [Test case 476](#) ☒ [Test case 477](#) ☒ [Test case 478](#) ☒ [Test case 479](#) ☒ [Test case 480](#) ☒ [Test case 481](#) ☒ [Test case 482](#) ☒ [Test case 483](#) ☒ [Test case 484](#) ☒ [Test case 485](#) ☒ [Test case 486](#) ☒ [Test case 487](#) ☒ [Test case 488](#) ☒ [Test case 489](#) ☒ [Test case 490](#) ☒ [Test case 491](#) ☒ [Test case 492](#) ☒ [Test case 493](#) ☒ [Test case 494](#) ☒ [Test case 495](#) ☒ [Test case 496](#) ☒ [Test case 497](#) ☒ [Test case 498](#) ☒ [Test case 499](#) ☒ [Test case 500](#) ☒ [Test case 501](#) ☒ [Test case 502](#) ☒ [Test case 503](#) ☒ [Test case 504](#) ☒ [Test case 505](#) ☒ [Test case 506](#) ☒ [Test case 507](#) ☒ [Test case 508](#) ☒ [Test case 509](#) ☒ [Test case 510](#) ☒ [Test case 511](#) ☒ [Test case 512](#) ☒ [Test case 513](#) ☒ [Test case 514](#) ☒ [Test case 515](#) ☒ [Test case 516](#) ☒ [Test case 517](#) ☒ [Test case 518](#) ☒ [Test case 519](#) ☒ [Test case 520](#) ☒ [Test case 521](#) ☒ [Test case 522](#) ☒ [Test case 523](#) ☒ [Test case 524](#) ☒ [Test case 525](#) ☒ [Test case 526](#) ☒ [Test case 527](#) ☒ [Test case 528](#) ☒ [Test case 529](#) ☒ [Test case 530](#) ☒ [Test case 531](#) ☒ [Test case 532](#) ☒ [Test case 533](#) ☒ [Test case 534](#) ☒ [Test case 535](#) ☒ [Test case 536](#) ☒ [Test case 537](#) ☒ [Test case 538](#) ☒ [Test case 539](#) ☒ [Test case 540](#) ☒ [Test case 541](#) ☒ [Test case 542](#) ☒ [Test case 543](#) ☒ [Test case 544](#) ☒ [Test case 545](#) ☒ [Test case 546](#) ☒ [Test case 547](#) ☒ [Test case 548](#) ☒ [Test case 549](#) ☒ [Test case 550](#) ☒ [Test case 551](#) ☒ [Test case 552](#) ☒ [Test case 553](#) ☒ [Test case 554](#) ☒ [Test case 555](#) ☒ [Test case 556](#) ☒ [Test case 557](#) ☒ [Test case 558](#) ☒ [Test case 559](#) ☒ [Test case 560](#) ☒ [Test case 561](#) ☒ [Test case 562](#) ☒ [Test case 563](#) ☒ [Test case 564](#) ☒ [Test case 565](#) ☒ [Test case 566](#) ☒ [Test case 567](#) ☒ [Test case 568](#) ☒ [Test case 569](#) ☒ [Test case 570](#) ☒

14 Clases: Dealing with complex number

HackerRank | Prepare > Python > Classes > Classes: Dealing with Complex Numbers

Exit Full Screen View

Submissions

- $C + D$
- $C - D$
- $C * D$
- C / D
- $\text{mod}(C)$
- $\text{mod}(D)$

Leaderboard

For complex numbers with non-zero real (A) and complex part (B), the output should be in the following format:
 $A + Bi$
Replace the plus symbol (+) with a minus symbol (-) when $B < 0$.

For complex numbers with a zero complex part i.e. real numbers, the output should be:
 $A + 0.00i$

For complex numbers where the real part is zero and the complex part (B) is non-zero, the output should be:
 $0.00 + Bi$

Sample Input

```
2 1
5 6
```

Sample Output

```
2 + 1i
5 + 6i
```

```
15 def truediv(self, no):
16     conjugate = Complex(no.real, (-no.imaginary))
17     num = self*conjugate
18     denom = no*conjugate
19     try:
20         return Complex((num.real/denom.real), (num.imaginary/
```

Line: 2 Col: 1

Upload Code as File ☐ Test against custom input **Run Code** **Submit Code**

Congratulations
You solved this challenge. Would you like to challenge your friends?
[f](#) [t](#) [in](#) **Next Challenge**

☒ Test case 0
☒ Test case 1
☒ Test case 2

Compiler Message
Success

Input (stdin)
Download

```
import math
```

```
class Complex(object):
    def __init__(self, real, imaginary):
        self.real = real
        self.imaginary = imaginary
    def __add__(self, no):
        return Complex((self.real+no.real), self.imaginary+no.imaginary)
    def __sub__(self, no):
        return Complex((self.real-no.real), (self.imaginary-no.imaginary))
    def __mul__(self, no):
        r = (self.real*no.real)-(self.imaginary*no.imaginary)
        i = (self.real*no.imaginary+no.real*self.imaginary)
        return Complex(r, i)
    def __truediv__(self, no):
        conjugate = Complex(no.real, (-no.imaginary))
        num = self*conjugate
        denom = no*conjugate
        try:
            return Complex((num.real/denom.real), (num.imaginary/denom.real))
        except Exception as e:
            print(e)
    def mod(self):
        m = math.sqrt(self.real**2+self.imaginary**2)
        return Complex(m, 0)
```

```

def __str__(self):
    if self.imaginary == 0:
        result = "%.2f+0.00i" % (self.real)
    elif self.real == 0:
        if self.imaginary >= 0:
            result = "0.00+%.2fi" % (self.imaginary)
        else:
            result = "0.00-%.2fi" % (abs(self.imaginary))
    elif self.imaginary > 0:
        result = "%.2f+%.2fi" % (self.real, self.imaginary)
    else:
        result = "%.2f-
%.2fi" % (self.real, abs(self.imaginary))
    return result

if __name__ == '__main__':
    c = map(float, input().split())
    d = map(float, input().split())
    x = Complex(*c)
    y = Complex(*d)
    print(*map(str, [x+y, x-
y, x*y, x/y, x.mod(), y.mod()]), sep='\n')

```

15 Athlete Sort

HackerRank | Prepare > Python > Built-Ins > Athlete Sort

Exit Full Screen View

Problem

You are given a spreadsheet that contains a list of N athletes and their details (such as age, height, weight and so on). You are required to sort the data based on the K^{th} attribute and print the final resulting table. Follow the example given below for better understanding.

Rank	Age	Height (in cm)	Rank	Age	Height (in cm)
1	32	190	5	24	176
2	35	175	4	26	195
3	41	188	1	32	190
4	26	195	2	35	175
5	24	176	3	41	188

Note that K is indexed from 0 to $M - 1$, where M is the number of attributes.

Note: If two attributes are the same for different rows, for example, if two athletes are of the same age, print the row that appeared first in the input.

Input Format

The first line contains N and M separated by a space.
The next N lines each contain M elements.
The last line contains K .

```

10 rows = [input() for _ in range(N)]
11 K = int(input())
12 for row in sorted(rows, key=lambda row: int(row.split()[K])):
13     print(row)
14

```

Line: 14 Col: 1

☐ Test against custom input

Congratulations

You solved this challenge. Would you like to challenge your friends?

Test case 0

```
#!/bin/python3
```

```

import math
import os
import random

```

```

import re
import sys

N, M = map(int, input().split())
rows = [input() for _ in range(N)]
K = int(input())
for row in sorted(rows, key=lambda row: int(row.split()[K])):
    print(row)

```

16 Ginorts

Problem

You are given a string S .
 S contains alphanumeric characters only.

Sorting

Your task is to sort the string S in the following manner:

- All sorted lowercase letters are ahead of uppercase letters.
- All sorted uppercase letters are ahead of digits.
- All sorted odd digits are ahead of sorted even digits.

Input Format

A single line of input contains the string S .

Constraints

- $0 < \text{len}(S) < 1000$

Output Format

Output the sorted string S .

Sample Input

```
02468
```

Test case 0 Success

```

# Enter your code here. Read input from STDIN. Print output to ST
DOUT
print(*sorted(input(), key=lambda c: (c.isdigit() - c.islower(),
c in '02468', c)), sep='')

```

17 validate email address with filter

```

def fun(email):
    try:
        username, url = email.split('@')
        website, extension = url.split('.')
    except ValueError:
        return False
    if username.replace('-',
', ').replace('_', ' ').isalnum() is False:
        return False
    elif website.isalnum() is False:
        return False
    elif len(extension) > 3:
        return False

```

```

        else:
            return True
def filter_mail(emails):
    return list(filter(fun, emails))

if __name__ == '__main__':
    n = int(input())
    emails = []
    for _ in range(n):
        emails.append(input())

filtered_emails = filter_mail(emails)
filtered_emails.sort()
print(filtered_emails)

```

The screenshot shows the HackerRank interface for the challenge 'Validating Email Addresses With a Filter'. The left sidebar contains links for 'Problem', 'Submissions', and 'Leaderboard'. The main content area includes the problem description, rules for valid email addresses, and a 'Concept' section explaining the filter function. On the right, there is a code editor with a 'Submit Code' button. Below the editor, a green 'Congratulations' banner indicates the user has solved the challenge. A 'Test case 0' section shows the compiler message 'Success'.

18 Reduce function

```

from fractions import Fraction
from functools import reduce
def product(fracs):
    t = Fraction(reduce(lambda x, y: x * y, fracs))
    return t.numerator, t.denominator

if __name__ == '__main__':
    fracs = []
    for _ in range(int(input())):
        fracs.append(Fraction(*map(int, input().split())))
    result = product(fracs)
    print(*result)

```

H HackerRank | Prepare > Python > Python Functionals > Reduce Function

Exit Full Screen View

Problem

Given a list of rational numbers, find their product.

Concept

The reduce() function applies a function of two arguments cumulatively on a list of objects in succession from left to right to reduce it to one value. Say you have a list, say [1,2,3] and you have to find its sum.

```
>>> reduce(lambda x, y : x + y,[1,2,3])  
6
```

You can also define an initial value. If it is specified, the function will assume initial value as the value given, and then reduce. It is equivalent to adding the initial value at the beginning of the list. For example:

```
>>> reduce(lambda x, y : x + y, [1,2,3], -3)  
3  
  
>>> from fractions import gcd  
>>> reduce(gcd, [2,4,8], 3)  
1
```

Input Format

This challenge has 1 input.

Line: 3 Col: 1

Upload Code as File

☐ Test against custom input

Run Code

Submit Code

Congratulations

You solved this challenge. Would you like to challenge your friends?

f t in

Next Challenge

Test case 0

Test case 1

Test case 2

Compiler Message

Success

HackerRank

[Prepare](#)
[Python](#)
[Regex and Parsing](#)
[Regex Substitution](#)

[Exit Full Screen View](#)

Problem

The `re.sub()` tool (sub stands for substitution) evaluates a pattern and, for each valid match, it calls a method (or lambda).

The method is called for all matches and can be used to modify strings in different ways.

The `re.sub()` method returns the modified string as an output.

Learn more about `re.sub()`.

Transformation of Strings

Code

```
import re

#Squaring numbers
def square(match):
    number = int(match.group(0))
    return str(number**2)

print re.sub(r"\d+", square, "1 2 3 4 5 6 7 8 9")
```

Output

```
1 4 9 16 25 36 49 64 81
```

Submissions

Leaderboard

Ins

Line: 5 Col: 1

Upload Code as File

☐ Test against custom input

Run Code

Submit Code

Congratulations

You solved this challenge. Would you like to challenge your friends?

Next Challenge

Test case 0

Test case 1

Test case 2

Compiler Message

Success

Input (stdin)

Download

20 Validating credit card number

HackerRank | Prepare > Python > Regex and Parsing > Validating Credit Card Numbers Exit Full Screen View

Problem

You and Fredrick are good friends. Yesterday, Fredrick received N credit cards from **ABCD Bank**. He wants to verify whether his credit card numbers are valid or not. You happen to be great at regex so he is asking for your help!

A valid credit card from **ABCD Bank** has the following characteristics:

- ▶ It must start with a 4, 5 or 6.
- ▶ It must contain exactly 16 digits.
- ▶ It must only consist of digits (0-9).
- ▶ It may have digits in groups of 4, separated by one hyphen "-".
- ▶ It must NOT use any other separator like ' ', '_', etc.
- ▶ It must NOT have 4 or more consecutive repeated digits.

Examples:

Valid Credit Card Numbers

```
4253625879615786
4424424424442444
5122-2368-7954-3214
```

Invalid Credit Card Numbers

```
14 | else:
15 |     valid = False
16 |     if valid == True:
17 |         print('Valid')
18 |     else:
19 |         print('Invalid')
```

Line: 20 Col: 1

☐ Test against custom input

Congratulations

You solved this challenge. Would you like to challenge your friends?

[f](#) [t](#) [in](#)

Test case 0 ☒ Compiler Message

Test case 1 ☒ Success

Test case 2 ☒ Input (stdin)

Enter your code here. Read input from STDIN. Print output to STDOUT

```
import re
n = int(input())
for t in range(n):
    credit = input().strip()
    credit_removed_hiphen = credit.replace('-', '')
    valid = True
    length_16 = bool(re.match(r'^[4-6]\d{15}$', credit))
    length_19 = bool(re.match(r'^[4-6]\d{3}-\d{4}-\d{4}-\d{4}$', credit))
    consecutive = bool(re.findall(r'(?=(\d)\1\1\1)', credit_removed_hiphen))
    if length_16 == True or length_19 == True:
        if consecutive == True:
            valid=False
    else:
        valid = False
    if valid == True:
        print('Valid')
    else:
        print('Invalid')
```

21 Words score

```
def is_vowel(letter):
    return letter in ['a', 'e', 'i', 'o', 'u', 'y']
def is_vowel(letter):
    return letter in ['a', 'e', 'i', 'o', 'u', 'y']
def score_words(words):
```

```

score = 0
for word in words:
    num_vowels = 0
    for letter in word:
        if is_vowel(letter):
            num_vowels += 1
    if num_vowels % 2 == 0:
        score += 2
    else:
        score += 1
return score

n = int(input())
words = input().split()
print(score_words(words))

```

The screenshot shows the HackerRank interface for a challenge titled "Words Score". The left sidebar contains links for "Problem", "Submissions", "Leaderboard", and "Stats". The main content area on the left provides the problem description: the task is to debug the existing code to successfully execute all provided test files. It defines vowels as 'a, e, i, o, u' and 'y'. The function `score_words` takes a list of lowercase words and returns a score. The score of a single word is 2 if it contains an even number of vowels, otherwise it is 1. The total score is the sum of scores for all words. The input format is a single integer `n` followed by `n` space-separated lowercase words. Constraints are $1 \leq n \leq 20$.

The right sidebar shows the code editor with the following code:

```

16 | return score
17 |
18 | > ...

```

Below the code editor are buttons for "Upload Code as File", "Test against custom input", "Run Code", and "Submit Code". A large green banner displays "Congratulations" and "You solved this challenge. Would you like to challenge your friends?" with social media icons and a "Next Challenge" button. At the bottom, a "Test Results" section shows "Test case 0" and "Test case 1" both passing, with a "Success" compiler message and "Input (stdin)" field.

22 Default arguments

```

class EvenStream(object):
    def __init__(self):
        self.current = 0

    def get_next(self):
        to_return = self.current
        self.current += 2
        return to_return

class OddStream(object):

```



```

def __init__(self):
    self.current = 1

def get_next(self):
    to_return = self.current
    self.current += 2
    return to_return

def print_from_stream(n, stream=EvenStream()):
    stream.__init__()
    for _ in range(n):
        print(stream.get_next())

queries = int(input())
for _ in range(queries):
    stream_name, n = input().split()
    n = int(n)
    if stream_name == "even":
        print_from_stream(n)
    else:
        print_from_stream(n, OddStream())

```

HackerRank | Prepare > Python > Debugging > Default Arguments

Exit Full Screen View

Problem

In this challenge, the task is to debug the existing code to successfully execute all provided test files.

Python supports a useful concept of default argument values. For each keyword argument of a function, we can assign a default value which is going to be used as the value of said argument if the function is called without it. For example, consider the following increment function:

```
def increment_by(n, increment=1):
    return n + increment
```

The function works like this:

```
>>> increment_by(5, 2)
7
>>> increment_by(4)
5
>>>
```

Debug the given function `print_from_stream` using the default value of one of its arguments.

The function has the following signature:

```
def print_from_stream(n, stream=EvenStream()):
```

Submissions

Upload Code as File ☐ Test against custom input **Run Code** **Submit Code**

Line: 18 Col: 1

Congratulations

You solved this challenge. Would you like to challenge your friends?

Next Challenge

Test case 0 ☒ Compiler Message

Test case 1 ☒ Success

Test case 2 ☒ Input (stdin) Download

23 Maximize it

```

# Enter your code here. Read input from STDIN. Print output to ST
DOUT
import itertools
NUMBER_OF_LISTS, MODULUS = map(int, input().split())
LISTS_OF_LISTS = []
for i in range(0, NUMBER_OF_LISTS):
    new_list = list(map(int, input().split()))

```

```

del new_list[0]
LISTS_OF_LISTS.append(new_list)
def squared(element):
    return element**2
COMBS = list(itertools.product(*LISTS_OF_LISTS))
RESULTS = []
for i in COMBS:
    result1 = sum(map(squared, [a for a in i]))
    result2 = result1 % MODULUS
    RESULTS.append(result2)
print(max(RESULTS))

```

The screenshot shows the HackerRank interface for a challenge titled 'Maximize It!'. The problem description on the left states: 'You are given a function $f(X) = X^2$. You are also given K lists. The i^{th} list consists of N_i elements. You have to pick one element from each list so that the value from the equation below is maximized: $S = (f(X_1) + f(X_2) + \dots + f(X_k)) \% M$. X_i denotes the element picked from the i^{th} list. Find the maximized value S_{max} obtained. $\%$ denotes the modulo operator. Note that you need to take exactly one element from each list, not necessarily the largest element. You add the squares of the chosen elements and perform the modulo operation. The maximum value that you can obtain, will be the answer to the problem.'

The code editor on the right contains the following Python code:

```

14 result1 = sum(map(squared, [a for a in i]))
15 result2 = result1 % MODULUS
16 RESULTS.append(result2)
17 print(max(RESULTS))
18

```

Below the code editor, there is a 'Congratulations' message: 'You solved this challenge. Would you like to challenge your friends?' with social media icons for Facebook, Twitter, and LinkedIn. A 'Next Challenge' button is also present. At the bottom, the test cases are listed: 'Test case 0' and 'Test case 1', both marked as successful. The compiler message shows 'Success'.

24 Validating postal code

```

regex_integer_in_range = r"^[1-9][\d]{5}$" # Do not delete 'r'.
regex_alternating_repetitive_digit_pair = r"(\d)(?=\d\1)" # Do not delete 'r'.

import re
P = input()

print (bool(re.match(regex_integer_in_range, P))
and len(re.findall(regex_alternating_repetitive_digit_pair, P)) <
2)

```

HackerRank

Prepare

Python

Regex and Parsing

Validating Postal Codes

Exit Full Screen View

Problem

A valid postal code P have to fulfill both below requirements:

- P must be a number in the range from **100000** to **999999** inclusive.
- P must not contain more than one alternating repetitive digit pair.

Alternating repetitive digits are digits which repeat immediately after the next digit. In other words, an alternating repetitive digit pair is formed by two equal digits that have just a single digit between them.

For example:

```
121426 # Here, 1 is an alternating repetitive digit.
523563 # Here, NO digit is an alternating repetitive d
552523 # Here, both 2 and 5 are alternating repetitive
```

Your task is to provide two regular expressions

`regex_integer_in_range` and

`regex_alternating_repetitive_digit_pair`. Where:

`regex_integer_in_range` should match only integers range from **100000** to **999999** inclusive

`regex_alternating_repetitive_digit_pair` should find alternating repetitive digits pairs in a given string.

Both these regular expressions will be used by the provided code

Submissions

Leaderboard

ans

Line: 1 Col: 1

Upload Code as File

Test against custom input

Run Code

Submit Code

Congratulations

You solved this challenge. Would you like to challenge your friends?

Next Challenge

Test case 0

Compiler Message

Success

Test case 1

Input (stdin)

Download

25 Matrix script

HackerRank

Prepare

Python

Regex and Parsing

Matrix Script

Exit Full Screen View

Problem

Neo has a complex matrix script. The matrix script is a $N \times M$ grid of strings. It consists of alphanumeric characters, spaces and symbols (`!@#$.%&.`).

Matrix Script

```
T       s       i
h       %       x
i       .       #
s       M
$       a
#       t       %
i       r       l
```

Matrix Decoded

```
This$#is% Matrix# %l
```

To decode the script, Neo needs to read each column and select only the alphanumeric characters and connect them. Neo reads the column from top to bottom and starts reading from the leftmost column.

If there are symbols or spaces between two alphanumeric characters of

Submissions

Leaderboard

ans

Line: 11 Col: 1

Upload Code as File

Test against custom input

Run Code

Submit Code

Congratulations

You solved this challenge. Would you like to challenge your friends?

Next Challenge

Test case 0

Compiler Message

Success

Test case 1

Input (stdin)

Download

```
import re
n, m = map(int, input().split())
character_ar = [''] * (n*m)
for i in range(n):
    line = input()
    for j in range(m):
        character_ar[i+(j*n)] = line[j]
decoded_str = ''.join(character_ar)
final_decoded_str = re.sub(r'(?<=[A-Za-z0-9])([ !@#$%&]+)(?=[A-Za-z0-9])', ' ', decoded_str)
```

```
print(final_decoded_str)
```