

Extra Credit: Solving HackerRank problem using LLM

Easy Problem:

Problem

You are asked to ensure that the first and last names of people begin with a capital letter in their passports. For example, `alison heck` should be capitalised correctly as `Alison Heck`.

`alison heck` \Rightarrow `Alison Heck`

Given a full name, your task is to capitalize the name appropriately.

Input Format

A single line of input containing the full name, S .

Constraints

- $0 < \text{len}(S) < 1000$
- The string consists of alphanumeric characters and spaces.

Note: in a word only the first character is capitalized. Example `12abc` when capitalized remains `12abc`.

Output Format

Print the capitalized string, S .

Sample Input

`chris alan`

Sample Output

`Chris Alan`

Submissions

Leaderboard

Discussions

Base Prompt Code :

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

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

```
# Complete the solve function below.
```

```
def solve(s):
    return " ".join([word.capitalize() for word in s.split(" ")])
```

```

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

    s = input()

    result = solve(s)

    fptr.write(result + '\n')

    fptr.close()

```

Congratulations

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

Next Challenge

<div> Test case 0 </div>	Compiler Message Success	
<div> Test case 1 </div>		
<div> Test case 2 </div>	Input (stdin) 1 hello world	Download
<div> Test case 3 </div>		
<div> Test case 4 </div>	Expected Output 1 Hello World	Download
<div> Test case 5 </div>		

Improved Prompt:

The “Perfect Prompt”

“Write the most efficient, clean, and production-grade Python solution to capitalize the first letter of each word in a full name string (like for passport formatting). The solution must:

- Handle edge cases such as multiple spaces, numeric prefixes (e.g., **12abc**), and empty input gracefully.*
- Preserve the exact spacing between words.*

- *Use Python best practices and minimal complexity.*
- *Explain why your approach is optimal compared to alternatives.*
- *Provide the final complete code in the Hackerrank-style boilerplate with a `solve(s)` function, ready to run.*
- *Keep the code simple, efficient ($O(n)$ time, $O(n)$ space), and pythonic."*

Improved Code:

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

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

```
# Complete the solve function below.
```

```
def solve(s):
    # Split by spaces to preserve exact spacing
    words = s.split(" ")
    # Capitalize each word safely
    capitalized_words = [w.capitalize() if w else "" for w in words]
    # Join back with spaces exactly as input had
    return " ".join(capitalized_words)
```

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

```
    s = input()
```

```
    result = solve(s)
```

```
    fptr.write(result + '\n')
```

```
    fptr.close()
```

Congratulations

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

[Next Challenge](#)

✓ Test case 0

Compiler Message

Success

✓ Test case 1

✓ Test case 2

Input (stdin)

1 **hello world**

[Download](#)

✓ Test case 3

Expected Output

1 **Hello World**

[Download](#)

✓ Test case 4

✓ Test case 5

=====

Medium 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	<div>sort based on k=1</div> <div>→</div> <div>i.e (age)</div>	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 .

Constraints

$1 \leq N, M \leq 1000$

$0 \leq K < M$

Each element ≤ 1000

Output Format

Print the N lines of the sorted table. Each line should contain the space separated elements. Check the sample below for clarity.

Sample Input 0

```
5 3
10 2 5
7 1 0
9 9 9
1 23 12
```

<https://www.hackerrank.com/challenges/python-sort-sort/problem>

Simple Prompt Code:

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

```
import math
import os
```

```
import random
import re
import sys

def solve():
    # Read n and m
    n, m = map(int, input().split())

    # Read the table (list of lists)
    table = [list(map(int, input().split())) for _ in range(n)]

    # Read k (column index to sort by)
    k = int(input())

    # Sort by the k-th column, stable sort ensures order preserved for ties
    table.sort(key=lambda row: row[k])

    # Print the sorted table
    for row in table:
        print(" ".join(map(str, row)))

if __name__ == '__main__':
    solve()
```

Output:

Wrong Answer :(

1/1 test case failed

✖ Sample Test case 0

Compiler Message

Wrong Answer

Input (stdin)

15 3

210 2 5

37 1 0

49 9 9

51 23 12

66 5 9

71

Your Output (stdout)

Download

Improved Prompt:

The Perfect Prompt

Task: Provide a production-grade Python solution for HackerRank's *Athlete Sort* using the exact boilerplate below.

Goal: Sort $n \times m$ integer rows by the k -th column (0-indexed), ascending, preserving original order for ties (stable sort).

Boilerplate (do not change input reading):

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

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

```
if __name__ == '__main__':
    nm = input().split()
```

```

n = int(nm[0])

m = int(nm[1])

arr = []

for _ in range(n):
    arr.append(list(map(int, input().rstrip().split())))

k = int(input())

```

Requirements:

- Use Python's stable sort (`list.sort` or `sorted`) with `key=lambda row: row[k]`.
- No extra prints/logging — only print the sorted rows, one per line, space-separated.
- Do not modify how input is read; place the sorting + printing logic after `k` is read.
- Time complexity: $O(n \log n)$; Space: $O(1)$ extra (in-place sort) or $O(n)$ if using `sorted`.
- Include a short explanation of stability and why this approach is optimal.
- Provide the final, paste-ready code using the boilerplate (no helper functions needed).

Acceptance criteria:

- Matches the sample I/O exactly.
- Preserves row order for equal `k` values.
- No trailing spaces, no blank lines.
- Works for any valid `n`, `m`, `k` per constraints.

Deliverables:

1. 3–5 sentence explanation.
2. Final code block ready to submit.

Improved Code:

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



```

import math
import os
import random
import re
import sys

if __name__ == '__main__':
    nm = input().split()

    n = int(nm[0])

    m = int(nm[1])

    arr = []

    for _ in range(n):
        arr.append(list(map(int, input().rstrip().split())))

    k = int(input())

    # In-place, stable sort by k-th column
    arr.sort(key=lambda row: row[k])

    # Print rows space-separated, no extra spaces/lines
    for row in arr:
        print(*row)

```

Output:

Congratulations

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

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

Next Challenge

Test case 0

Compiler Message
Success

Test case 1

Input (stdin)
Download

```

1  5 3
2 10 2 5
3  7 1 0
4  9 9 9
5  1 23 12
6  6 5 9
7  1

```

Expected Output
Download

=====

Hard Question :

Problem

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

1. P must be a number in the range from 100000 to 999999 inclusive.
2. 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 digit.
552523 # Here, both 2 and 5 are alternating repetitive digits.
```

Submissions

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 template to check if the input string P is a valid postal code using the following expression:

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

Leaderboard

Discussions

Input Format

Locked stub code in the editor reads a single string denoting P from stdin and uses provided expression and your regular expressions to validate if P is a valid postal code.

Output Format

You are not responsible for printing anything to stdout. Locked stub code in the editor does that.

Editorial

Sample Input 0

```
110000
```

Sample Output 0

```
False
```

Explanation 0

Simple Code:

```
import re

P = input().strip()

regex_integer_in_range = r'^[1-9]\d{5}$'
regex_alternating_repetitive_digit_pair = r'(?=(\d)\d\1)'

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

Output :

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

Ask your friends for help: [f](#) [t](#) [in](#)

✖ Test case 1

✖ Test case 3

✖ Test case 4

✔ Test case 0

✔ Test case 2

✔ Test case 5

✔ Test case 6

Compiler Message

Wrong Answer

Hidden Test Case

Unlock this testcase for 5 hackos.

Unlock

Improved Prompt :

The Perfect Prompt

Task: Give me production-ready Python regular expressions only (no `if` logic) to validate a postal code `P` per HackerRank's "Validating Postal Codes" rules.

Rules to enforce (simultaneously):

1. `P` must be exactly six digits in the inclusive range 100000–999999 (no leading zero).
2. `P` must contain fewer than 2 alternating repetitive digit pairs (i.e., any pattern `x y x`, where `x` is the same digit and `y` is any digit).
 - Pairs must be counted with overlaps (e.g., `12121` has two pairs for digit `1` at positions 0 and 2, and 2 and 4).

Boilerplate (do not change structure):

```
regex_integer_in_range = r"_____"      # Do not delete 'r'.
regex_alternating_repetitive_digit_pair = r"_____"      # Do not
delete 'r'.

import re
P = input().strip()

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

Requirements:

- Provide only two regex strings that drop into the placeholders above.
- Use anchors to enforce full string match for the range regex.
- Use a lookahead for counting overlapping alternating pairs.
- Must pass tricky cases such as:

- 110000 → False (two alternating pairs for 0)
- 121426 → False (two pairs for 1 and 2)
- 523563 → True (no alternating pair)
- 552523 → False (two pairs)
- 000000 → False (out of range)
- 100000 → True (edge of range)
- No flags, no extra code, no print statements beyond the boilerplate.

Deliverables:

- The two final regex assignments only, filled in, ready to paste.
- A 2–3 sentence explanation of each regex (why it's correct and robust).

Quality bar: clear, minimal, standards-compliant, handles overlaps correctly, and passes all hidden tests.

Output :

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 🔒	Input (stdin)	Download
✓ Test case 3 🔒	1 110000	
✓ Test case 4 🔒	Expected Output	Download
✓ Test case 5 🔒	1 False	
✓ Test case 6 🔒		