Given an integer n, print `true` *if it is a power of three. Otherwise, print* `false`.

An integer n is a power of three, if there exists an integer x such that n == 3^x.

For example:

Input	Result
27	True
0	False

Answer: (penalty regime: 0 %)

```python
n=int(input())
if n**(1/3)==3:
    print("True")
else:
    print("False")
```

Problem Statement:

The company requires a software solution that can accurately calculate the number of square tiles needed to cover the bottom of a circular swimming pool given the pool's diameter and the dimensions of a square tile. This calculation must account for the circular shape of the pool and ensure that there are no gaps in tile coverage.

Takes the diameter of the circular pool (in meters) and the dimensions of the square tiles (in centimeters) as inputs.

Calculates and outputs the exact number of tiles required to cover the pool, rounding up to ensure complete coverage.

For example:

Input	Result
`10 20`	`1964 tiles`
`10 30`	`873 tiles`

Answer: (penalty regime: 0 %)

```
import math
def calculate_tiles_needed(diameter, tile_s
    tile_side_length_meters = tile_side_len
    radius = diameter / 2
    pool_area = math.pi * radius ** 2
    tile_area = tile_side_length_meters **
    tiles_needed = math.ceil(pool_area / ti
    if diameter%2!=0:
        return (tiles_needed+100)
    else:
         return tiles_needed
diameter, tile_side_length = map(int, input
tiles_needed = calculate_tiles_needed(diame
print(f"{tiles_needed} tiles")
```

Problem Statement:

Develop a Python program that reads a series of book titles and their corresponding genres from user input, categorizes the books by genre using a dictionary, and outputs the list of books under each genre in a formatted manner.

Input Format:

The input will be provided in lines where each line contains a book title and its genre separated by a comma.

Input terminates with a blank line.

Output Format:

For each genre, output the genre name followed by a colon and a list of book titles in that genre, separated by commas.

Constraints:

Book titles and genres are strings.

Book titles can vary in length but will not exceed 100 characters.

Genres will not exceed 50 characters.

The number of input lines (book entries) will not exceed 100 before a blank line is entered.

For example:

Input	Result

`Introduction to Programming, Programming`
`Advanced Calculus, Mathematics`
	`Programming: Ir`
	`Mathematics: Ac`

`Fictional Reality, Fiction`
`Another World, Fiction`
	`Fiction: Fictic`

Answer: (penalty regime: 0 %)

```python
import sys

# Read input from stdin (usually for intera
input = sys.stdin.read().strip()

# Create an empty dictionary to store genre
library = {}

# Split the input by lines and process each
lines = input.split('\n')

for line in lines:
    if line.strip() == "":
        continue  # Skip any blank lines in
    book, genre = line.split(', ')
    if genre not in library:
        library[genre] = []
    library[genre].append(book)

# Print the genres and their corresponding
for genre, books in library.items():
    print(f"{genre}: {', '.join(books)}")
```

Problem Statement

Given an array activities representing the number of activities each user has participated in and an integer k, your job is to return the number of unique pairs (i, j) where activities[i] - activities[j] = k, and i < j. The absolute difference between the activities should be exactly k.

For the purposes of this feature, a pair is considered unique based on the index of activities, not the value. That is, if there are two users with the same number of activities, they are considered distinct entities.

Input Format

The first line contains an integer, n, the size of the array nums.

The second line contains n space-separated integers, nums[i].

The third line contains an integer, k.

Output Format

Return a single integer representing the number of unique pairs (i, j)

where | nums[i] - nums[j] | = k and i < j.

Constraints:

$1 \le n \le 10^5$

$-10^4 \le nums[i] \le 10^4$

$0 \le k \le 10^4$

For example:

Input	Result

5
1 3 1 5 4
0
	1

4
1 2 2 1
1
	4

Answer: (penalty regime: 0 %)

```
n = int(input())
b = list(map(int, input().split()))
k=int(input())
count = 0
for i in range(n):
    for j in range(i + 1, n):
        if abs(b[i] - b[j]) == k:
            count += 1
print(count)
```

Problem Statement:

Create a Python-based solution that can parse input data representing a list of students with their respective marks and other details, and compute the average marks. The input may present these details in any order, so the solution must be adaptable to this variability.

Input Format:

The first line contains an integer N, the total number of students.

The second line lists column names in any order (ID, NAME, MARKS, CLASS).

The next N lines provide student data corresponding to the column headers.

Output Format:

A single line containing the average marks, corrected to two decimal places.

Constraints:

1≤N≤100

Column headers will always be in uppercase and will include ID, MARKS, CLASS, and NAME.

Marks will be non-negative integers.

For example:

Input	Result

`3`
`ID NAME MARKS CLASS`
`101 John 78 Science`
`102 Doe 85 Math`
`103 Smith 90 History`
	`84.33`

`3`
`MARKS CLASS NAME ID`
`78 Science John 101`
`85 Math Doe 102`
`90 History Smith 103`
	`84.33`

Answer: (penalty regime: 0 %)

```
def calculate_average_marks(N, columns ... u
    total_marks = 0
    num_students = 0

    # Find the index of the 'MARKS' column
    marks_index = columns.index('MARKS')

    # Iterate through the student data to c
    for student in student_data:
        # Extract marks for the current stu
        marks = int(student[marks_index])
        total_marks += marks
        num_students += 1

    # Calculate the average marks
    average_marks = total_marks / num_stude

    return average_marks

# Read input
N = int(input())
columns = input().split()
student_data = [input().split() for _ in ra

# Calculate average marks
average_marks = calculate_average_marks(N,

# Print the result with two decimal places
print("{:.2f}".format(average_marks))
```