**Codewriting**

**Problem 1:**

You are given a matrix of **positive** floating point numbers **dataset** Some

part of the data is missing, and instead of the actual value, it's set to o

Your task is to perform 2 steps:

1. Replace each o with the mean of the non-zero values in its column (it's guaranteed that there are at least two different non-zero values in each column)

2. Normalize the matrix in such way that

* The mean value of each column is **0**
* The standard deviation of the values of each column is **1**

In other words, change each element **e** in each column **C** to e -

You are allowed to use libraries like ***sklearn, numpy*** and ***pandas*** in python.

**Example**

For

dataset - [[1, 2, 0],

[0, 1, 1],

[5, 6, 5]]

the output should be

**solution(dataset) = [[-1.224744871391589, -0.4629100498862757, 0]**

**[0, -0.9258200997725514, -1.22374487]**

**[1.224744871391589, 1.38873015, 1.22474487]]**

After replacing the 0 values, **dataset** will look like this:

**[[1, 2, 3.0],**

**[3. 0, 1, 1],**

**[5, 6, 5]]**

As the mean of the non-zero values in the first column is (1+5)/ 2 = 3.0, which

is the same as in the third column.

The mean values for columns are [3.0, 3.0, 3.0] as (1+3.0+5) /3 = 3.0,

(2+1+6)/ 3 = 3.0 and (3.0-1+5)/ 3 = 3.0.

The standard deviations for columns are. **[1. 6329931618554518, 2. 6457513110645907, 1.6329931618554518]**

As for example for the first column

The final dataset is

**[[(1 - 3.0) / 1.6323, (2 - 3.0) / 2.64575, (3.0 - 3.0) / 1.632**

**[(3.0 - 3.0) / 1.6323, (1 - 3.0) / 2.64575, (1 - 3.0) / 1.6323**

**[(5 - 3.0) / 1.6323, (6 - 3.0) / 2.64575, (5 - 3.0) / 1.6323]**

**Note** that the answer will be considered correct if for each element of the output, the absolute difference between your value and the correct one doesn't exceed .

**Input/Output**

* **[execution time limit] 4 seconds (py3)**
* **[input] array.array.float dataset**

The dataset represented as a matrix. It's guaranteed that each column has at least two different non-zero elements.

Guaranteed constraints:

**dataset [i]. length = dataset[0]. Length**

* **[output] array.array.float**

The normalized version of the dataset. The answer will be considered correct if for each element of the output, the absolute difference between your value and the correct one doesn’t exceed .

**Note: Write a test case to validate your code**

**Problem 2:**

We want to query an API endpoint to receive data about currently available apartment listings from a rental website. Among the data fields is a column called **num\_bedrooms** which takes the value of **1** for a "**1-bedroom**" apartment and **0** for a **"studio"**

**Note**: This rental agency only works with studios and 1-bedroom apartments, so there will never be 2+ bedroom listings. Each listing includes information about a **"studio"** or a **"1-bedroom"** apartment, so there will never be a listing with both a **"studio"** and **"1-bedroom"** offerings in one posting.

The algorithm used occasionally mistags the num\_bedrooms value. Specifically, sometimes a **"studio"** is tagged as having **num\_bedrooms = 1** or a **"1-bedroom"** is tagged as **num\_bedrooms = 0**

Further investigation revealed it to be an issue with one of the data fields, **description** and the way our algorithm parsed the field to extract a **num\_bedrooms** value.

For example: **"description": "Beautiful 1-bedroom apartment with nearby yoga studio**." was detected as a **yoga studio** instead of **1- bedroom** and incorrectly had **num\_bedrooms = 0**

Your task is to write a function that takes in the **jsonData** and corrects this problem. The **GET** request retrieves the data as a **string** which looks like this:

**jsonData - [**

**{**

**"id": "3",**

**"agent": "Ton Jett",**

**"unit": "#12",**

**"description": "Beautiful 1-bedroom apartment with nearby**

**"num\_bedrooms": 1**

**},**

**]**

While correcting the problem, remember the following edge cases:

* If the word "**studio**" or "**1-bedroom**" is preceded immediately by any of the words: "**yoga**" "**dance**" or "**art**" don't consider it for **num\_bedrooms** value.
* If the description does not contain the word "**studio**" or **"1-bedroom"** do not change the value for **num\_bedrooms**.
* The rules above should be applied regardless of punctuation or letter casing within the description field.

Your end goal is to return an array of integers representing **num\_bedrooms** for each rental listing, example: [0, 1, 1, 1, 0,0 ]

**Example**

For

**jsonData - "[{"id": "1", "agent": "Radulf Katlego", "unit": "#"}]"**

the output should be **solution(jsonData) - [0, 1, 1, 0]**

The above **jsonData** represents the following JSON:

[

{

"id": "1",

"agent": "Radulf Katlego",

"unit": "#3",

"description": "This luxurious studio apartment is in the heart of downtown.",

"num\_bedrooms": 1

},

{

"id": "2",

"agent": "Kelemen Konrad",

"unit": "#36",

"description": "We have a 1-bedroom available on the third floor.",

"num\_bedrooms": 1

},

{

"id": "3",

"agent": "Ton Jett",

"unit": "#12",

"description": "Beautiful 1-bedroom apartment with nearby yoga studio.",

"num\_bedrooms": 1

},

{

"id": "4",

"agent": "Fishel Salman",

"unit": "#13",

"description": "Beautiful studio with a nearby art studio.",

"num\_bedrooms": 1

}

]

Explanation:

In the first listing, **description** = **"This luxurious studio apartment is in the heart of downtown. "**

* "**studio**" should have **num\_bedrooms - 0**;

In the second listing, description = **"We have a 1-bedroom available on the third floor."**

* "**1-bedroom**" should have **num\_bedrooms - 1**;

In the third listing, description = **"Beautiful 1-bedroom apartment with nearby yoga studio."**

* **"1-bedroom"** should have **num\_bedrooms =1**. Ignore "**studio**" since it is followed by "**yoga**"

In the fourth listing, **description** = **"Beautiful studio with a nearby art studio**.”

* "**studio**" should have **num\_bedrooms – 0**. Ignore the second appearance of "**studio**" since it is followed by "**art**"

**Input/Output**

* **[execution time limit] 4 seconds (py3)**
* **[input] string jsonData**

String in JSON format. It's guaranteed that each listing contains the fields

**"id" , "agent" , "unit" , "description"** and **"num\_bedrooms".**

*Guaranteed constraints:*

**[output] array.integer**

Return an array that contains the correct values for **num\_bedrooms**

**Note: Write a test case to validate your code**