# Lab: Lists Advanced

Problems for in-class lab for the [Python Fundamentals Course @SoftUni](https://softuni.bg/trainings/2442/python-fundamentals-september-2019). Submit your solutions in the SoftUni judge system at <https://judge.softuni.bg/Contests/1730>

## Trains

You will receive how many **wagons** the train has. Create a **list** with that length **with all zeros**. Until you receive the command **"End"**, you get some of the following commands:

* **add {people}** -> adds the people in the last wagon
* **insert {index} {people}** -> adds the people at the given wagon
* **leave {index} {people}** -> removes the people from the wagon

After you receive the **"End"** command print the train

### Example

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3  add 20  insert 0 15  leave 0 5  End | [10, 0, 20] |
| 5  add 10  add 20  insert 0 16  insert 1 44  leave 1 12  insert 2 100  insert 4 61  leave 4 1  add 15  End | [16, 32, 100, 0, 105] |

## Todo List

You will receive a **todo-notes** until you get the command **"End"**. The notes will be in the format: **"{importance}-{value}"**. Return the list of **todo-notes** sorted by **importance.** The maximum importance will be **10**

### Hint

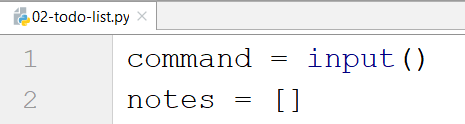
* Use the **insert()** method

### Example

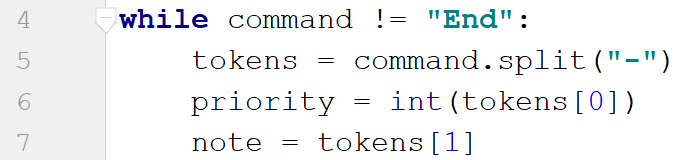
|  |  |
| --- | --- |
| **Input** | **Output** |
| 2-Walk the dog  1-Drink coffee  6-Dinner  5-Work  End | ['Drink coffee', 'Walk the dog', 'Work', Dinner'] |

### Hints

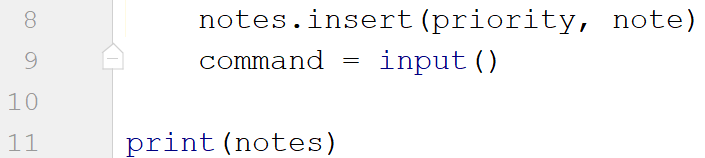
We start by reading the first command and creating an empty list



We create a while loop and split the command



We insert the note into the list, read the new command and finally print the result



## Palindrome Strings

Write a program that receives on the **first line** words separated by a **space** and a **searched palindrome** on the **second**. Print **all the palindromes** on the first line. Print all the **occurrences** of the **searched** palindrome in the format: **"Found palindrome {count} times"**

### Example

|  |  |
| --- | --- |
| **Input** | **Output** |
| wow father mom wow shirt stats  wow | ['wow', 'mom', 'wow', 'stats']  Found palindrome 2 times |
| hey how you doin? lol  mom | ['lol']  Found palindrome 0 times |

### Hints

First, read all the **strings** and the **searched** palindrome, and create an **empty list** for the found **palindromes**



Create a loop and check if each word is a palindrome



* We use the **join()** method with the **reversed()** method, because **reversed()** returns an **iterator**, not a **string**, so we **make it into one**

Finally, we print the result



## Even Numbers

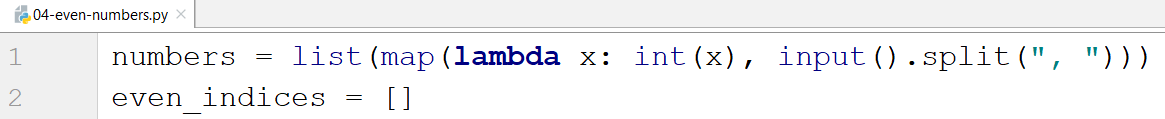
Write a program that reads **a single string** with **numbers** separated by comma and space **", "**. Print the **indices** of all **even numbers**

### Example

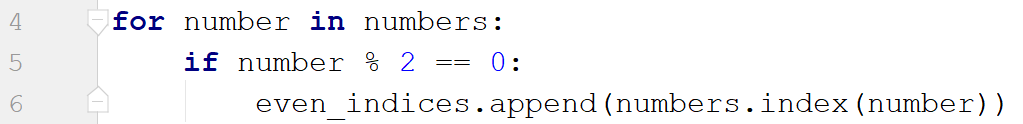
|  |  |
| --- | --- |
| **Input** | **Output** |
| 3, 2, 1, 5, 8 | [1, 4] |
| 2, 4, 6, 9, 10 | [0, 1, 2, 4] |

### Hints

Read the string, split it and convert the list of strings into a list of numbers using lambda



Create a loop to check all the numbers if they are even, and add the indices for those who are



Print the result



## The Office

### You will receive two lines of input: a list of employee's happiness as string with numbers separated by a single space and a happiness improvement factor (single number). Your task is to find out if the employees are generally happy in their office. To do that you have to increase their happiness by multiplying the all the employee's happiness (the numbers from the list) by the factor, filter the employees which happiness is greater than or equal to the average in the new list and print the result

There are **two types of output**:

* If the **half or more** of the employees have **happiness >= than the average**: **"Score: {happy\_count}/{unhappy\_count}. Employees are happy!"**
* Otherwise: **"Score: {happy\_count}/{unhappy\_count}. Employees are not happy!"**

### Example

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comment** |
| 1 2 3 4 2 1  3 | Score 2/6. Employees are not happy! | After the mapping:  3 6 9 12 6 3  After the filtration:  9 12  2/6 people are happy, so the overall happiness is bad |
| 2 3 2 1 3 3  4 | Score: 3/6. Employees are happy! | Half of the people are happy, so the overall happiness is good |

### Hints

First, we **read** the input



Then we use the **map()** function to **multiply** each element with **the factor**



* Since all of the elements in the employees list are **strings**, we **parse** them to **ints** before we **multiply** them
* Don't forget that the map function returns a **map object**, so we need to **cast it to a list**

Now, it is time to filter the elements that are greater than the average



* We find the average by **summing** the elements and **divide** the result to its **length**

Finally we print the result

