# Lab: Lists Advanced

Problems for in-class lab for the [Python Fundamentals Course @SoftUni](https://softuni.bg/trainings/3368/python-fundamentals-may-2021).

Submit your solutions in the SoftUni judge system at <https://judge.softuni.bg/Contests/1730>.

## No Vowels

Using a comprehension write a program that receives a **text** and **removes** all the **vowels** from it, **case insensitive**. Print the new text **string after removing the vowels**. The vowels that should be considered are **'**a**', '**o**', '**u**', '**e**', '**i**'**.

text = input()  
  
vowels = ['a', 'o', 'u', 'e', 'i', 'A', 'O', 'U', 'E', 'I']  
  
for i in text:  
 if i not in vowels:  
 text = ''.join(i)  
  
 print(text, end='')

A possible solution with comprehension lists would be:

text = input()  
vowels = ['a', 'u', 'e', 'i', 'o', 'A', 'U', 'E', 'I', 'O']  
no\_vowels = ''.join([x for x in text if x not in vowels])  
print(no\_vowels)

RS:

text\_input = input()  
  
no\_vowels\_list = []  
no\_vowels\_str = ''  
vowels = ['a', 'o', 'u', 'e', 'i', 'A', 'O', 'U', 'E', 'I']  
  
for i in text\_input:  
 if i not in vowels:  
 no\_vowels\_list.append(i)  
  
no\_vowels\_str = ''.join([x for x in no\_vowels\_list])  
print(no\_vowels\_str)

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Python | Pythn |
| ILovePython | LvPythn |

## Trains

You will receive a number representing the number of **wagons** a train has. Create a **list** (train)with the given length containing **only zeros**. Until you receive the command **"End"**, you will receive some of the following commands:

* **"add {people}"** – you should add the number of people in the last wagon
* **"insert {index} {people}"** - you should add the number of people at the given wagon
* **"leave {index} {people}"** - you should remove the number of people from the wagon

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

number\_wagons = int(input())  
train\_list = [0] \* number\_wagons  
  
command = input()  
  
while command != 'End':  
 data = command.split()  
 if data[0] == 'add':  
 n\_people = int(data[1])  
 train\_list[-1] += n\_people  
 elif data[0] == 'insert':  
 n\_people = int(data[2])  
 wagon\_index = int(data[1])  
 train\_list[wagon\_index] += n\_people  
 elif data[0] == "leave":  
 n\_people = int(data[2])  
 wagon\_index = int(data[1])  
 train\_list[wagon\_index] -= n\_people  
  
 command = input()  
  
print(train\_list)

### 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] |

## To-do List

You will be receiving **to-do notes** until you get the command **"End"**. The notes will be in the format: **"{importance}-{note}"**. Return the list of **to-do notes** sorted by **importance.** The importance value will always be an integer between **1** and **10 (inclusive)**.

### Hint

* Use the **pop()** and **insert()** methods.

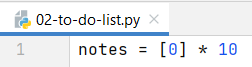
command = input()  
  
notes = [0] \* 10  
  
while not command == 'End':  
 importance, text = command.split('-')  
 current\_index = int(importance) - 1  
 notes[current\_index] = text  
 command = input()  
  
print([x for x in notes if x != 0])

### Example

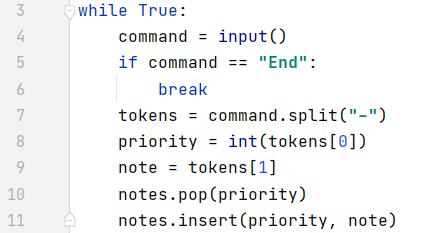
|  |  |
| --- | --- |
| **Input** | **Output** |
| 2-Walk the dog  1-Drink coffee  6-Dinner  5-Work  End | ['Drink coffee', 'Walk the dog', 'Work', 'Dinner'] |

### Hints

Start by and creating an empty list:



Create a while loop which reads the command and splits it if it is not "End". Then, remove the zero from the list by index (priority) and insert the note on its place:



Finally, filter only the nonzero elements from the notes list and print the result:



## Palindrome Strings

On the **first line** you willreceive words separated by a **single** **space**. On the **second line** you will receive a **palindrome**. First, you should print a list containing **all the found palindromes in the sequence**. Then, you should print the number of **occurrences** of the givenpalindrome in the format: **"Found palindrome {number} times"**.

input\_list = input().split()  
palindrome = input()  
  
palindromes\_list = [x for x in input\_list if x == x[::-1]]  
  
print(palindromes\_list)  
  
if palindrome in palindromes\_list:  
 n\_occurrence = palindromes\_list.count(palindrome)  
 print(f"Found palindrome {n\_occurrence} times")  
else:  
 print("Found palindrome 0 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 we create an **empty list** for the found **palindromes**:



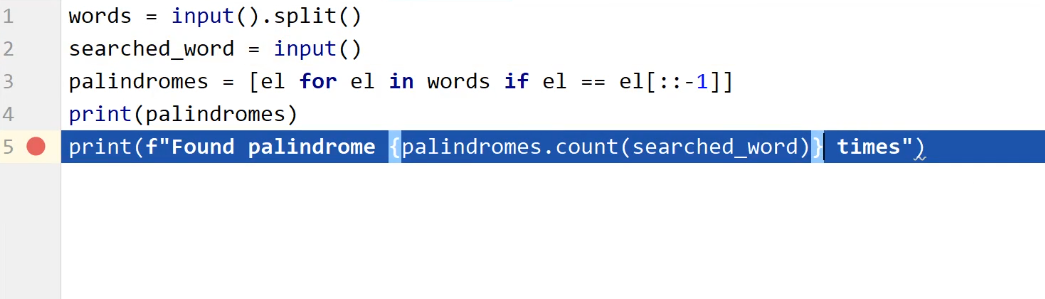
Then, we create a loop which checks if each word is a palindrome:



* We use the **join()** method with the **reversed()** method, because **reversed()** returns an **iterator**, not a **string**, so we should **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**.

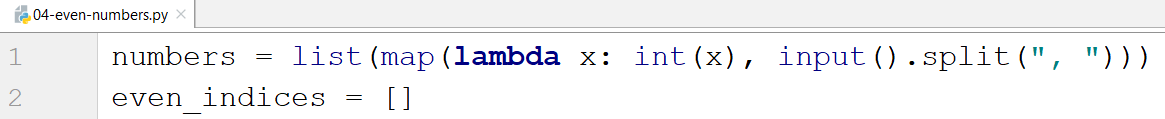
input\_list = input().split(', ')  
input\_list\_int = list(map(int, input\_list))  
  
output\_list = []  
  
for i in range(len(input\_list\_int)):  
 if input\_list\_int[i] % 2 == 0:  
 output\_list.append(i)  
  
print(output\_list)

### Example

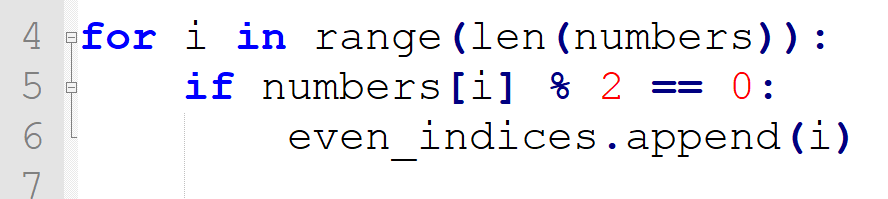
|  |  |
| --- | --- |
| **Input** | **Output** |
| 3, 2, 1, 5, 8 | [1, 4] |
| 2, 4, 6, 9, 10 | [0, 1, 2, 4] |

### Hints

Read the string, split it, and convert the list of strings into a list of numbers using map function:



Create a loop to find all the even numbers, and add their indices to a list:



Print the result:



## The Office

You will receive two lines of input:

* a list of **employees' happiness** as string of numbers separated by a single space
* a happiness improvement **factor** (single number).

Your task is to find out if the employees are generally happy in their office. You should **increase** their happiness by **multiplying** each of the employees' happiness by the factor. Then, **print** **one** of the following lines:

* If **half or more** of the employees have happiness **greater than** **or equal to the average**:

**"Score: {happy\_count}/{total\_count}. Employees are happy!"**

* Otherwise:

**"Score: {happy\_count}/{total\_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, **read** the input:



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



* Since all the elements in the employees list are **strings**, we **parse** them to **integers** before we **multiply** them.
* Do not forget that the map function returns a **map object**, so we need to **cast it to a list**.

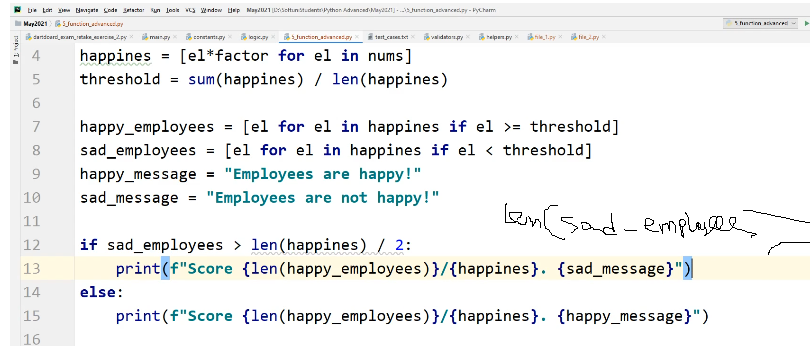
Now, it is time to filter the elements that are greater than the average:



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

Finally, we print the result:





employees = input().split()  
employees\_ints = list(map(int, employees))  
factor = int(input())  
employees\_happiness = []  
  
for emp in employees\_ints:  
 multiplied\_int = emp \* factor  
 employees\_happiness.append(multiplied\_int)  
  
sum\_happiness = sum(employees\_happiness)  
employees\_count = len(employees\_ints)  
average\_happiness = sum\_happiness / employees\_count  
happy\_employees = [x for x in employees\_happiness if x >= average\_happiness]  
happy\_employees\_count = len(happy\_employees)  
  
if happy\_employees\_count >= employees\_count / 2:  
 print(f"Score: {happy\_employees\_count}/{employees\_count}. Employees are happy!")  
  
else:  
 print(f"Score: {happy\_employees\_count}/{employees\_count}. Employees are not happy!")