# Exercise: Lists

Problems for exercises and homework for the [“Python Fundamentals” course @ SoftUni](https://softuni.bg/opencourses/python-fundamentals-course).

You can check your solutions here: <https://judge.softuni.bg/Contests/924/>.

## Sum List Items

Write a program, which reads a **list** of integers, calculates its **sum** and **prints** it.

The input consists of a **number** n (the number of items) + n integers, each as a separate line.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| **4**  1  2  3  4 | 10 |
| **5**  1  1  1  1  1 | 5 |
| **4**  2  -1  -2  8 | 7 |

### Hints

* First, read the number n.
* Read the integers in a for-loop.

## Multiply a List of Integers

Write a program to read **a list of integers**, an integer p, multiply each item by p and print the resulting list.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 1 3 12 4  4 | 4 12 48 16 |
| 6 8 1 -9  3 | 18 24 3 -27 |

### Hints

* Read the list
* **Loop through** the list, **multiplying each item** by **p**
* Finally, **print** the resulting list, using a **for** loop

## Smallest Item in List

Write a program to read **a list of integers**, find the **smallest item** and **print** it.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| **1** 2 3 4 | 1 |
| 3 2 9 **-9** 6 1 | -9 |
| -6 0 **-17** -1 | -17 |

### Hints

* **Loop through** the **integer list** until you find the **smallest item**

## Rotate List of Strings

Write a program to read **a list of strings**, **rotate** it to the right and **print** its rotated items.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| a b c d e | e a b c d |
| soft uni hi | hi soft uni |
| i r a b | b i r a |

### Hints

* You can store the rotated list in a **second list** alongside the first one

## Count of Odd Numbers in List

Write a program to read **a list of integers** and find **how many odd items it holds**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| **1** -2 **3** 4 | 2 |
| **3** **9** **-9** -6 **1** -2 | 4 |
| 66 0 2 1 | 1 |

### Hints:

* You can check if a number is **odd** if you **divide it by 2** and check whether you get **a remainder of 1**.
* Odd numbers, which are negative, have a **remainder** of **-1**.

## Odd Numbers at Odd Positions

Write a program to read a list of integers and find how many **odd numbers** at **odd positions** the list holds. If there are no numbers, which match this criterion, **do not print anything**

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Explanation** |
| 2 **3** 5 2 7 **9** -1 **-7** | Index 1 -> 3  Index 5 -> 9  Index 7 -> -7 | Indexes: 0 **1** 2 3 4 **5** 6 **7**  Numbers: 2 **3** 5 2 7 **9** **-1** -7  Odd positions with odd numbers: 1, 5 and 7 |
| 2 **3** 55 2 4 **1** | Index 1 -> 3  Index 5 -> 1 | Indexes: 0 **1** 2 3 4 **5**  Numbers: 2 **3** 55 2 4 **1**  Odd positions with odd numbers: 1 and 5 |
| 5 0 1 2 | *(no output)* | Indexes: 0 1 2 3  Numbers: 5 0 1 2  Odd positions with odd numbers: **none** |

### Hints

* Positions are counted **from** **0** from left to right, so if for example the second item (**index 1**) is **odd**, then we **should** count it, and so on…
* Do **NOT** count odd numbers, which are at **even** positions (0, 2, 4, etc…)

## Remove Negatives and Reverse

Read a **list of integers**, **remove all negative numbers** from it and print the remaining items in **reversed order**. In case of no items left in the list, print “empty”.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 10 -5 7 9 -33 50 | 50 9 7 10 |
| 7 -2 -10 1 | 1 7 |
| -1 -2 -3 | empty |

### Hints

* Read the list
* Create a new empty list for the results.
* Scan the input list from the end to the beginning. Check each item and append all non-negative items to the result list.
* Finally, print the results list (at a single line holding space-separated numbers).

## Append Lists

Write a program to **append several lists** of numbers.

* Lists are separated by ‘|’.
* Values are separated by spaces (‘ ’, one or several)
* Order the lists from the **last** to the **first**, and their values from **left** to **right**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 1 2 3 |4 5 6 | 7 8 | 7 8 4 5 6 1 2 3 |
| 7 | 4 5|1 0| 2 5 |3 | 3 2 5 1 0 4 5 7 |
| 1| 4 5 6 7 | 8 9 | 8 9 4 5 6 7 1 |
| 1 | | |||2 3 |4 5 6 | 7 8| -3 2 1||1 2|3|4|99 77 | 99 77 4 3 1 2 -3 2 1 7 8 4 5 6 2 3 1 |

### Hints

* Create a new empty list for the results.
* Split the input by ‘|’ into list of tokens.
* Pass through each of the obtained tokens from tight to left.
  + For each token, split it by space and append all non-empty tokens to the results.
* Print the results.

## Sort Numbers

Read a **list of integers** and **sort** them in **ascending order**. Print the output as shown in the examples below.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 8 2 7 3 | 2 <= 3 <= 7 <= 8 |
| 2 4 -9 | -9 <= 2 <= 4 |

### Hints

* Use the built-in method list.sort().

## Square Numbers

Read a **list of integers** and **extract all square numbers** from it and print them in **descending order**. A **square number** is an integer which is the square of any integer. For example, 1, 4, 9, 16 are square numbers.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3 **16 4** 5 6 8 **9** | 16 9 4 |
| 12 **1 9 4 16** 8 **25 49 16** | 49 25 16 16 9 4 1 |
| -1 2 -5 100 121 81 49 36 25 24 169 -169 |  |

### Hints

* To find out whether an integer is “**square number**”, check whether its square root is integer number (has no fractional part):
  + if (√num == (int)√num) …
* To order the results list in descending order use sorting with reverse