# Exercise: Functions

Problems for exercise and homework 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/1728>

## Smallest of Three Numbers

Write a function which receives **three integer** numbers and returns the **smallest**. Use appropriate name for the function.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2  5  3 | 2 |
| 600  342  123 | 123 |
| 25  21  4 | 4 |

## Add and Subtract

You will receive **three** **integer numbers.**

Write a function sum\_numbers() to get the sum of the first **two** integers and subtract()function that subtracts the **third** integer from the result. Wrap the two functions in a function called **add\_and\_subtract()** which will receive the three numbers

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 23  6  10 | 19 |
| 1  17  30 | -12 |
| 42  58  100 | 0 |

## Characters in Range

Write a function that receives **two characters** and returns a single string with all the characters in between them according to the **ASCII** code.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| a  d | b c |
| #  : | $ % & ' ( ) \* + , - . / 0 1 2 3 4 5 6 7 8 9 |
| C  # | $ % & ' ( ) \* + , - . / 0 1 2 3 4 5 6 7 8 9 : ; < = > ? @ A B |

## Odd and Even Sum

You will receive a **single number.** You have to write a function that returns the **sum** of **all even** and **all odd** digits from that number as a single string like in the examples below.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 1000435 | Odd sum = 9, Even sum = 4 |
| 3495892137259234 | Odd sum = 54, Even sum = 22 |

## Palindrome Integers

A palindrome is a number which reads the same **backward as forward**, such as 323 or 1001. Write a function which receives a **list of positive integers** and checks if each integer is a palindrome or not. Print the results on the console

The input will be a single string containing the numbers separated by comma and space **", "**

### Examples

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| 123, 323, 421, 121 | False  True  False  True |  | 32, 2, 232, 1010 | False  True  True  False |

### Hints

* Read more about palindromes: <https://en.wikipedia.org/wiki/Palindrome>

## Password Validator

Write a function that checks if a given password is valid. Password validations are:

* The **length** should be **6 - 10** characters (inclusive)
* It should consists **only** **letters** and **digits**
* It should have **at least 2** digits

If a password is valid print **"Password** **is** **valid"**.

If it is **NOT** valid, for every unfulfilled rule print a message:

* **"Password must be between 6 and 10 characters"**
* **"Password must consist only of letters and digits"**
* **"Password must have at least 2 digits"**

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| logIn | Password must be between 6 and 10 characters  Password must have at least 2 digits |
| MyPass123 | Password is valid |
| Pa$s$s | Password must consist only of letters and digits  Password must have at least 2 digits |

## Perfect Number

Write a function that receives an integer **number** and returns if this number is **perfect** or **NOT**.

A perfect number is a **positive** integer that is equal to the **sum** ofits **proper positive divisors**. That is the sum of its positive **divisors** excluding the number itself (also known as its **aliquot sum**).

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 6 | We have a perfect number! | 1 + 2 + 3 |
| 28 | We have a perfect number! | 1 + 2 + 4 + 7 + 14 |
| 1236498 | It's not so perfect. |  |

### Hint

Equivalently, a perfect number is a number that is **half the sum** of all of its positive divisors (including itself) => 6 is a perfect number, because it is the sum of 1 + 2 + 3 (all of which are divided without residue).

* Read about the Perfect number here: <https://en.wikipedia.org/wiki/Perfect_number>

## \* Loading Bar

You will receive a **single integer number** between **0** and **100** which is divided with 10 without residue (0, 10, 20, 30...).

Your task is to create a function that visualizes a **loading bar** depending on that number you have received in the input.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 30 | 30% [%%%.......]  Still loading... |
| 50 | 50% [%%%%%.....]  Still loading... |
| 100 | 100% Complete!  [%%%%%%%%%%] |

## \* Factorial Division

Write a function that receives **two** integer numbers. Calculate **factorial** of each number. Divide the first result by the second and print the division formatted to the **second decimal** point.

### Examples

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| 5  2 | 60.00 |  | 6  2 | 360.00 |

### Hints

* Read more about factorial here: <https://en.wikipedia.org/wiki/Factorial>

## \*Array Manipulator

Trifon has finally become a junior developer and has received his first task. It's about manipulating an array of integers. He is not quite happy about it, since he hates manipulating arrays. They are going to pay him a lot of money, though, and he is willing to give somebody half of it if to help him do his job. You, on the other hand, love arrays (and money) so you decide to try your luck.

The array may be manipulated by one of the following commands

* **exchange {index}** – splits the array **after** the given index, and exchanges the places of the two resulting sub-arrays. E.g. [1, 2, 3, 4, 5] -> exchange 2 -> result: [4, 5, 1, 2, 3]
  + If the index is outside the boundaries of the array, print "**Invalid index**"
* **max** **even/odd**– returns the **INDEX** of the max even/odd element -> [1, 4, 8, 2, 3] -> **max odd** -> print **4**
* **min** **even/odd** – returns the **INDEX** of the min even/odd element -> [1, 4, 8, 2, 3] -> **min even** > print **3**
  + If there are two or more equal **min/max** elements, return the index of the **rightmost** one
  + If a **min/max even/odd** element **cannot** be found, print **"No matches"**
* **first {count}** **even/odd**– returns the first {count} elements -> [1, 8, 2, 3] -> **first 2 even** -> print [**8, 2]**
* **last {count}** **even/odd** – returns the last {count} elements -> [1, 8, 2, 3] -> **last 2 odd** -> print [**1, 3]**
  + If the count is greater than the array length, print "**Invalid count**"
  + If there are **not** **enough** elements to satisfy the count, print as many as you can. If there are **zero** **even/odd** elements, print an empty array "[]"
* **end** – stop taking input and print the final state of the array

### Input

* The input data should be read from the console.
* On the first line, the initial array is received as a line of integers, separated by a single space
* On the next lines, until the command "**end**" is received, you will receive the array manipulation commands
* The input data will always be valid and in the format described. There is no need to check it explicitly.

### Output

* The output should be printed on the console.
* On a separate line, print the output of the corresponding command
* On the last line, print the final array in **square brackets** with its elements separated by a comma and a space
* See the examples below to get a better understanding of your task

### Constraints

* The **number of input lines** will be in the range [2 … 50].
* The **array elements** will be integers in the range [0 … 1000].
* The **number of elements** will be in the range [1 .. 50]
* The **split index** will be an integer in the range [-231 … 231 – 1]
* **first/last count** will be an integer in the range [1… 231 – 1]
* There will **not** be redundant whitespace anywhere in the input
* Allowed working time for your program: 0.1 seconds. Allowed memory: 16 MB.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 1 3 5 7 9  exchange 1  max odd  min even  first 2 odd  last 2 even  exchange 3  end | 2  No matches  [5, 7]  []  [3, 5, 7, 9, 1] |
| **Input** | **Output** |
| 1 10 100 1000  max even  first 5 even  exchange 10  min odd  exchange 0  max even  min even  end | 3  Invalid count  Invalid index  0  2  0  [10, 100, 1000, 1] |
| **Input** | **Output** |
| 1 10 100 1000  exchange 3  first 2 odd  last 4 odd  end | [1]  [1]  [1, 10, 100, 1000] |