# Lab: Functions Advanced

This document defines the exercises for the ["Python Advanced" course at @Software University.](https://softuni.bg/modules/74/python-advanced)

Please, submit your source code solutions for the described problems to the [Judge System](https://judge.softuni.org/Contests/1838/Functions-Advanced-Lab).

## Multiplication Function

Write a function **called multiply** that can receive any quantity of numbers (integers) as different parameters and **returns the result** of the multiplication of all of them. Submit **only your function** in the Judge system.

### Examples

|  |  |
| --- | --- |
| **Test Code** | **Output** |
| print(multiply(1, 4, 5))  print(multiply(4, 5, 6, 1, 3))  print(multiply(2, 0, 1000, 5000)) | 20  360  0 |

## Person Info

Write a function **called** get\_info that receives a **name**, an **age,** and a **town** and **returns** a string in the format:   
**"This is {name} from {town} and he is {age} years old"**. Use dictionary unpacking when testing your function. Submit **only the function** in the judge system.

### Examples

|  |  |
| --- | --- |
| **Test Code** | **Output** |
| print(get\_info(\*\*{"name": "George", "town": "Sofia", "age": 20})) | This is George from Sofia and he is 20 years old |

## Cheese Showcase

White a function called sorting\_cheeses that receives **keywords arguments**:

* The **key** represents the **name** of the cheese
* The **value** is a **list** **of quantities** (integers) of the **pieces** of the given cheese

The function should **return** the **cheeses** and their pieces' **quantities** **sorted by the number of pieces** of a cheese kind in **descending** order. If two or more cheeses have the **same number of pieces**, sort them by **their names** in **ascending** order (**alphabetically**). For **each kind of cheese**, return their **piece quantities** in **descending** order.

For more clarifications, see the examples below.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| print(  sorting\_cheeses(  Parmesan=[102, 120, 135],  Camembert=[100, 100, 105, 500, 430],  Mozzarella=[50, 125],  )  ) | Camembert  500  430  105  100  100  Parmesan  135  120  102  Mozzarella  125  50 |
| print(  sorting\_cheeses(  Parmigiano=[165, 215],  Feta=[150, 515],  Brie=[150, 125]  )  ) | Brie  150  125  Feta  515  150  Parmigiano  215  165 |

## Rectangle

Create a function called **rectangle()**. It must have two parameters - **length** and **width**.

First, you need to check if the **given arguments are** **integers**:

* If one/ both of them is/ are **NOT** integer/s, **return** the string **"Enter valid values!"**

Create two inner functions:

* **area()** - returns the **area of the rectangle** with the given length and width
* **perimeter()** - returns the **perimeter of the rectangle** with the given length and width

In the end, the **rectangle** function should **return** a string containing the **area** and the **perimeter** of a rectangle in the following format:

**"Rectangle area: {ract\_area}**

**Rectangle perimeter: {rect\_perim}"**

### Examples

|  |  |
| --- | --- |
| **Test Code** | **Output** |
| print(rectangle(2, 10)) | Rectangle area: 20  Rectangle perimeter: 24 |
| print(rectangle('2', 10)) | "Enter valid values!" |

## Recursive Power

Create a **recursive** function called recursive\_power() which should receive a **number** and a **power**. Using **recursion,** **return** the result of **number \*\* power**. Submit only the function in the judge system.

### Examples

|  |  |
| --- | --- |
| **Test Code** | **Output** |
| print(recursive\_power(2, 10)) | 1024 |
| print(recursive\_power(10, 100)) | 10000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 |

## Operate

Write a function **called** operate that receives an operator **("+", "-", "\*" or "/")** as а first argument and multiple numbers (integers) as additional arguments (\*args). The function should **return the result** of the operator applied to all the numbers. For more clarification, see the examples below.

Submit **only your function** in the Judge system.

***Note: Be careful when you have multiplication and division***

### Examples

|  |  |  |
| --- | --- | --- |
| **Test Code** | **Output** | **Comment** |
| print(operate("+", 1, 2, 3)) | 6 | 1 + 2 + 3 = 6 |
| print(operate("\*", 3, 4)) | 12 | 3 \* 4 = 12 |