# Exercise: Syntax, Functions and Statements

Problems for in-class lab for the ["JavaScript Advanced" course @ SoftUni](https://softuni.bg/courses/js-advanced). Submit your solutions in the SoftUni judge system at <https://judge.softuni.bg/Contests/1796/Exercise-Syntax-Functions-and-Statements>

1. **Fruit**

Write a function that calculates how much money you need to buy fruit. You will receive a **string** for the type of fruit you want to buy, **a number** for weight in grams and another **number** for the price per kilogram.

Print the following text on the console:

**'I need ${money} to buy {weight} kilograms {fruit}.'**

Print the weight and the money **rounded** to two decimal places.

The **input** comes as **three arguments** passed to your function.

The **output** should be printed on the console.

**Example**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 'orange', 2500, 1.80 | I need $4.50 to buy 2.50 kilograms orange. |

|  |  |
| --- | --- |
| **Input** | **Output** |
| 'apple', 1563, 2.35 | I need $3.67 to buy 1.56 kilograms apple. |

1. **Greatest Common Divisor - GCD**

Write a function that takes **two** **positive** **numbers** as input and compute the greatest common divisor.

The **input** comes **as two positive integer numbers**.

The **output** should be printed on the console.

**Example**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| 15, 5 | 5 |  | 2154, 458 | 2 |

1. **Same Numbers**

Write a function that takes **an integer** **number** as an input and check if all the digits in a given number are the same or not.

Print on the console **true** if all numbers are same and **false** if not. On the next line print the **sum of all the digits.**

The **input** comes as an integer number.

The **output** should be printed on the console.

**Examples**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| 2222222 | true  14 |  | 1234 | false  10 |

1. **Time to Walk**

Write a function that **calculates** how long it takes a student to get to university.   
The function takes **three numbers**:

* The **first** is the number of **steps** the student takes from their home to the university
* Тhe **second** number is the length of the student's footprint in **meters**
* Тhe **third** number is the student speed in **km/h**

Every 500 meters the students a rest and takes a **1 minute break**.

Calculate how long the student walks from home to university and print on the console the result in the following format: **'**hours:minutes:seconds**'**.

The **input** comes as **three numbers**.

The **output** should be printed on the console.

**Example**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| 4000, 0.60, 5 | 00:32:48 |  | 2564, 0.70, 5.5 | 00:22:35 |

1. **Calorie Object**

Write a function that composes an object by given properties. The input comes as an **array of strings**. Every **even index** of the array represents the **name of the food**. Every **odd index** is a **number** that is equal to the **calories in 100 grams of the given product**. Assign each value to its corresponding property and print it on the console.

The **input** comes as an **array of string** **elements**.

The **output** should be printed on the console.

**Examples**

|  |  |
| --- | --- |
| **Input** | **Output** |
| ['Yoghurt', 48, 'Rise', 138, 'Apple', 52] | { Yoghurt: 48, Rise: 138, Apple: 52 } |
| ['Potato', 93, 'Skyr', 63, 'Cucumber', 18, 'Milk', 42] | { Potato: 93, Skyr: 63, Cucumber: 18, Milk: 42 } |

1. **Road Radar**

Write a function that determines whether a driver is within the speed limit. You will receive the speed and the area. Each area has a different limit:

* On the **motorway** the limit is **130 km/h**
* On the **interstate** the limit is **90 km/h**
* In the **city** the limit is **50 km/h**
* Within a **residential** area the limit is **20 km/h**

If the driver is **within the limits**, there should not be any output. If the driver is **over the limit**, however, your function should print the severity of the infraction.

For speeding up to **20** km/hover the limit, speeding should be printed

For speeding up to **40** km/h over the limit, excessive speeding should be printed

For anything else, reckless driving should be printed

The **input** comes as an **array of elements**. The first element is the current speed (**number**), the second element is the area.

The **output** should be printed on the console. Note that in certain cases there isn’t any output.

**Examples**

|  |  |
| --- | --- |
| **Input** | **Output** |
| [40, 'city'] |  |
| [21, 'residential'] | speeding |
| [120, 'interstate'] | excessive speeding |
| [200, 'motorway'] | reckless driving |

1. **Cooking by Numbers**

Write a program that receives a **number** and a **list** of five operations. Perform the operations **sequentially** by starting with the **input number** and using the result of every operation as starting point for the next one. Print the result of every operation in order. The operations can be one of the following:

* **chop** - divide the number by two
* **dice** - square root of number
* **spice** - add 1 to number
* **bake** - multiply number by 3
* **fillet** - subtract 20% from number

The **input** comes as an **array of 6 string elements**. The first element is the starting point and must be **parsed** to a number. The remaining 5 elements are the names of the operations to be performed.

The **output** should be printed on the console.

**Examples**

|  |  |
| --- | --- |
| **Input** | **Output** |
| ['32', 'chop', 'chop', 'chop', 'chop', 'chop'] | 16 8 4 2 1 |

|  |  |
| --- | --- |
| **Input** | **Output** |
| ['9', 'dice', 'spice', 'chop', 'bake', 'fillet'] | 3  4  2  6  4.8 |

1. **Validity Checker**

Write a program that receives two points in the format **[x1, y1, x2, y2].** Check if the distance between each point and the start of the cartesian coordinate system (0, 0) is **valid**. A distance between two points is considered **valid**, if it is an **integer value**.

In case a distance is valid, print"{x1, y1} to {x2, y2} is valid**"**

If the distance is invalid, print **"**{x1, y1} to {x2, y2} is invalid**"**

The order of comparisons should always be first **{x1, y1}** to **{0, 0}**, then **{x2, y2}** to **{0, 0}** and finally **{x1, y1}** to **{x2, y2}**.

The **input** consists of two points given as an **array of numbers**.

For each comparison print either "{x1, y1} to {x2, y2} is valid**"** if the distance is valid, or **"**{x1, y1} to {x2, y2} is invalid**"** if it is invalid.

**Examples**

|  |  |
| --- | --- |
| **Input** | **Output** |
| [3, 0, 0, 4] | {3, 0} to {0, 0} is valid  {0, 4} to {0, 0} is valid  {3, 0} to {0, 4} is valid |
| [2, 1, 1, 1] | {2, 1} to {0, 0} is invalid  {1, 1} to {0, 0} is invalid  {2, 1} to {1, 1} is valid |

1. **\*Coffee Machine**

Write a program for a coffee machine. Calculate whether the money inserted in the machine is enough to make the order and print the corresponding output.

**Input**

The input comes as an **array of strings**. Each string represents one order with different elements, separated by a single space **' '**.

* The **first element** is the **coins inserted**
* The **second** one is the **type of drink** (**coffee or tea**)
* If the drink type is **coffee**, you will receive **'caffeine'** or **'decaf'**
* You may receive **'milk',** if the ordered drink is with milk. **It costs** **10% of the drink price, rounded to first decimal point**
* The last element is the **quantity of sugar, between 0 and 5**. **No matter the quantity (except 0) it costs 0.10. Add the sugar at the end!**

The **prices of drinks** are:

|  |  |
| --- | --- |
| **Type** | **Price** |
| coffee caffeine | 0.80 |
| coffee decaf | 0.90 |
| tea | 0.80 |

**Constrains**

* The input will always be **valid.**

**Output**

For each order there are **two possible** outputs:

* If the money inserted is enough, calculate the change of the order:

**'You ordered {drink}. Price: ${price} Change:** $**{change}'**

* If the money is not enough:

**'Not enough money for {drink}. Need ${moneyNeeded} more'**

After proceeding all orders, print the **total money earned** from the **successful** orders in the format: **'Income Report: ${totalMoney}'**

All of the numbers should be **formatted to the second decimal point**.

**Example**

|  |  |
| --- | --- |
| **Input** | **Output** |
| ['1.00, coffee, caffeine, milk, 4', '0.40, tea, milk, 2', '1.00, coffee, decaf, 0'] | You ordered coffee. Price: $1.00 Change: $0.00  Not enough money for tea. Need $0.60 more.  You ordered coffee. Price: $0.90 Change: $0.10  Income Report: $1.90 |
| **Comments** | |
| The first order is coffee with caffeine, milk and sugar. The price of the drink is $0.80, we calculate the milk, 10% of the price, rounded to the first decimal point - $0.1, and we add the sugar => 0.80 + 0.10 + 0.10 = 1.00.  The second order is tea with milk and sugar (0.80 + 0.10 + 0.10 = 1.00), but the money inserted is not enough.  Next, we receive order for coffee decaf with no milk and 0 sugar => $0.90. The change is $0.10.  Total income = 1.90 | |
| **Input** | **Output** |
| ['8.00, coffee, decaf, 4', '1.00, tea, 2'] | You ordered coffee. Price: $1.00 Change: $7.00  You ordered tea. Price: $0.90 Change: $0.10  Income Report: $1.90 |