# JS Syntax Fundamentals – Exercises

Problems with exercise and homework for the ["JS Front-End" Course @ SoftUni.](https://softuni.bg/trainings/4240/js-front-end-october-2023)

## Ages

Write a function that **determines** whether based on the given **age** a person is: **baby**, **child**, **teenager**, **adult**, **elder**.

The input comes as a **single number parameter**. The bounders are:

* 0-2 (age) – is a baby;
* 3-13 (age) – is a child;
* 14-19 (age) – is a teenager;
* 20-65 (age) – is an adult;
* **>=66 (age) – is an elder;**
* In all other cases print **–** "**out of bounds**";

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

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 20 | adult |
| 1 | baby |
| 100 | elder |
| -1 | out of bounds |

## Vacation

You are given a **group of people**, the **type of the group**, and the **day of the week** they are going to stay. Based on that information **calculate** how much they have to pay and **print** that price on the console. Use the table below. In each cell is the price for a **single person**.

The output should look like that: **`Total price: {price}`**. The **price** should be **formatted** to the second decimal point.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Friday** | **Saturday** | **Sunday** |
| **Students** | 8.45 | 9.80 | 10.46 |
| **Business** | 10.90 | 15.60 | 16 |
| **Regular** | 15 | 20 | 22.50 |

There are also **discounts** based on some conditions:

* **Students –** if the group is bigger than or equal to 30 people you should reduce the **total** price by 15%
* **Business –** if the group is bigger than or equal to 100 people **10** of them can stay **for free**
* **Regular –** if the group is bigger than or equal to 10 and less than or equal to 20 reduce the total price by 5%

**Note: You should reduce the prices in that EXACT order.**

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 30,  "Students",  "Sunday" | Total price: 266.73 |
| 40,  "Regular",  "Saturday" | Total price: 800.00 |

## Leap Year

Write a JS function to check whether a **year** is a **leap**. Leap years are either **divisible** by **4** but not by **100** or are divisible by **400**. The **output** should be following:

* If the year is a leap, print: "**yes**"
* Otherwise, print: "**no**"

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 1984 | yes |
| 2003 | no |
| 4 | yes |

## Print and Sum

Write a function that displays numbers from **given start** to given **end** and their **sum**. The input comes as **two number parameters**. **Print** the result like the examples below:

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 5, 10 | 5 6 7 8 9 10  Sum: 45 |
| 0, 26 | 0 1 2 … 26  Sum: 351 |
| 50, 60 | 50 51 52 53 54 55 56 57 58 59 60  Sum: 605 |

## Multiplication Table

You will receive a **number** as a **parameter**. Print the **10 times table** for this **number**. See the examples below for more information.

### Output

**Print** every row of the table in the following format:

{number} X {times} = {product}

### Constraints

* The number will be an **integer** will be in the interval **[1…100]**

### Examples

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| 5 | 5 X 1 = 5  5 X 2 = 10  5 X 3 = 15  5 X 4 = 20  5 X 5 = 25  5 X 6 = 30  5 X 7 = 35  5 X 8 = 40  5 X 9 = 45  5 X 10 = 50 |  | 2 | 2 X 1 = 2  2 X 2 = 4  2 X 3 = 6  2 X 4 = 8  2 X 5 = 10  2 X 6 = 12  2 X 7 = 14  2 X 8 = 16  2 X 9 = 18   1. X 10 = 20 |

## Sum Digits

Write a **function**, which will be given a single **number**. Your task is to find the **sum** of its digits.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 245678 | 32 |
| 97561 | 28 |
| 543 | 12 |

## Chars to String

Write a **function**, which receives **3 parameters**. Each parameter is a single character. Combine all the characters into **one** string and print it on the console.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 'a',  'b',  'c' | abc |
| '%',  '2',  'o' | %2o |
| '1',  '5',  'p' | 15p |

## Reversed Chars

Write a program that takes **3 parameters** (characters) and prints them in **reversed order** with a space between them.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 'A',  'B',  'C' | C B A |
| '1',  'L',  '&' | & L 1 |

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.

**Examples**

|  |  |
| --- | --- |
| **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. **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 the same and **false** if not. On the next line print the **sum of all 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. **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 be a printed speed and the speed limit.

**`Driving {speed} km/h in a {speed limit} zone`**

If the driver is **over the limit**, however, your function should print the severity of the infraction and the difference in speeds.

**`The speed is {difference} km/h faster than the allowed speed of {speed limit} - {status}`**

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

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

For anything else, **status** should be reckless driving.

The **input** comes as **2 string parameters**. The first element is the current speed (**number**), the second element is the area.

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

**Examples**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 40, 'city' | Driving 40 km/h in a 50 zone |
| 21, 'residential' | The speed is 1 km/h faster than the allowed speed of 20 - speeding |
| 120, 'interstate' | The speed is 30 km/h faster than the allowed speed of 90 - excessive speeding |
| 200, 'motorway' | The speed is 70 km/h faster than the allowed speed of 130 - reckless driving |

1. **Cooking by Numbers**

Write a program that receives 6 parameters which are 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 a 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 a number
* **spice** - add 1 to the number
* **bake** - multiply number by 3
* **fillet** - subtract 20% from the number

The **input** comes as **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 |