# Lab: Data Types and Variables

Problems for exercise and homework for the ["JS Fundamentals" Course @ SoftUni.](https://softuni.bg/trainings/3449/programming-fundamentals-with-javascript-september-2021)   
Submit your solutions in the SoftUni judge system at: <https://judge.softuni.org/Contests/1242>

1. **Concatenate Names**

Write a **function**, which receives two **names** as **string parameters** and a **delimiter**. Print the names **joined** by the delimiter.

**Examples**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 'John',  'Smith',  '->' | John->Smith |
| 'Jan',  'White',  '<->' | Jan<->White |
| 'Linda',  'Terry',  '=>' | Linda=>Terry |

### Hints

Use [string interpolation](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Template_literals).



## Right Place

You will receive **3 parameters (string, char, string).**  
The first string will be a word with a **missing char** replaced with an underscore '**\_**'.  
You have to **replace** the missing character (**underscore**) of the first string with the character passed as the second parameter and **compare** the result with the second string.

If they are equals, you should print "**Matched**", otherwise print "**Not Matched**".

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 'Str\_ng', 'I', 'Strong' | Not Matched |
| 'Str\_ng', 'i', 'String' | Matched |

### Hints



## Integer and Float

You will receive **3 numbers**. Your task is to find their **sum** and print result to the console in the following format:  
**`{sum} - {type of the number (Integer or Float)}`**

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 9, 100, 1.1 | 110.1 - Float |
| 100, 200, 303 | 603 - Integer |

### Hints



## Amazing Numbers

Write a **function**, which as **input** will receive a **number**.

**Check** and print if it is **amazing** or **not** into the following format:

**"{number} Amazing? {True or False}"**

An amazing number includes the **digit 9** the sum of its digits.

**Examples** for amazing numbers are 1233 (1 + 2 + 3 + 3 = 9), 583472 (5 + 8 + 3 + 4 + 7 + 2 = 29)

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 1233 | 1233 Amazing? True |
| 999 | 999 Amazing? False |

### Hints

Use **includes()**



## Gramophone

Write a **function**, which as **input** will receive **3 parameters (strings)**

* **The first string** is the name of the **band**
* **The second string** is the name of the **album**
* **The third** is holding a **song** name from the album

You have to find out how many **times** the plate will **rotate** the given song from the album.

The plate makes a full rotation every **2.5** seconds.

The song **duration in seconds** is calculate by the given formula:

**(albumName.length \* bandName.length) \* song-name.length / 2**

As **output,** you should print the following message:

**`The plate was rotated {rotations} times.`**

Rotations should be **rounded up**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 'Black Sabbath', 'Paranoid', 'War Pigs' | The plate was rotated 167times. |
| 'Rammstein', 'Sehnsucht', 'Engel' | The plate was rotated 81 times. |

### Hints



## Required reading

Write a **function** to help **Ivan** calculate how many hours a day he has to spend reading the necessary literature from the list given for the summer vacation.

As **input,** you will receive **3 parameters:**

* **Number of pages of the current book** - **integer** [1… 1000]
* **Pages read in 1 hour** - **integer** [1… 1000]
* **The number of days for which you must read the book** - **integer** [1… 1000]

As **output** print on the console the **number of hours**, that Ivan has to read each day.

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Explanations** |
| 212,  20 ,  2 | 5.3 | Total time to read the book: **212 pages / 20 pages per hour = 10.6 hours**  Required hours per day: **10.6 hours / 2 days = 5.3 hours** per day |
| **Input** | **Output** |  |
| 432,  15 ,  4 | 7.2 | Total reading time of the book: **432 pages / 15 pages per hour = 28.8 hours**  Required hours per day: **28.8 hours / 4 days = 7.2 hours** per day |

## Centuries to Minutes

Write a program that receives a **number** of **centuries** and converts it to **years**, **days**, **hours**, and **minutes**.

**Examples**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 1 | 1 centuries = 100 years = 36524 days = 876576 hours = 52594560 minutes |
| 5 | 5 centuries = 500 years = 182621 days = 4382904 hours = 262974240 minutes |

**Hint**

* Assume that a year has 365.2422 days on average ([the Tropical year](https://en.wikipedia.org/wiki/Tropical_year)).

**Solution**

You might help yourself with the code below:



1. **Special Numbers**

Write a program that receives a number **n.** For all numbers in the range **[1…n]** print the number and if it is special or not (**True** / **False**).

* A **number** is **special** when its **sum of digits is 5, 7 or 11**.

**Examples**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 15 | 1 -> False  2 -> False  3 -> False  4 -> False  5 -> True  6 -> False  7 -> True  8 -> False  9 -> False  10 -> False  11 -> False  12 -> False  13 -> False  14 -> True  15 -> False |
| 20 | 1 -> False  2 -> False  3 -> False  4 -> False  5 -> True  6 -> False  7 -> True  8 -> False  9 -> False  10 -> False  11 -> False  12 -> False  13 -> False  14 -> True  15 -> False  16 -> True  17 -> False  18 -> False  19 -> False  20 -> False |

**Hints**

To calculate the sum of digits of given number **num**, you might repeat the following: sum the last digit (**num** **%** **10**) and remove it (**sum** **=** **sum** **/** **10**) until **num** reaches **0**. Use **parseInt()** while dividing to get only integer numbers.

1. **Triples of Latin Letters**

Write a program that receives a string of **number** **n** and print all **triples** of the first **n small Latin letters**, ordered alphabetically:

**Examples**

|  |  |
| --- | --- |
| **Input** | **Output** |
| '3' | aaa  aab  aac  aba  abb  abc  aca  acb  acc  baa  bab  bac  bba  bbb  bbc  bca  bcb  bcc  caa  cab  cac  cba  cbb  cbc  cca  ccb  ccc |
| 2 | aaa  aab  aba  abb  baa  bab  bba  bbb |

**Hints**

Perform 3 nested loops from **0** to **n**. For each number **num** print its corresponding Latin letter as follows:



The function **String.fromCharCode()** gets the value in **decimal** and transforms it to a character from the **ASCII table**.