String Processing - Exercises

[1. Print Characters 1](#_Toc146711811)

[2. Substring 1](#_Toc146711812)

[3. Censor 2](#_Toc146711813)

[4. Reveal Words 2](#_Toc146711814)

[5. #HashTag 2](#_Toc146711815)

[6. Extract File 3](#_Toc146711816)

[7. Substring 3](#_Toc146711817)

[8. Replace Repeating Symbols 4](#_Toc146711818)

[9. Pascal-Case Splitter 4](#_Toc146711819)

[10. Cut and Reverse 4](#_Toc146711820)

[11. Letter 4](#_Toc146711821)

[12. Match Full Name 5](#_Toc146711822)

[13. Match Phone Number 5](#_Toc146711823)

[14. Match Dates 6](#_Toc146711824)

[15. Star Battles Enigma 7](#_Toc146711825)

# Print Characters

Write a function that **receives a** **string** and **prints all the** **characters** on separate lines.

|  |  |
| --- | --- |
| Input | Output |
| 'AWord' | A  W  o  r  d |
| 'Sentence' | S  e  n  t  e  n  c  e |

# Substring

Write a function that **receives a string** and **two numbers**. The numbers will be a **starting index** and **count** of elements to substring. Print the result.

|  |  |
| --- | --- |
| Input | Output |
| 'ESentence', 1, 8 | Sentence |
| 'DropWord', 4, 7 | Word |

# Censor

Write a function that **receives a text as** a first parameter and a **single word** as a second. Find **all occurrences** of that word in the text and replace them with the corresponding count of **'\*'**.

The repeat() function should take the length of the word and return that amount of stars '\*'.

|  |  |
| --- | --- |
| Input | Output |
| 'A small sentence with some words', 'small' | A \*\*\*\*\* sentence with some words |
| 'Find the hidden word', 'hidden' | Find the \*\*\*\*\*\* word |
| 'A small sentence with small words', 'small' | A \*\*\*\*\* sentence with \*\*\*\*\* words |

# Reveal Words

Write a function, which receives **two parameters**.

The first parameter will be a string with some words **separated by ', '**.

The second parameter will be a string that contains **templates containing '\*'**.

Find the word with the **same length** as the template and **replace** it.

|  |  |
| --- | --- |
| Input | Output |
| 'great',  'JavaScript\* is \*\*\*\*\* programming language' | JavaScript\* is great programming language |
| 'the, best, learn',  'JavaScript is \*\*\* \*\*\*\* language to \*\*\*\*\*' | JavaScript is the best language to learn |

# #HashTag

Receive a **single string. Find all** special words **starting with #**. If the found special word does not consist only of letters, then it is invalid and should not be printed.

Finally, print out all the hashtags you found without the label **(#)** on a new line.

|  |  |
| --- | --- |
| Input | Output |
| 'Everyone uses # to tag a #special word in #facebook | special  facebook |
| 'The symbol # is known #variously in English-speaking #regions as the #number sign' | variously  regions  number |

# Extract File

Write a function that receives a single string - the path to a file (the '**\**' character is escaped)

Your task is to subtract the **file name** and its **extension**. (Beware of files like **template.error.pptx,** as **template.error** should be the file name, while **pptx** is the extension).

|  |  |
| --- | --- |
| Input | Output |
| 'C:\\desktop\\academy\\template.pptx' | File name: template  File extension: pptx |
| 'C:\\Projects\\website.folder\\file.name.js' | File name: file.name  File extension: js |

# Substring

The input will be given as **two** separate strings(a **word** as a first parameter and a **text** as a second).

Write a function that checks the text for the given word. The comparison should be **case insensitive.** Once you find a match, **print** the word.

If you don't find the word print: **"{word} not found!"**

|  |  |
| --- | --- |
| Input | Output |
| 'javascript', 'JavaScript is the best programming language' | javascript |
| 'python',  'JavaScript is the best programming language' | python not found! |

# Replace Repeating Symbols

Write a function that receives a single string and **replace** any sequence of the **same symbols** with a single corresponding letter.

|  |  |
| --- | --- |
| Input | Output |
| 'aaaaabbbbbcdddeeeedssaa' | abcdedsa |
| 'qqqwerqwecccwd' | qwerqwecwd |

# Pascal-Case Splitter

You will receive a **single** **string**.

This string is written in **PascalCase** format. Your task here is to split this string by **every word** in it.

Print them joined by **comma** and **space.**

|  |  |
| --- | --- |
| Input | Output |
| 'SplitMeIfYouCan' | Split, Me, If, You, Can |
| 'HoldTheDoor' | Hold, The, Door |
| 'ThisIsSoAnnoyingToDo' | This, Is, So, Annoying, To, Do |

# Cut and Reverse

Write a function that cuts the given string **into half** and **reverses** the **two halves.**

Print each half on a **separate line.**

|  |  |
| --- | --- |
| Input | Output |
| 'tluciffiDsIsihTgnizamAoSsIsihT' | ThisIsDifficult  ThisIsSoAmazing |
| 'sihToDtnaCuoYteBIboJsihTtAdooGoSmI' | IBetYouCantDoThis  ImSoGoodAtThisJob |

# Letter

You will receive an **array**, which holds the **string** and **another array**.

The string is a letter which has a few **holes**, you must fill with **strings from the array** you receive at the second index.

If the **length** oftheholeis **4** youmust **replace** itwith **string** withthe **same length** and so on…

**Examples**

|  |
| --- |
| Input |
| 'Hi, grandma! I\'m so \_\_\_\_ to write to you. \_\_\_\_\_\_ the winter vacation, so many \_\_\_\_\_\_\_ things happened. My dad bought me a sled. Mom started a new job as a \_\_\_\_\_\_\_\_\_\_. My brother\'s ankle is \_\_\_\_\_\_\_\_, and now it bothers me even more. Every night Mom cooks \_\_\_ on your recipe because it is the most delicious. I hope this year Santa will \_\_\_\_\_ me a robot.', ['pie', 'bring', 'glad', 'During', 'amazing', 'pharmacist', 'sprained'] |
| Output |
| Hi, grandma! I'm so glad to write to you. During the winter vacation, so many amazing things happened. My dad bought me a sled. Mom started a new job as a pharmacist. My brother's ankle is sprained, and now it bothers me even more. Every night Mom cooks pie on your recipe because it is the most delicious. I hope this year Santa will bring me a robot. |

This lasagna recipe takes a little work, but it is so satisfying and filling that it's worth it!

# Match Full Name

Write a JavaScript function to **match full names** from a list of names and **print** them on the console.

|  |
| --- |
| Input |
| "Ivan Ivanov, Ivan ivanov, ivan Ivanov, IVan Ivanov, Test Testov" |
| Output |
| Ivan Ivanov, Test Testov |

# Match Phone Number

Write a regular expression to match a **valid phone number** from **Sofia**. After you find all **valid phones**, **print** them on the console, separated by a **comma and a space ", "**.

**Compose the Regular Expression**

A valid number has the following characteristics:

* It starts with **"+359"**
* Then, it is followed by the area code (always **2**)
* After that, it’s followed by the **number** itself:
  + The number consists of **7 digits** (separated into **two** **groups** of **3** and **4** **digits** respectively).
* The different **parts** are **separated** by **either a space or a hyphen** (**'-'**).

You can use the following RegEx properties to **help** with the matching:

* Use **quantifiers** to match a **specific number** of **digits**
* Use a **capturing group** to make sure the delimiter is **only one of the allowed characters** **(space or hyphen)** and **not** a **combination** of both (e.g. **+359 2-111 111** has **mixed delimiters**, it is **invalid**). Use a **group backreference** to achieve this.
* Add a **word boundary** at the **end** of the match to avoid **partial matches**
* Ensure that before the **'+'** sign there is either a **space** or the **beginning of the string**.

|  |
| --- |
| Input |
| ['+359 2 222 2222,359-2-222-2222, +359/2/222/2222, +359-2 222 2222 +359 2-222-2222, +359-2-222-222, +359-2-222-22222 +359-2-222-2222'] |
| Output |
| +359 2 222 2222, +359-2-222-2222 |
| Input |
| ['+359 2 357 3351 +359 2 22 2222 +359 2 173 3408 +359-2-789-2584 +359 2 193 3953 +359-2-961-0248 +359-2-789-2584 +359 2 222 222 +360 2 222 2222 +359 2 727 9740 +359-2-854-2280 +359 2 193 3953 +359 2 357 3351 +359 2 558 8560 +359 2 222 222'] |
| Output |
| +359 2 357 3351, +359 2 173 3408, +359-2-789-2584, +359 2 193 3953, +359-2-961-0248, +359-2-789-2584, +359 2 727 9740, +359-2-854-2280, +359 2 193 3953, +359 2 357 3351, +359 2 558 8560 |

# Match Dates

Write a program, which matches a date in the format **"dd{separator}MMM{separator}yyyy"**.

Every valid date has the following characteristics:

* Always starts with **two digits**, followed by a **separator**
* After that, it has **one uppercase** and **two lowercase** letters (e.g. **Jan**, **Mar**).
* After that, it has a **separator** and **exactly 4 digits** (for the year).
* The separator could be either of three things: a period (**"."**), a hyphen (**"-"**) or a forward-slash (**"/"**)
* The separator needs to be **the same** for the whole date (e.g. 13**.**03**.**2016 is valid, 13**.**03**/**2016 is **NOT**).

|  |
| --- |
| Input |
| ['13/Jul/1928, 10-Nov-1934, 01/Jan-1951, 25.Dec.1937, 23#09#1973, 1/Feb/2016'] |
| Output |
| Day: 13, Month: Jul, Year: 1928  Day: 10, Month: Nov, Year: 1934  Day: 25, Month: Dec, Year: 1937 |
| Input |
| ['1/Jan-1951 23/October/197 11-Dec-2010 18.Jan.2014'] |
| Output |
| Day: 11, Month: Dec, Year: 2010  Day: 18, Month: Jan, Year: 2014 |

…for the first time since the War of the Pacific…

# Star Battles Enigma

The war is at its peak, but you, young Padawan, can turn the tides with your programming skills. You are tasked to create a program to **decrypt** the messages of The Order and prevent the death of hundreds of innocents.

You will receive several messages, which are **encrypted** using the legendary star enigma. You should **decrypt the messages**, following these rules:

To properly decrypt a message, you should **count all the letters** **[s, t, a, r]** – **case insensitive** and **remove** the count from the **current ASCII value of each symbol** of the encrypted message.

After decryption:

Each message should have a **planet name, population, attack type ('A', as an attack or 'D', as destruction), and soldier count.**

The planet name **starts after** **'@'** and contains **only letters from the Latin alphabet**.

The planet population **starts after ':'** and is an **Integer**;

The attack type may be **"A"(attack) or "D"(destruction)** and must be **surrounded by "!"** (exclamation mark (lightsaber 😉)).

The **soldier count** starts after **"->"** and should be an Integer.

The order in the message should be: **planet name -> planet population -> attack type -> soldier count.** Each part can be separated from the others by **any character except: '@', '-', '!', ':' and '>'.**

After decrypting all messages, you should print the decrypted information in the following format:

First print the attacked planets, then the destroyed planets.  
**"Attacked planets: {attackedPlanetsCount}"  
"-> {planetName}"  
"Destroyed planets: {destroyedPlanetsCount}"  
"-> {planetName}"**

|  |  |  |
| --- | --- | --- |
| Input | Output | Comments |
| ['STCDoghudd4=53333$D$0A53333',  'EHfsytsnhf?8555&I&2C9555SR'] | Attacked planets: 1  -> Alderaa  Destroyed planets: 1  -> Cantonica | We receive two messages, to decrypt them we calculate the key:  The first message has decryption key 3. we subtract from each character’s code 3.  **PQ@Alderaa1:20000!A!->20000**  The second message has key 5.  **@Cantonica:3000!D!->4000NM**  **Both messages are valid** |
| Input | **Output** |  |
| ["tt(''DGsvywgerx>6444444444%H%1B9444",  'GQhrr|A977777(H(TTTT',  'EHfsytsnhf?8555&I&2C9555SR'] | Attacked planets: 0  Destroyed planets: 2  -> Coruscant  -> Cantonica |  |

"It’s a trap!" – Admiral Ackbar