# Exercises: PHP Basics

Problems for exercises and homework for the [“Software Technologies” course @ SoftUni](https://softuni.bg/courses/software-technologies).

You can submit your solutions here <https://judge.softuni.bg/Contests/240/PHP-Basics-Exercises>.

**Important: Upload only your PHP code!**

## Reverse Numbers

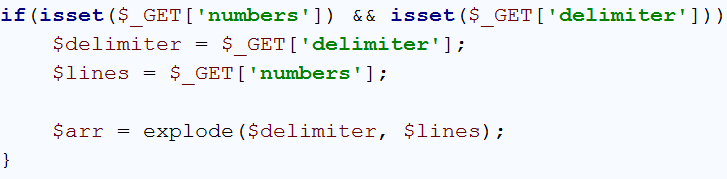
You will be given a few numbers as input. You need to print them in backward order, each on a new line.

### Examples

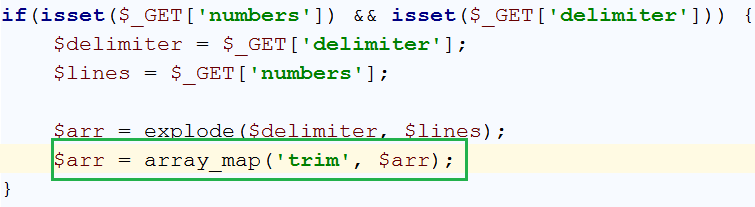
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Parameter name** | **Input** | **Output** |  | **Input** | **Output** |
| numbers | 1  2 | 2<br>1<br> |  | 1pesho2pesho3pesho4 | 4<br>3<br>2<br>1<br> |
| delimiter | (new line) |  | pesho |

### Hints

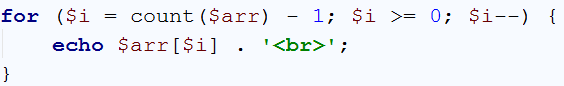
* We need to check if the input values are set, and if so, we extract them with $\_GET. This time our input will come in the form of **multiple lines** and we need to store them somewhere. An **array** is perfect for this use. However, we receive the input in the form of a string, which we must **explode** in order to get the array. Splitting a string with a delimiter in PHP is done with function explode(). We are also given a custom delimiter by the input, so we need to split the array with it.



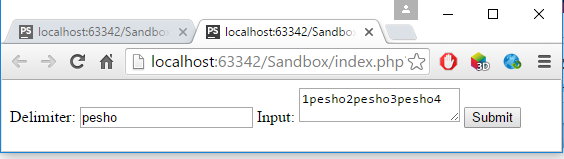
* After that we need to check if there are any unneeded spaces left, and if there are, we need to trim them. This is done by mapping the array, applying the trim function to each element, using the function array\_map().

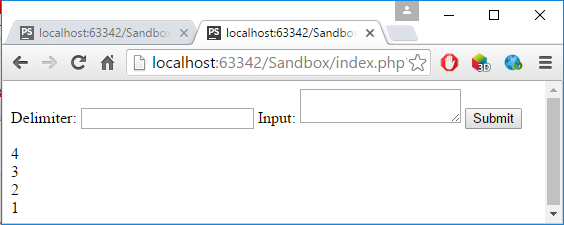


* After we’ve extracted the input and stored it into an array, we can safely traverse the array with a for loop and print all of it’s elements in backward order – as the problem requires



* If you’ve followed all the steps correctly, you should see the following results:





## Print Lines

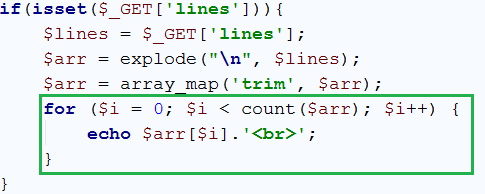
You will be given lines of text. Print each of those lines at the moment you read them, until you reach the command **“Stop”**. Do **NOT** print the ending command.

### Examples

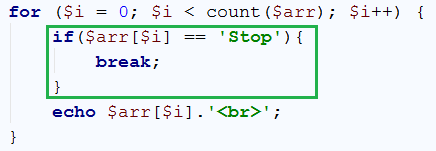
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Parameter name** | **Input** | **Output** |  | **Input** | **Output** |
| lines | Line 1  Line 2  Line 3  Stop | Line 1<br>Line 2<br>Line 3<br> |  | 3  6  5  4  Stop  10  12 | 3<br>6<br>5<br>4<br> |

### Hints

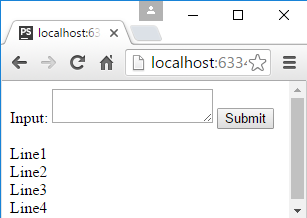
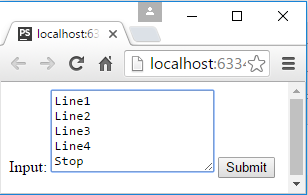
* You must take your input in a similar way to the previous problem
* After we’ve extracted the input and stored it into an array, we can safely traverse the array with a for loop and print all of it’s elements – as the problem requires.



* However, we must stop printing when we receive the input command – Stop. Simple – we just add an if statement in the for loop, and break it when we receive the command.



* If you’ve followed all the steps correctly, you should see the following results:



## Custom Delimiter

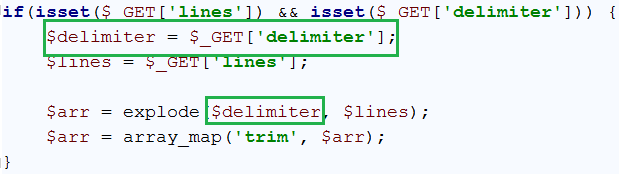
You will be given lines of text. Those lines will be separated with custom delimiter. Print each of those lines at the moment you read them, until you reach the command **“Stop”**. Do **NOT** print the ending command.

### Examples

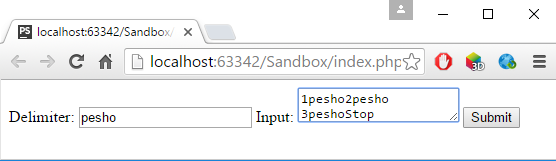
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Parameter name** | **Input** | **Output** |  | **Input** | **Output** |
| lines | 1pesho2pesho3peshoStop | 1<br>2<br>3<br> |  | 1 pesho 2 gosho 3 stamat Stop pesho 1 2 3 | 1<br>pesho<br>2<br>gosho<br>3<br>stamat<br> |
| delimiter | pesho |  | (space) |

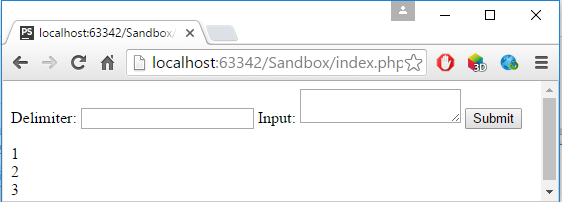
### Hints

* Following the previous problem, aside from a hardcoded string, we should be able to explode a string with a custom delimeter. Basically, everything is the same except, you pass the extracted input delimeter, after, ofcourse, you’ve checked if it’s value is set, to the explode() function, instead of a hardcoded string.



* If you’ve done everything correctly, you should see the followng:





## Array Indexes

You will be given **array of elements**. You will receive the **length of that array**. Then you will start receiving an **index** and a **value**, separated by a **custom delimiter**, that you will also receive. For each received line you **must set the value at the given index to the given value**. When you’ve processed all input data, **print the array’s** elements each on a **new line**.

### Examples

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Parameter name** | **Input** | **Output** |  | **Input** | **Output** |
| key-value-pairs | 0 – 5  1 – 6  2 - 7 | 5<br>6<br>7<br> |  | 0$$$5  0$$$6  0$$$7 | 7<br>0<br>0<br> |
| delimiter | (space)-(space) |  | $$$ |
| array-size | 3 |  | 3 |

### Hints

* You need to use your **custom delimiter** to split the strings
* Create an array and fill all of the cells with ‘0’ in the beginning

## Add / Remove Elements

You will be given a sequence of **commands** (pairs of elements separated by a custom delimiter): **command** and **argument**. You start by an empty array.

* The command “**add**” appends the given **number** to the array.
* The command “**remove**”removes the element at the specified **index**. If the index is nonexistent, ignore that input line. When an element is deleted, all elements on the right from it, go one position left.

When you process all input data, **print the array’s elements** each on a separate line.

### Hints

* Create a for loop that traverses the commands and explode each command by the given delimiter
* After exploding the command, check if the first part is add or remove.
* There is an array\_splice() function, that works similarly to the JavaScript function.

### Examples

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Parameter name** | **Input** | **Output** |  | **Input** | **Output** |
| commands | add 1  add 2  add 3 | 1<br>2<br>3<br> |  | add$$$1  add$$$2  add$$$3  remove$$$0 | 2<br>3<br> |
| delimiter | (space) |  | $$$ |

## Key-Value Pairs

You will be given lines of text, each holding **two elements** separated by a custom delimiter. The first is the **key** and the second is the **value**. Your task is to store the value for each key. If a key already exists, you need to **replace** the old value with the new one. At the last line of input you will receive a **key**. You must print the **value** corresponding to that **key**. If there is no such, just print “**None**”.

### Examples

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Parameter name** | **Input** | **Output** |  | **Input** | **Output** |
| key-value-pairs | gosho dava  pesho dava  stamat dava  mariika nedava | dava |  | key#value  key#eulav  test#tset | eulav |
| delimiter | (space) |  | # |
| target-key | stamat |  | key |

## Key-Multi-Values

You will be given input lines, each holding **two elements** separated by a space: **key** and **value**. You need to store the given values to the given keys. At the last line of the input you will receive a **key**. You must **print all the values** corresponding to that key. If there are no such, just print “**None**”.

### Examples

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Parameter name** | **Input** | **Output** |  | **Input** | **Output** |
| key-value-pairs | key value  key eulav  test tset | value<br>eulav |  | 3#test  3#test1  4#test2  4#test3  4#test5 | test2<br>test3<br>test5 |
| delimiter | (space) |  | # |
| target-key | key |  | 4 |

### Hints

* Each time you split a row, check if the key already exists
* If it doesn’t exist, create a new array for that key and add the value in that nested array
* If it exists just add the value
* Use the function implode() to join the array by **new line**, when you need to print it

## Storing Objects

You will be given input lines, each holding information about a **student**: **name**, **age** and **grade**. The data comes in the following format:

“{STUDENT\_NAME}{CUSTOM\_DELIMITER}{STUDENT\_AGE}{CUSTOM\_DELIMITER}{STUDENT\_GRADE}”

Extract that information from the input lines into **PHP objects**.

**Print** the objects in **the order of appearance**, using **unordered** **list** for every person and list his information as **list** **items**.

### Examples

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter Name** | **Input** | **Output** | **Output Image** |
| input | Pesho -> 13 -> 6.00  Ivan -> 12 -> 5.57  Toni -> 13 -> 4.90 | <ul><li>Name: Pesho</li><li>Age: 13</li><li>Grade: 6</li></ul><ul><li>Name: Ivan</li><li>Age: 12</li><li>Grade: 5.57</li></ul><ul><li>Name: Toni</li><li>Age: 13</li><li>Grade: 4.9</li></ul> |  |
| delimiter | (space)->(space) |

## Object to JSON String

You will be given **input lines** holding information about an **object** in the form of **key / value pairs**. The key/value pairs will define **class members** and **their values**. The key/value pairs will be **separated by a custom delimiter**, that you will receive.

Store the data into a **PHP object**. The input data will **always** be the same. Create a **PHP function** that turns the PHP object into **JSON string**. Print the **JSON** version of the object.

Use the json\_encode(object) function. When encoding an object into JSON, PHP tends to escape special characters, such as **slashes in dates**. However, the json\_encode() function can be given a **second parameter**, which can help you avoid that problem. Give the function, as second parameter, JSON\_UNESCAPED\_SLASHES.



### Examples

|  |  |  |
| --- | --- | --- |
| **Parameter name** | **Input** | **Output** |
| input | name -> Angel  surname -> Georgiev  age -> 20  grade -> 6.00  date -> 23/05/1995  town -> Sofia | {"name":"Angel","surname":"Georgiev","age":20,"grade":6,"date":"19/05/1995","town":"Sofia"} |
| Delimiter | -> |

## Dates

You will be given a **date** that you need to modify. This date will be in format “d/M/y”. Then you will start modifying that date with sequence of **commands**.

* The command “**add**” contains only 1 argument -> number **N**. This command adds **N** days to the date.
* The command “**subtract**” contains only 1 argument -> number **N**. This command removes **N** days from the date.

### Examples

|  |  |  |
| --- | --- | --- |
| **Parameter name** | **Input** | **Output** |
| commands | subtract 20 | 11 June 2016 |
| date | 01/07/2016 |
| format | d F y |