# Homework: PHP Syntax

This document defines the homework assignments from the ["PHP Fundamentals" Course @ Software University](https://softuni.bg/trainings/1746/php-web-developmentbasics-september-2017). Please submit ……………………………………………….???

## Personal Info

Write a PHP script **PersonalInfo.php**. Declare a few variables. The **first** variable should hold your **first name**, the **second** should hold your **last name**, the **third -** your **age**, and the last one should hold your **full name** (use **concatenation**). The result should be printed. Sample output:

|  |
| --- |
| **Output** |
| My name is Mister DakMan and I am 21 years old. |
| My name is Pesho Peshev and I am 55 years old. |

## Sum Two Numbers

Write a PHP script **SumTwoNumbers.php** that decleares two variables, **firstNumber** and **secondNumber.** They should hold integer or floating-point numbers (hard-coded values). Print their **sum** in the output in the format shown in the examples below. The numbers should be **rounded to the second** number after the decimal point. Find in Internet how to **round** a given number in PHP. Examples:

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2  5 | $firstNumber + $secondNumber = 2 + 5 = 7.00 |
| 1.567808  0.356 | $firstNumber + $secondNumber = 1.567808 + 0.356 = 1.92 |
| 1234.5678  333 | $firstNumber + $secondNumber = 1234.5678 + 333 = 1567.57 |

## Dump Variable

Write a PHP script **DumpVariable.php** that declares a variable. If the variable is **numeric**, **print it** by the built-in function **var\_dump()**. If the variable is **not numeric**, **print its type** at the input. Examples:

|  |  |
| --- | --- |
| **Input** | **Output** |
| "hello" | string |
| 15 | int(15) |
| 1.234 | float(1.234) |
| array(1,2,3) | array |
| (object)[2,34] | object |

## Interval of Numbers

Write a program, which takes **two numbers** as input and prints the **interval of numbers between them**, **starting** from the **smaller one** and **ending** with the **larger** one.

### Input

You will receive **two lines**. Each of them will contain **one integer**.

### Output

Print all the numbers separated on **new lines**.

### Constraints

* The numbers, which you receive will be in the interval **[0…100]**.
* The two numbers, which you take as an input will **not be equal**.

### Examples

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| 42  48 | 42  43  44  45  46  47  48 | 100  14 | 14  15  16  *continues...*  98  99  100 |

## Restaurant Discount

A restaurant wants to automate their reservation process. They need a program that reads the **count of people** andthe **package** from the console and calculates **how much** the customer should **pay** to book the place.

Different halls have different prices:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Small Hall** | **Terrace** | **Great Hall** |
| **Price** | 2500$ | 5000$ | 7500$ |
| **Capacity** | 50 | 100 | 120 |

The restaurant has **discounts** depending on the **service package,** which the group wants.

You can see the discounts in the table below:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Normal** | **Gold** | **Platinum** |
| **Discount** | 5% | 10% | 15% |
| **Price** | 500$ | 750$ | 1000$ |

You should **add** the **price** of the **package** to the **price** of the **hall**. The discount is calculated based on the **hall’s price + package’s price**.

Example: The group has **10 people** and wants the **gold package  $292.50 per person**:

* **10 people** <= 50  they get the **Small Hall**  $2500
* Gold package  **$750**, **10%** discount on the entire purchase
* Total price: **$2500 + $750** = **$3250**
* Discount: $3250 - **10% discount** = $2925
* Price per person: $2925 / **10 people** = **$292.50 per person**

You should print **which hall** is the **most suitable** for the group and the **price per person**. If the group is **bigger than 120** people – print “**We do not have an appropriate hall.**”.

### Input

* First line: the **group size** as an integer.
* Second line: the **type** of the **package** as **string**

### Output

Print the output in the following format:

|  |
| --- |
| We can offer you the {hallName}  The price per person is {price}$ |

**Format** the **price** to the **2nd decimal place**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 20  Gold | We can offer you the **Small Hall**  The price per person is **146.25$** |
| 90  Platinum | We can offer you the **Terrace**  The price per person is **56.67$** |
| 150  Normal | We do not have an appropriate hall. |

## String Length

Write a program that reads from the console a string of maximum 20 characters. If the length of the string is less than 20, the rest of the characters should be filled with \*. Print the resulting string on the console.

Examples:

|  |  |
| --- | --- |
| **Input** | **Output** |
| Welcome to SoftUni! | Welcome to SoftUni!\* |
| a regular expression (abbreviated regex or regexp and sometimes called a rational expression) is a sequence of characters that forms a search pattern | a regular expression |
| PHP | PHP\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |

## Find the Letter

Write a program, which receives a string and prints the index of a given letter in the string. The tricky part is that you will have to find **not** the first letter, but the **nthletter**.

### Input

* On the first line, you will receive the **string** you are going to search through.
* On the **second** line, you will receive an **array** with exactly **two** **elements**:
  + The **first** element will be the **letter**, which you have to search for.
  + The **second** element will be an **integer N**, showing us which **occurrence** ofthe **letter** we are searching for.

**Example**: If we receive the string “Programming is awesome!” and on the next line we receive the array “m 3”. We should find the **third** occurrence of the letter ‘**m**’. It can be found on **20th** index.

### Output

If the nth occurrence of the letter is present in the string, print **the index** of that occurrence.

If the letter is **not present** in the string, or there are **less occurrences** than **N**, print:

* “Find the letter yourself!”

*Note: the* ***comparison*** *should be* ***case******sensitive****.*

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Programming is awesome!  m 3 | 20 |
| Strings, strings everywhere...  e 5 | Find the letter yourself! |

## Points Counter

Write a program, which receives data about a **team**, **player** and **points**.

### Input

You can have **two** types of input:

* {TEAM}|{Player}|{points}
* {Player}|{TEAM}|{points}

The **team** **name** will always consist of **uppercase** **Latin** **letters** and the player name will always **start** with **uppercase** **Latin** **letter** and **all** **other** **letters** will be **lowercase**.

The **team** and **player** names **might** be **polluted** with some **prohibited** symbols (‘**@**’, ‘**%**’, ‘**$**’ and ‘**\***’). You have to **delete** **every** **occurrence** of these symbols in **every** **team** and **player** name.

Then, calculate every team’s total score. Every **team’s** **total** **score** equals to the **total** **sum** of the **points** made by **every** **player** **in** the **team**.

### Output

When you receive the command “**Result**”, print **all** teams, ordered in **descending order** by the **points** they made and the player with **most** points in the **team** in the following format:

|  |
| --- |
| {teamName} => {totalSumOfPoints}  Most points scored by: {nameOfThePlayer} |

In case of **repeating** **player** **names** for one team, **save** the **value**, which is **received** **last**.

### Constraints

* The **team** **names** will be **at least** **2** characters **long**
* The **points** for each **player** will be in the interval [1…100]

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| LA|Bryant|70  L%@A|Odom|67  James|%CAVA@@LIE$$$RS|54  C@art%er|GR%%IZZ%%LIE@S@@@|49  Anthony|KNICKS|11  UTAH|Jo%%%%hn$$so@@n|24  S@@PU\*R\*S$|Ga\*\*\*so\*\*l|32  Jone@@@@s|KNICKS|5  Result | LA => 137  Most points scored by Bryant  CAVALIERS => 54  Most points scored by James  GRIZZLIES => 49  Most points scored by Carter  SPURS => 32  Most points scored by Gasol  UTAH => 24  Most points scored by Johnson  KNICKS => 16  Most points scored by Anthony |
| SO@@@FTU%\*NI|P\*&@esho|30  SO$$FT\*UNI|Gos%@ho|42  PAARTHURNAX|Maria|35  S\*OFT$$$UNI|Iv\*\*\*\*@an|3  L@u\*b%o@|HE\*\*\*RO@@ES|11  Result | SOFTUNI => 75  Most points scored by Gosho  PAARTHURNAX => 35  Most points scored by Maria  HEROES => 11  Most points scored by Lubo |

## Max Sequence of Equal Elements

Write a program that finds the **longest sequence of equal elements** in an array of integers. If several longest sequences exist, print the leftmost one.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2 1 1 2 3 3 **2 2 2** 1 | 2 2 2 |
| **1 1 1** 2 3 1 3 3 | 1 1 1 |
| **4 4 4 4** | 4 4 4 4 |
| 0 **1 1** 5 2 2 6 3 3 | 1 1 |

### Hints

* Start with the sequence that consists of the first element: start=0, len=1.
* Scan the elements from left to right, starting at the second element: pos=1…n-1.
  + At each step compare the current element with the element on the left.
    - Same value  you have found a sequence longer by one  len++.
    - Different value  start a new sequence from the current element: start=pos, len=1.
  + After each step remember the sequence it is found to be longest at the moment: bestStart=start, bestLen=len.
* Finally, print the longest sequence by using bestStart and bestLen.

## \*Max Sequence of Increasing Elements

Write a program that finds the **longest increasing subsequence** in an array of integers. The longest increasing subsequence is a **portion of the array** (subsequence) that is strongly **increasing** and has the **longest possible length**. If several such subsequences exist, find the left most of them.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3 **2 3 4** 2 2 4 | 2 3 4 |
| 4 5 **1 2 3 4 5** | 1 2 3 4 5 |
| **3 4 5 6** | 3 4 5 6 |
| **0 1** 1 2 2 3 3 | 0 1 |

### Hints

* Use the same algorithm like in the previous problem (Max Sequence of Equal Elements).

## \*Non-Repeating Digits

Write a PHP script **NonRepeatingDigits.php** that declares an integer variable **N**, and then finds all 3-digit numbers that are less or equal than **N (<= N)** and consist of unique digits. Print "no" if no such numbers exist. Examples:

|  |  |
| --- | --- |
| **Input** | **Output** |
| 1234 | 102, 103, 104, 105, 106, 107, 108, 109, 120, 123, 124, 125, 126, 127, 128, 129, 130, 132, 134, 135, …, 980, 981, 982, 983, 984, 985, 986, 987 |
| 145 | 102, 103, 104, 105, 106, 107, 108, 109, 120, 123, 124, 125, 126, 127, 128, 129, 130, 132, 134, 135, 136, 137, 138, 139, 140, 142, 143, 145 |
| 15 | no |
| 247 | 102, 103, 104, 105, 106, 107, 108, 109, 120, 123, 124, 125, 126, 127, 128, 129, 130, 132, 134, 135, 136, 137, 138, 139, 140, 142, 143, 145, 146, 147, 148, 149, 150, 152, 153, 154, 156, 157, 158, 159, 160, 162, 163, 164, 165, 167, 168, 169, 170, 172, 173, 174, 175, 176, 178, 179, 180, 182, 183, 184,  185, 186, 187, 189, 190, 192, 193, 194, 195, 196, 197, 198, 201, 203, 204, 205, 206, 207, 208, 209, 210, 213, 214, 215, 216, 217, 218, 219, 230, 231, 234, 235, 236, 237, 238, 239, 240, 241, 243, 245, 246, 247 |

## \*Lazy Sundays

Write a PHP script **LazySundays.php** which prints the **dates** of all Sundays in the current month. Example:

## HTML Table

Write a PHP script **InformationTable.php** which **generates an HTML table** by given information about a person (**name**, **phone number**, **age**, **address**). Styling the table is *optional*. Semantic HTML is required. Examples:

|  |  |
| --- | --- |
| **Input** | **Output** |
| Gosho  0882-321-423  24  Hadji Dimitar |  |
| Pesho  0884-888-888  67  Suhata Reka |  |

## Form Data

Write a PHP script **GetFormData.php** which retrieves the **input data from an HTML form**, and displays it as string. The input fields should hold **name**, **age** and **gender** (radio buttons). The returned string should be a message containing the information submitted by the form. 100% accuracy is *NOT* required. Semantic HTML is required. Example:

|  |  |
| --- | --- |
| **Input** | **Output** |
|  |  |

## \* Time Until New Year

Write a PHP script **TimeUntilNewYear.php.** Use the built-in function **getdate()** to get the current date and time. Print how many **hours, minutes and seconds** are left until New Year and how many **days, hours, minutes and seconds** are leftin a **counter format** . Look at examples below to get a better idea. The **examples** show the current date and time in **"d-m-Y G:i:s"** format. Use **the current time**. Check [here](http://bg2.php.net/manual/en/function.date.php) for date/time formats in PHP. (Note: Keep [the Spring/Autumn time shifts](http://en.wikipedia.org/wiki/Daylight_saving_time" \l "Procedure) in mind in your calculations.)

|  |  |
| --- | --- |
| **Examples** | **Output** |
| 12-08-2014 13:07:09 | Hours until new year : 3394  Minutes until new year : 203 692  Seconds until new year : 12 221 570  Days:Hours:Minutes:Seconds 141:10:52:50 |
| 12-08-2014 11:08:47 | Hours until new year : 3396  Minutes until new year : 203 811  Seconds until new year : 12 228 672  Days:Hours:Minutes:Seconds 141:12:51:12 |

## \*\* Awesome Calendar

Write a PHP script **AwesomeCalendar.php** which creates a calendar in HTML **for a given year**. Styling the calendar is *optional*. Semantic HTML is required. Hint: Embed HTML in your PHP code. Use *tables* or *divs* for structuring the calendar. Look for a way to find the day of the week.

