{EPITECH}

WEB DEV SEMINAR

DAY 08 - PHP

WEB DEV SEMINAR

Contents

You will now be introduced to PHP. A language mainly used for web back-end.

This day is tested by the autograder!

✓ Contents
- Task 01
- Task 02
- Task 03
- Task 04
- Task 05
- Task 06

Task 01

Delivery: `./task01.php`
Prototype: `discover_type(int $age, string $name, float $gpa, bool $isStudent)`

Create a `discover_type` function that:

- ✓ complies with the provided prototype ;
- ✓ and prints:
  - when `isStudent` is true:
    - `Hello my name is $name, I'm $age years old. I'm a student at Epitech with a GPA of $gpa.`
  - otherwise:
    - `Hello my name is $name, I'm $age years old. I'm not a student.`

Examples:

Terminal

```
~/T-WEB-500> cat index.php
<?php
require(“task01.php”);
discover_type(21, “Maria”, 3.53, true);
discover_type(52, “Jules”, 0, false);
?>
```

Terminal

```
~/T-WEB-500> php index.php
Hello my name is Maria, I’m 21 years old. I’m a student at Epitech with a GPA of 3.53.
Hello my name is Jules, I’m 52 years old. I’m not a student.
```

Task 02

Delivery: `./task02.php`

Create a `dog_bark` that:

- ✓ takes an integer `woof_nb` as parameter ;
- ✓ displays `woof_nb` times a “woof” ;
- ✓ separates each “woof” by a single space ;
- ✓ ends the sentence made of “woof” with a newline.

Examples:

Terminal

```
~/T-WEB-500> cat index.php
<?php
require(‘‘task02.php’’);
dog_bark(1);
dog_bark(-42);
dog_bark(3);
?>
```

Terminal

```
~/T-WEB-500> php index.php | cat -e
woof$
$
woof woof woof$
```

Task 03

Delivery: `./task03.php`

Create a `get_shortest` function that:

- ✓ takes an array of strings as parameter ;
- ✓ returns the shortest string from the array when unique ;
- ✓ returns the first shortest string from the array when multiple.

Examples:

```
Terminal
~/T-WEB-500> cat index.php
<?php
require(''task03.php'');
var_dump(get_shortest(array(''Time'', ''flowing'', ''like'', ''a'', ''river'')));
var_dump(get_shortest(array(''Run'', ''like'', ''hell'')));
var_dump(get_shortest(array(''chou'', ''be'', ''do'')));
?>
```

```
Terminal
~/T-WEB-500> php index.php
string(1) ''a''
string(3) ''Run''
string(2) ''be''
```

Task 04

Delivery: `./task04.php`
Prototype: `calc(string $operator, int $nbr1, int $nbr2)`

Create a `calc` function that:

- ✓ complies with the provided prototype ;
- ✓ allows `+`, `-`, `*`, `/` and `%` as valid operators ;
- ✓ returns the result of `$nbr1 $operator $nbr2` ;
- ✓ returns “Unknown operator” if an invalid operator is given ;
- ✓ returns “Cannot divide by 0” when necessary.

Examples:

Terminal

```
~/T-WEB-500> cat index.php
<?php
require(“task04.php”);
var_dump(calc(“+”, 1, 3));
var_dump(calc(“*”, 7, 6));
var_dump(calc(“/”, 4, 3));
var_dump(calc(“+/”, 4, 2));
var_dump(calc(“/”, 4, 0));
?>
```

Terminal

```
~/T-WEB-500> php index.php
int(4)
int(42)
float(1.3333333333333)
string(16) “Unknown operator”
string (18) “Cannot divide by 0”
```

Task 05

Delivery: `./task05.php`

Write a `calc_average` function that:

- ✓ takes an array of numbers as parameter ;
- ✓ calculate the average of those numbers ;
- ✓ return the average value rounded to the 1/10th.

Examples:

Terminal

```
~/T-WEB-500> cat index.php
<?php
require(“task05.php”);
var_dump(calc_average(array(1, 4, 28, 12, 3, 4, 4, 27)));
?>
```

Terminal

```
~/T-WEB-500> php index.php
float(10.4)
```

Task 06

Delivery: `./task06.php`

Write a function that outputs the following sequence to the n^{th} iteration.

Your function should not print anything if you pass it a negative number.

Here is the beginning of the sequence:
1
11
21
1211
111221
312211

The first iteration is 0.

Examples:

```
Terminal
~/T-WEB-500> cat index.php
<?php
require(“task06.php”);
echo “Sequence 0:” . PHP_EOL;
sequence(0);
echo “Sequence 3:” . PHP_EOL;
sequence(3);
?>
```

```
Terminal
~/T-WEB-500> php index.php | cat -e
Sequence 0:$
1$
Sequence 3:$
1$
11$
21$
1211$
```

v 4.3