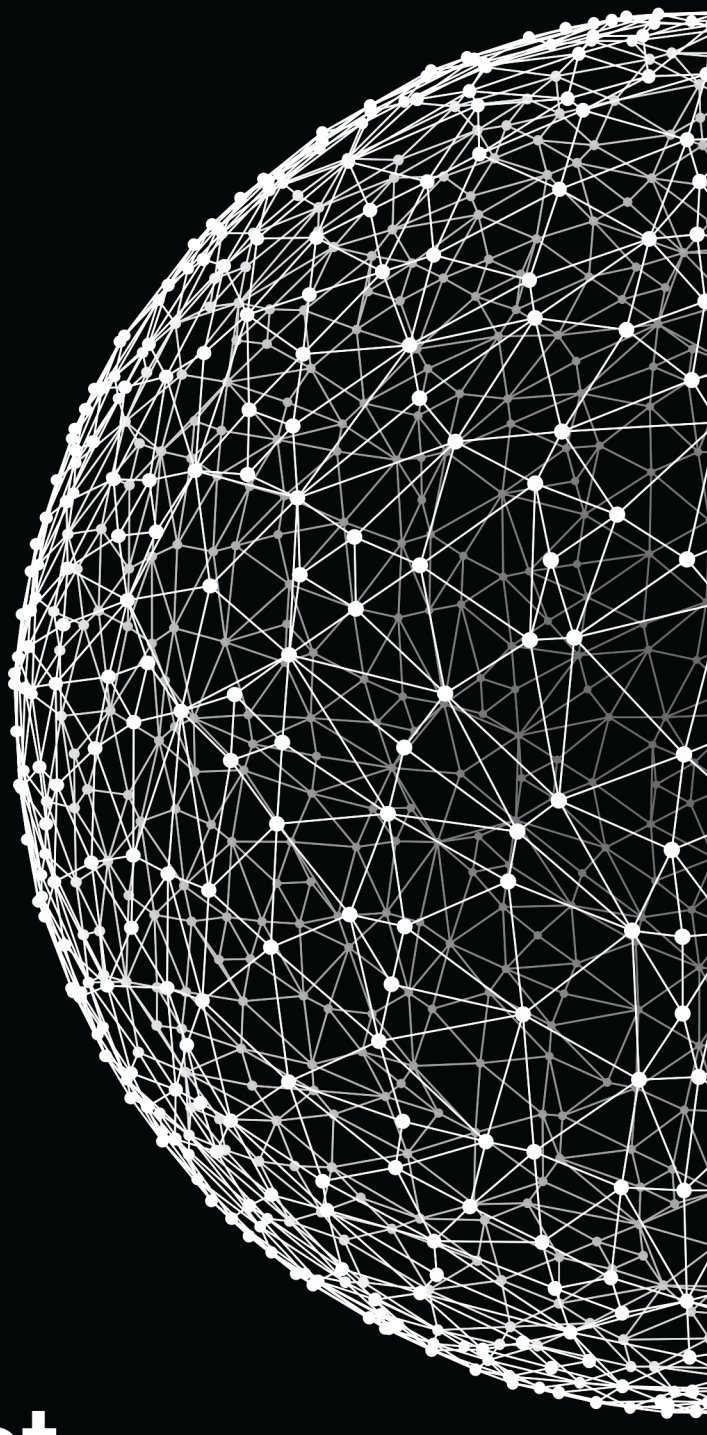
Sprint 05

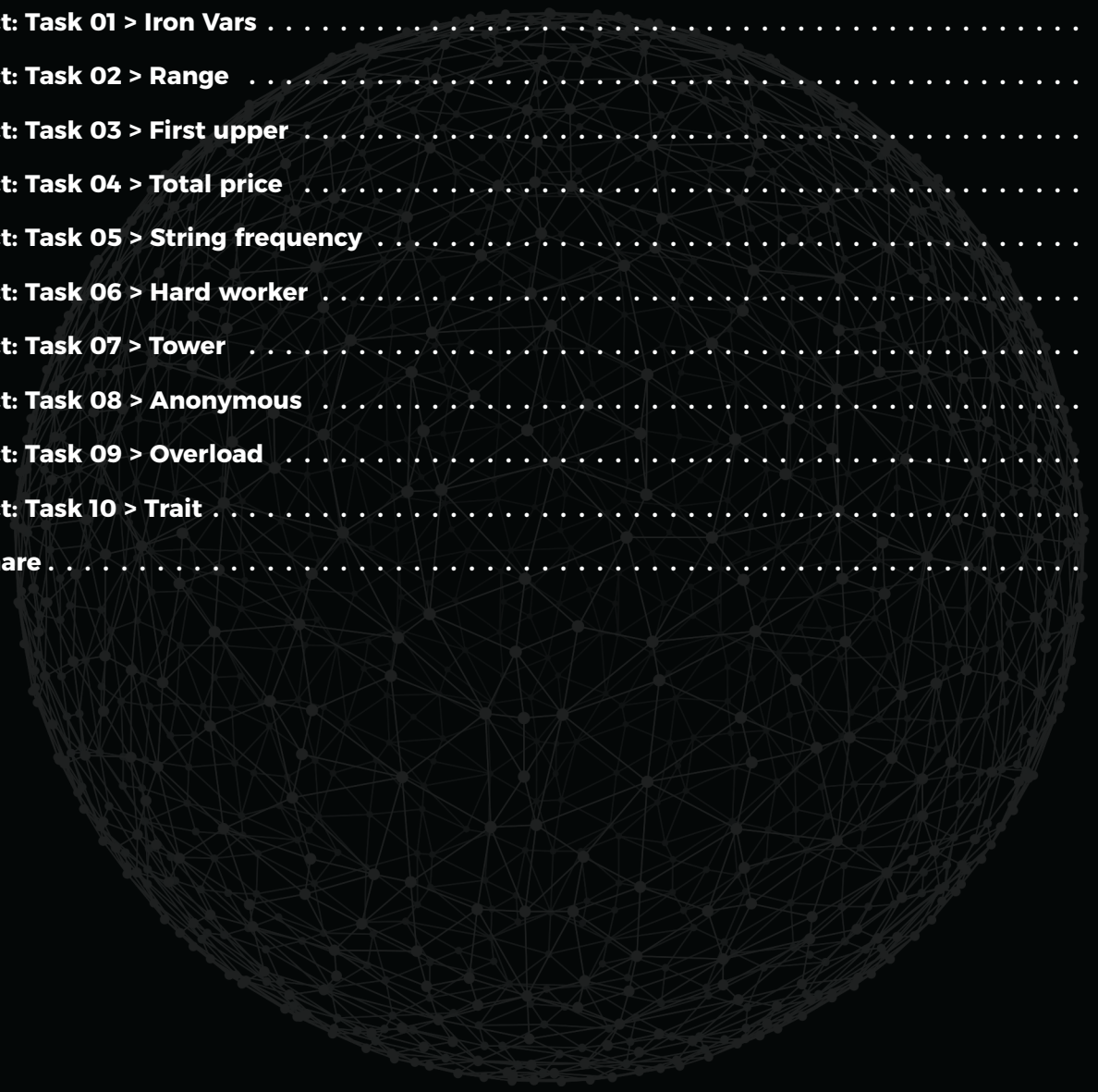
Half Marathon Web

March 25, 2021



ucode connect

Contents



ucode connect

Engage

DESCRIPTION

Hello again! Get ready to meet PHP - the most common language of backend development.

PHP is a server language (backend). So far, you've only worked with frontend. A web service is like a restaurant: frontend is the dining area, and backend is the kitchen.

To get started, an environment for local development should be prepared. Without a configured local server, PHP scripts can't be executed. You can configure it by yourself if you are an advanced user and have heard about Apache and Nginx. Otherwise, we recommend to download `MAMP` and not to get to stuck on the stage of web configuring. It's only a recommendation and you are free to use any tool for local server deployment you want.

Today, you will get used to the language syntax of PHP.

New day, new language!

BIG IDEA

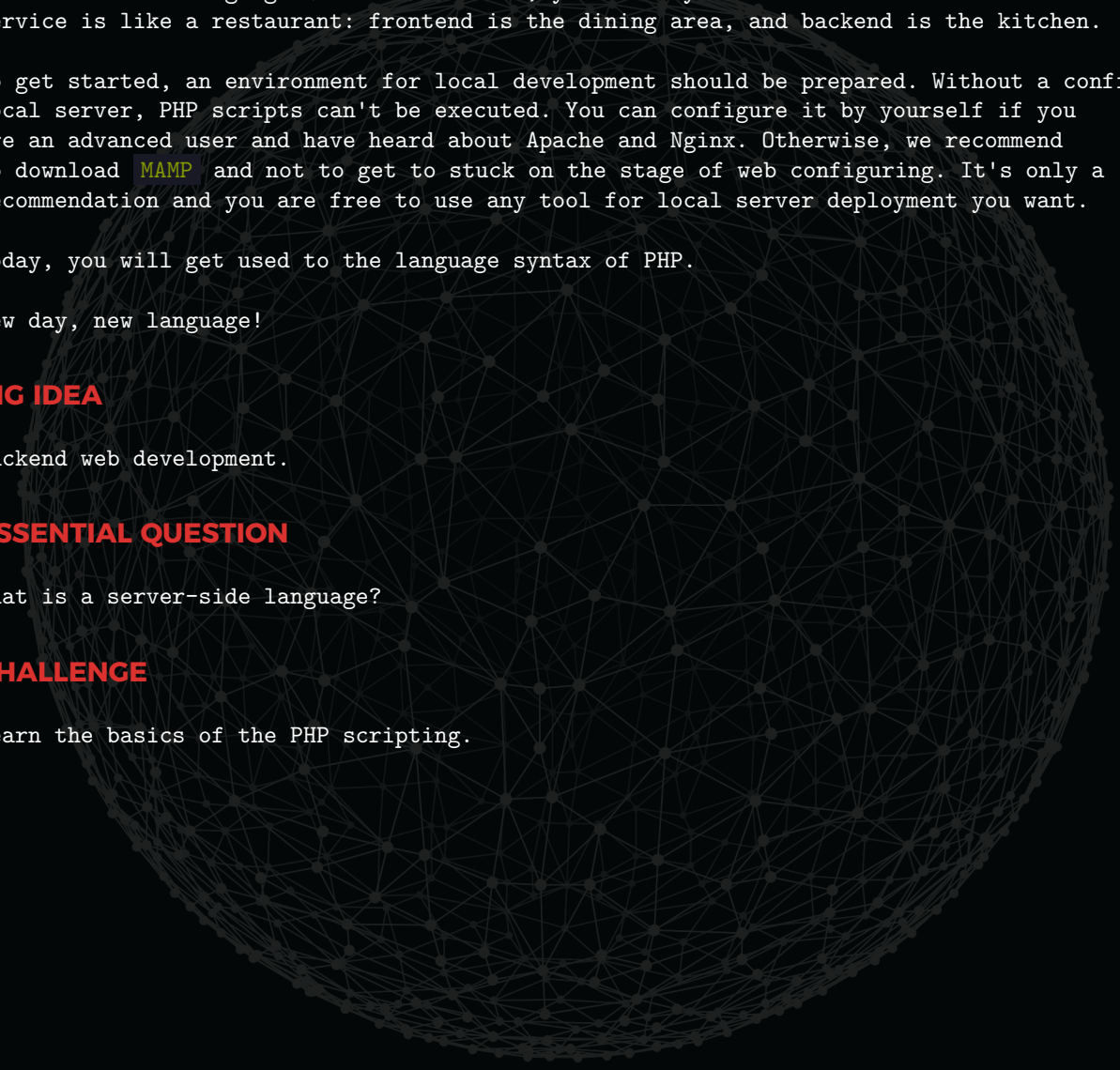
Backend web development.

ESSENTIAL QUESTION

What is a server-side language?

CHALLENGE

Learn the basics of the PHP scripting.



Investigate

GUIDING QUESTIONS

We invite you to find answers to the following questions. By researching and answering them, you will gain the knowledge necessary to complete the challenge. To find answers, ask the students around you and search the internet. We encourage you to ask as many questions as possible. Note down your findings and discuss them with your peers.

- What are PHP tags? Why are they useful?
- Which data types exist in PHP?
- Which elements may a class contain?
- Which naming rules does PHP have?
- How are class constants different from normal variables in PHP?
- How to access non-static and static properties within class methods?
- How to set a scope of visibility for a class element?
- For what is a pseudo-variable `this` used in PHP?
- When do you need to use a scope resolution operator?
- What is the keyword `static` used for?
- What is a `trait` in PHP?
- What is an anonymous class? In what cases is it used?
- What is an overload in PHP?

GUIDING ACTIVITIES

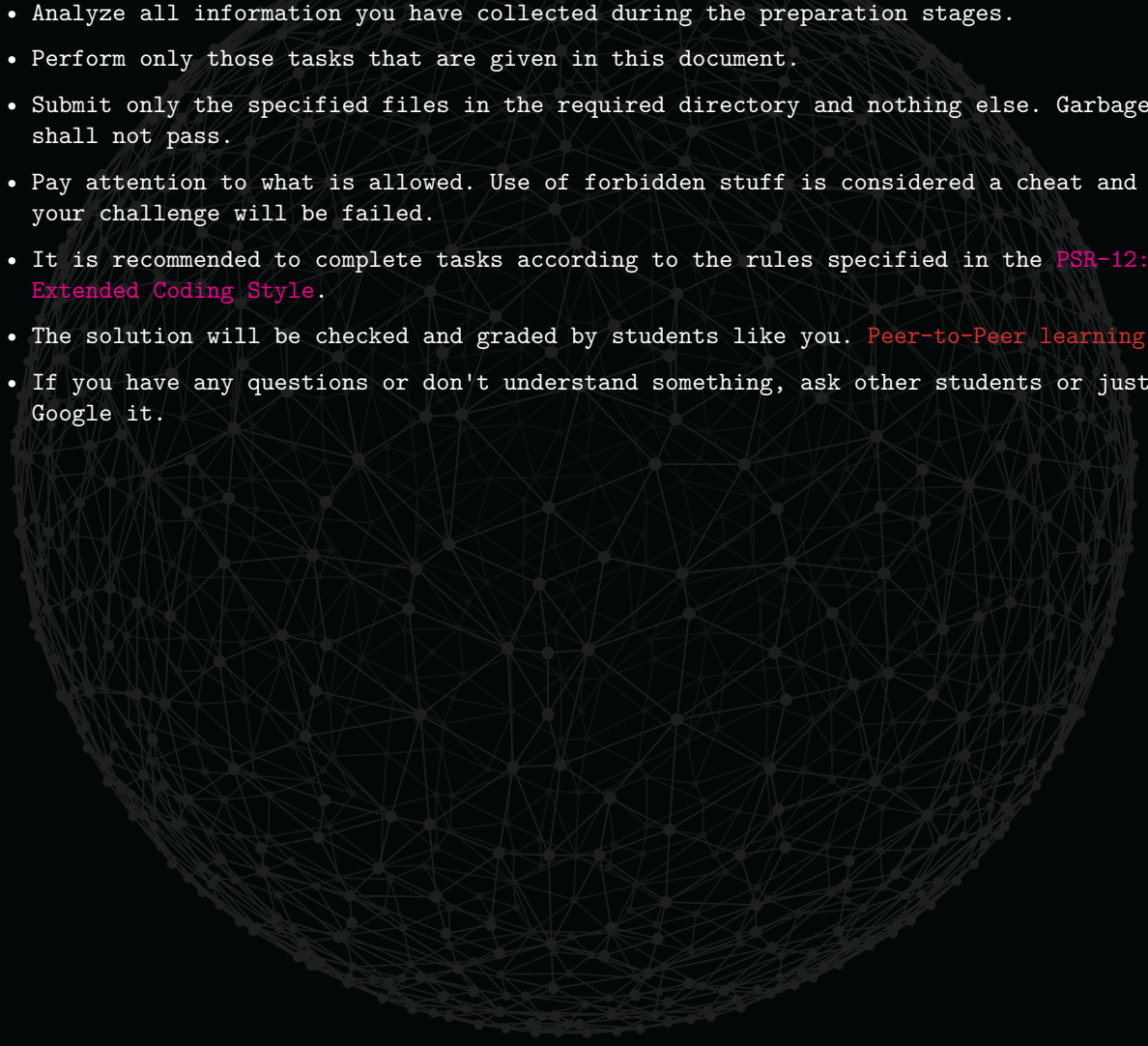
Complete the following activities. Don't forget that you have a limited time to overcome the challenge. Use it wisely. Distribute tasks correctly.

- Watch this video.
- Prepare the environment for local development: install an interpreter and a web server and configure it all to work together.
- Read about the advantages and disadvantages of PHP among other server languages.
- Find the most informative resources with PHP documentation.
  We advise you to visit php.net.
- Write your first PHP script which will output "Hello PHP world!" to the console.
- Read about PHP debugging.
- Clone your git repository issued on the challenge page in the LMS.
- Proceed with tasks.

ANALYSIS

Analyze your findings. What conclusions have you made after completing guiding questions and activities? In addition to your thoughts and conclusions, here are some more analysis results.

- Be attentive to all statements of the story. Examine the given examples carefully. They may contain details that are not mentioned in the task.
- Analyze all information you have collected during the preparation stages.
- Perform only those tasks that are given in this document.
- Submit only the specified files in the required directory and nothing else. Garbage shall not pass.
- Pay attention to what is allowed. Use of forbidden stuff is considered a cheat and your challenge will be failed.
- It is recommended to complete tasks according to the rules specified in the PSR-12: Extended Coding Style.
- The solution will be checked and graded by students like you. Peer-to-Peer learning.
- If you have any questions or don't understand something, ask other students or just Google it.



Act: Task 00

NAME

Good morning!

DIRECTORY

`t00/`

SUBMIT

`index.php`

ALLOWED

PHP

DESCRIPTION

Set up the environment for running PHP scripts.

Create a PHP script that outputs a message. See the CONSOLE OUTPUT for an illustration of how your script must work.

CONSOLE OUTPUT

```
>php t00/index.php | cat -e
Good morning!$
It's 7 A.M.$
The weather in Malibu is 72 degrees with scattered clouds.$
The surf conditions are fair, high tide will be at 10:52 a.m.$
```

SEE ALSO

PHP

Act: Task 01

NAME

Iron Vars

DIRECTORY

`t01/`

SUBMIT

`index.php`

ALLOWED

PHP

LEGEND

138. 138 combat missions.
That's how many I've flown, Tony.
Every one of them could've been my last, but I flew 'em.
'Cause the fight needed to be fought.

DESCRIPTION

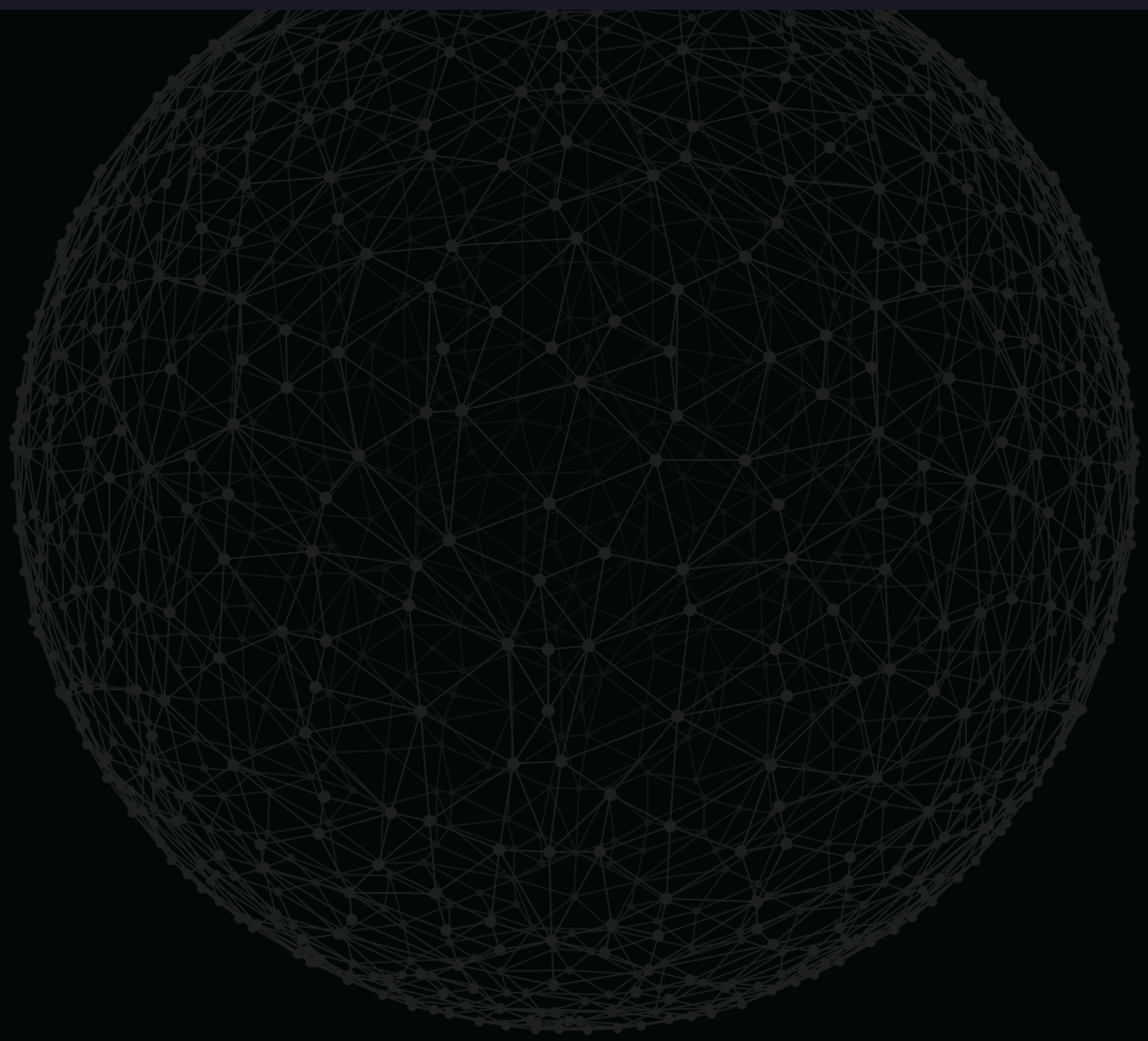
Create a script with declared and initialized variables of the available scalar types.
The script from the SYNOPSIS must produce output as shown in the CONSOLE OUTPUT section.

SYNOPSIS

```php
<?php

/*
  Task 01 (test.php)
  Task name: Iron Vars
*/

include 'index.php';

echo '$iron_man     is ' . gettype($iron_man) .' '. $iron_man . "\n";
echo '$war_machine  is ' . gettype($war_machine) .' '. $war_machine . "\n";
echo '$var0         is ' . gettype($var0) .' '. $var0 ."\n";
echo '$var1         is ' . gettype($var1) .' '. $var1 . "\n";
echo '$var2         is ' . gettype($var2) .' '. $var2 . "\n";
echo '$var3         is ' . gettype($var3) .' '. $var3 . "\n";
```

CONSOLE OUTPUT

```
>php tests/t01/test.php | cat -e
$iron_man     is string Anthony Edward "Tony" Stark$
$war_machine  is string Colonel James Rupert "Rhodey" Rhodes$
$var0         is boolean 1$
$var1         is string string$
$var2         is double 12.5$
$var3         is integer 1$
```



Act: Task 02

NAME

Range

DIRECTORY

`t02/`

SUBMIT

`index.php`

ALLOWED

PHP

DESCRIPTION

Create a function that takes two numbers as parameters. These two numbers are the start and end of an inclusive range. The function prints information for all the numbers in that range to the console.

Information includes whether a number is

- divisible by 2
- divisible by 3
- divisible by 10

The default function range is `1 - 60`.

The script from the SYNOPSIS must produce output as shown in the CONSOLE OUTPUT section.

SYNOPSIS

```php
<?php

/*
  Task 02 (test.php)
  Task name: Range
*/

include 'index.php';

echo "*** Range is 3 - 7 checkDivision(3,7) ***\n";
checkDivision(3,7);
```

```php
echo "\n*** Range is 58 - ... checkDivision(58) ***\n";
checkDivision(58);

echo "\n*** Range is ... - ... checkDivision() ***\n";
checkDivision();
```

CONSOLE OUTPUT

```
>php t02/test.php | cat -e
*** Range is 3 - 7 checkDivision(3,7) ***$
The number 3 is divisible by 3$
The number 4 is divisible by 2$
The number 5 -$
The number 6 is divisible by 2, is divisible by 3$
The number 7 -$
$
*** Range is 58 - ... checkDivision(58) ***$
The number 58 is divisible by 2$
The number 59 -$
The number 60 is divisible by 2, is divisible by 3, is divisible by 10$
$
*** Range is ... - ... checkDivision() ***$
The number 1 -$
The number 2 is divisible by 2$
The number 3 is divisible by 3$
The number 4 is divisible by 2$
The number 5 -$
...
The number 60 is divisible by 2, is divisible by 3, is divisible by 10$
```

Act: Task 03

NAME

First upper

DIRECTORY

`t03/`

SUBMIT

`index.php`

ALLOWED

PHP

DESCRIPTION

Create a function `firstUpper(str)` that:

- takes a string as input
- returns that string with only the first letter capitalized, or an empty string

Remove all whitespaces at the beginning and at the end of the string.

The script from the SYNOPSIS must produce output as shown in the CONSOLE OUTPUT section.

SYNOPSIS

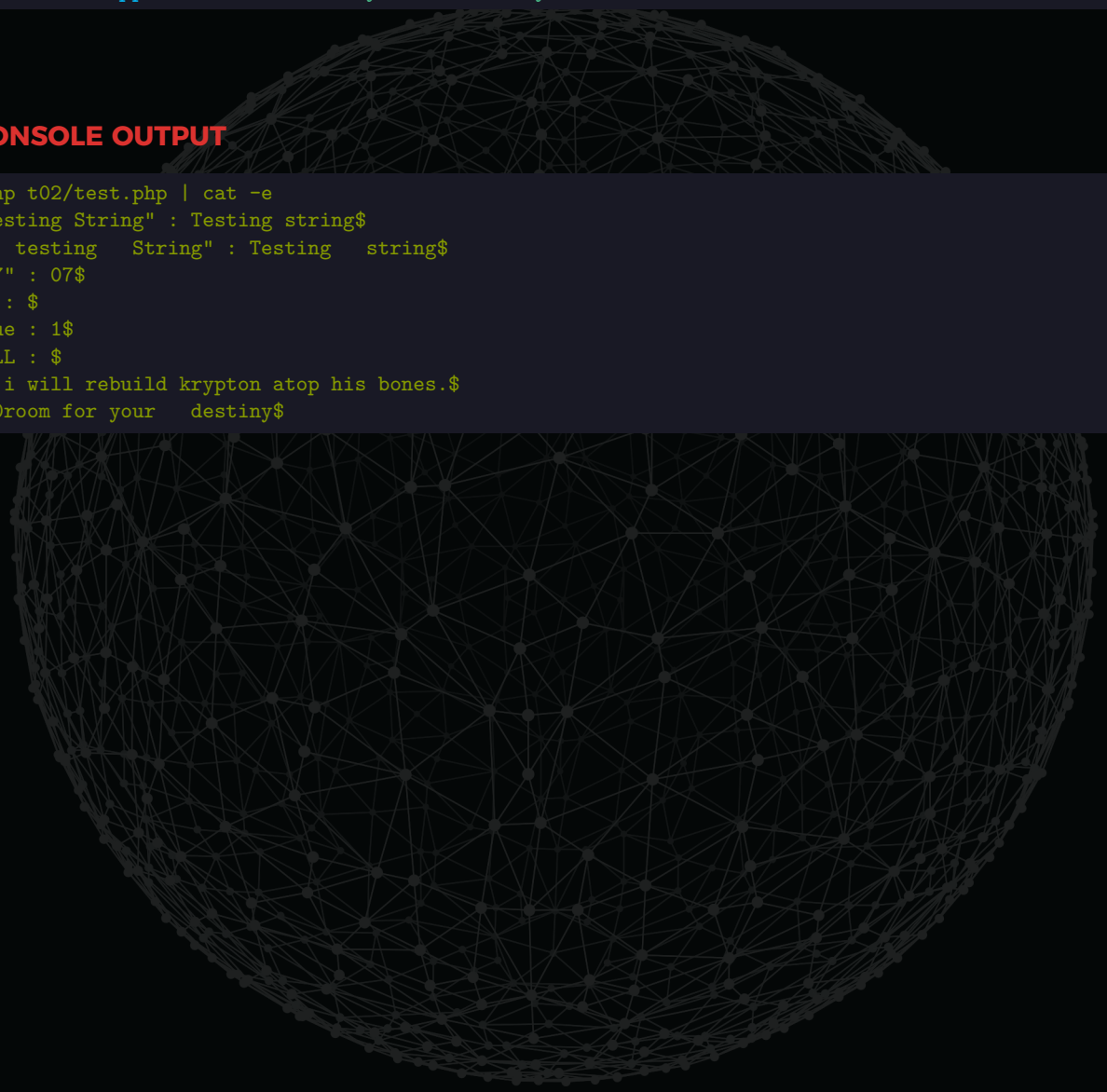
```
firstUpper (string) : string
```

```php
<?php

/*
  Task 03 (test.php)
  Task name: firstUpper
*/

include 'index.php';

echo ('"testing String" : ' . firstUpper("testing String")) . "\n";
echo ('"   testing   String" : ' . firstUpper("   testing   String")) . "\n";
echo ('"07" : ' . firstUpper("07")) . "\n";
echo ('"" : ' . firstUpper("")) . "\n";
```

```php
echo ('true : ' . firstUpper(true)) . "\n";
echo ('NULL : ' . firstUpper(NULL)) . "\n";

echo(firstUpper("   ...I Will Rebuild Krypton Atop His Bones.")) . "\n";
echo(firstUpper(" 300room FOR yoUr   DESTiny   ")) . "\n";
```

CONSOLE OUTPUT

```
>php t02/test.php | cat -e
"testing String" : Testing string$
"   testing   String" : Testing   string$
"07" : 07$
"" : $
true : 1$
NULL : $
...i will rebuild krypton atop his bones.$
300room for your   destiny$
```



Act: Task 04

NAME

Total price

DIRECTORY

`t04/`

SUBMIT

`index.php`

ALLOWED

PHP

DESCRIPTION

Create a script with a function that sums up the total price of an order. This task is similar to Task 05 in Sprint 02, but this time you have to do it with PHP istead of JS.

The script must contain a function that:

- takes three parameters:
  - number of items
  - price per item
  - current total
- returns the total order price

```
total(addCount, addPrice, currentTotal = 0)
```

The script from the SYNOPSIS must produce output as shown in the CONSOLE OUTPUT section.

SYNOPSIS

```
total(float, float, float) : float
```

```php
<?php

/*
  Task 04 (test.php)
  Task name: Total price
*/

include 'index.php';

$basket_total = total(1, 0.1);
$basket_total = total(1, 0.2,  $basket_total);

echo "\nPrice of order is $basket_total\n";

$basket_total = total(3, 1.4,  $basket_total);

echo "\nPrice of order is $basket_total\n";
```

CONSOLE OUTPUT

```
>php t02/test.php | cat -e
$
Price of order is 0.3$
$
Price of order is 4.5$
```

Act: Task 05

NAME

String frequency

DIRECTORY

`t05/`

SUBMIT

`index.php`

ALLOWED

PHP

DESCRIPTION

Create a class `StrFrequency` that:

- initializes the value of the string using the constructor
- has `letterFrequencies()` method that counts the frequency of each letter in the string
- has `wordFrequencies()` method that counts the frequency of each word in the string
- has `reverseString()` method that inverts the order of letters in the string

Ignore the letter case in all operations. Class methods must work only with letter characters.

The script from the SYNOPSIS must produce output as shown in the CONSOLE OUTPUT section.

SYNOPSIS

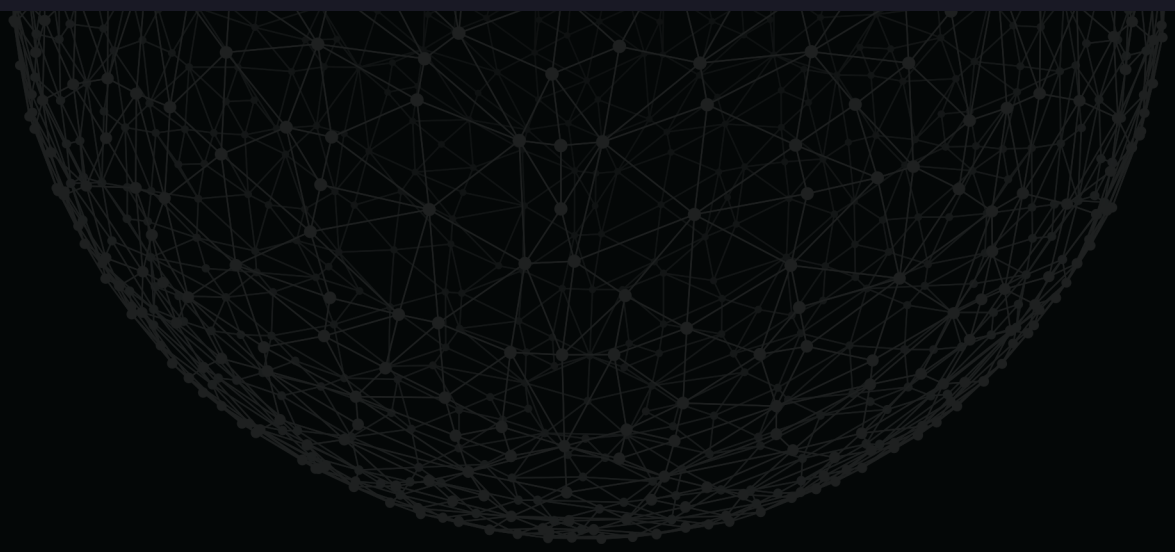
```php
<?php

/*
  Task 05 (example.php)
  Task name: String frequency
*/

include 'index.php';

function test($string)
{
  $obj = new StrFrequency($string);
  $symbol = $obj->letterFrequencies();
```

```php
  echo "Letters in " . $string . "\n";
  foreach ($symbol as $k => $v) {
      echo "Letter ". $k . " is repeated " . $v . " times\n";
  }

  $symbol = $obj->wordFrequencies();
  echo "Words in " . $string . "\n";
  foreach ($symbol as $k => $v) {
      echo "Word ". $k . " is repeated " . $v . " times\n";
  }

  echo "Reverse the string: " . $string . "\n";
  echo $obj->reverseString() . "\n";
}

test("Face it, Harley-- you and your Puddin' are kaput!");
echo "**************\n";
test("  Test test 123 45 !0 f   HeLlO wOrLd  ");
echo "**************\n";
test("");
```

CONSOLE OUTPUT

```
>php example.php | cat -e
Letters in Face it, Harley-- you and your Puddin' are kaput!$
Letter A is repeated 5 times$
Letter C is repeated 1 times$
Letter D is repeated 3 times$
Letter E is repeated 3 times$
Letter F is repeated 1 times$
Letter H is repeated 1 times$
Letter I is repeated 2 times$
Letter K is repeated 1 times$
Letter L is repeated 1 times$
Letter N is repeated 2 times$
Letter O is repeated 2 times$
Letter P is repeated 2 times$
Letter R is repeated 3 times$
Letter T is repeated 2 times$
Letter U is repeated 4 times$
Letter Y is repeated 3 times$
Words in Face it, Harley-- you and your Puddin' are kaput!$
Word FACE is repeated 1 times$
Word IT is repeated 1 times$
Word HARLEY is repeated 1 times$
Word YOU is repeated 1 times$
Word AND is repeated 1 times$
Word YOUR is repeated 1 times$
Word PUDDIN is repeated 1 times$
Word ARE is repeated 1 times$
```

```
Word KAPUT is repeated 1 times$
Reverse the string: Face it, Harley-- you and your Puddin' are kaput!$
 tupak era  nidduP ruoy dna uoy   yelraH  ti ecaF$
**********$
Letters in   Test test 123 45 !0 f   HeLlO wOrLd  $
Letter D is repeated 1 times$
Letter E is repeated 3 times$
Letter F is repeated 1 times$
Letter H is repeated 1 times$
Letter L is repeated 3 times$
Letter O is repeated 2 times$
Letter R is repeated 1 times$
Letter S is repeated 2 times$
Letter T is repeated 4 times$
Letter W is repeated 1 times$
Words in   Test test 123 45 !0 f   HeLlO wOrLd  $
Word TEST is repeated 2 times$
Word F is repeated 1 times$
Word HELLO is repeated 1 times$
Word WORLD is repeated 1 times$
Reverse the string:   Test test 123 45 !0 f   HeLlO wOrLd  $
  dLrOw OlLeH   f           tset tseT  $
**********$
Letters in $
Words in $
Reverse the string: $
$
```



Act: Task 06

NAME

Hard worker

DIRECTORY

`t06/`

SUBMIT

`HardWorker.php`

ALLOWED

PHP

LEGEND

I can't have this, any of this. There is no place on Earth I can go where I'm not a monster.

DESCRIPTION

Create a class `HardWorker` with private properties: name, age, salary.
Implement the following public methods for this class:

- `setName`
- `getName`
- `setAge` (from 1 to 100)
- `getAge`
- `setSalary` (from 100$ to 10000$)
- `getSalary`
- `toArray` - returns an array with all properties

The methods `setAge` and `setSalary` return `true` if the input is valid and the property has been updated, and `false` otherwise.

The script from the SYNOPSIS must produce output as shown in the CONSOLE OUTPUT section.

SYNOPSIS

```php
<?php

/*
  Task 06 (test.php)
  Task name: HardWorker
*/

include 'HardWorker.php';

$Bruce = new HardWorker();

$Bruce->setName("Bruce");

echo $Bruce->getName() . "\n";

$Bruce->setAge(50);
$Bruce->setSalary(1500);

print_r ($Bruce->toArray());

$Bruce->setName("Linda");
$Bruce->setAge(140);

print_r ($Bruce->toArray());
```

CONSOLE OUTPUT

```
>php example.php | cat -e
Bruce$
Array$
($
    [name] => Bruce$
    [age] => 50$
    [salary] => 1500$
)$
Array$
($
    [name] => Linda$
    [age] => 50$
    [salary] => 1500$
)$
```

Act: Task 07

NAME

Tower

DIRECTORY

`t07/`

SUBMIT

`Tower.php`

ALLOWED

PHP

DESCRIPTION

Create a `Tower` class that inherits from the `Building` class (see in resources).

Add the following private properties and their public getters and setters:

- `elevator (bool)`
  - getter - `hasElevator()`
  - setter - `setElevator()`
- `arc_capacity (int)`
  - getter - `getArcCapacity()`
  - setter - `setArcCapacity()`
- `height (float)`
  - getter - `getHeight()`
  - setter - `setHeight()`

Also, add the method `public function getFloorHeight(): float` which returns height / floors. Don't forget to update parent `toString` method.

Also, you can add other useful properties and methods.

The script from the SYNOPSIS must produce output as shown in the CONSOLE OUTPUT section.

SYNOPSIS

```php
<?php

/*
  Task 07 (test.php)
  Task name: Tower
*/

require_once(__DIR__ . "/Building.php");
require_once(__DIR__ . "/Tower.php");

$StarkTower = new Tower(93, "Different", "Manhattan, NY");

$StarkTower->setElevator(true);
$StarkTower->setArcCapacity(70);
$StarkTower->setHeight(1130);
echo $StarkTower->toString();
```

CONSOLE OUTPUT

```
>php t07/test.php | cat -e
Floors : 93$
Material : Different$
Address : Manhattan, NY$
Elevator: +$
Arc reactor capacity : 70$
Height : 1130$
Floor height : 12.150537634409$
```

Act: Task 08

NAME

Anonymous

DIRECTORY

`t08/`

SUBMIT

`Anonymous.php`

ALLOWED

PHP

LEGEND

"Some people call me a terrorist. I consider myself a teacher."

DESCRIPTION

Create a function that returns an instance of an anonymous class with private fields and methods to access them.

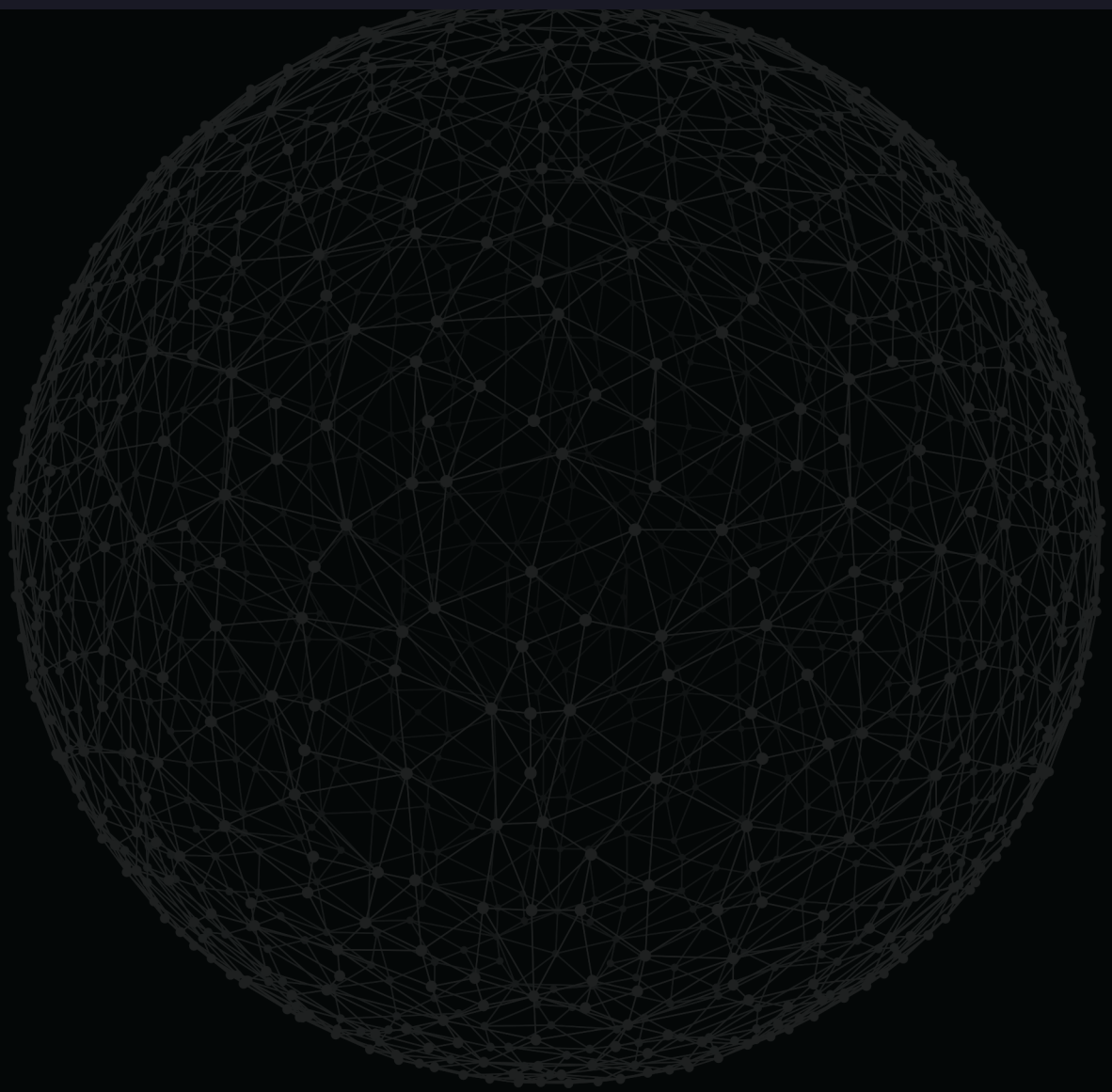
The script from the SYNOPSIS must produce output as shown in the CONSOLE OUTPUT section.

SYNOPSIS

```php
<?php

/*
  Task 08 (test.php)
  Task name: Anonymous
*/

require_once(__DIR__ . "/Anonymous.php");

$mandarin = get_anonymous("Unknown", "Mandarin", "Ten Rings");

print(implode("\n", array("name" => $mandarin->getName(),
    "alias" => $mandarin->getAlias(), "affiliation" => $mandarin->getAffiliation())));
```

CONSOLE OUTPUT

```
>php t08/test.php
Unknown
Mandarin
Ten Rings%
```



Act: Task 09

NAME

Overload

DIRECTORY

`t09/`

SUBMIT

`Overload.php`

ALLOWED

PHP

LEGEND

Things are different now. I have to protect the one thing that I can't live without.
That's you.

DESCRIPTION

Create a class `Overload` with the following behavior:

- on writing data to an inaccessible property - create such a property
- on reading data from an inaccessible property - return `"NO DATA"`
- on checking whether a property is set on an inaccessible property - create a property with `"NOT SET"`
- on deleting an inaccessible property - create a property with `NULL`

The script from the SYNOPSIS must produce output as shown in the CONSOLE OUTPUT section.

SYNOPSIS

```php
<?php

/*
  Task 09 (test.php)
  Task name: Overload
*/

require_once(__DIR__ . "/Overload.php");
$overload = new Overload();

$overload->mark_LXXXV = "INACTIVE";
echo $overload->mark_LXXXV;

echo "\n" . $overload->mark_LXXXVI;

if (isset($overload->mark_LXXXVI))
    echo "\n" . $overload->mark_LXXXVI;


unset($overload->mark_IV);
if ($overload->mark_IV == NULL)
    echo "\nNULL\n";
```

CONSOLE OUTPUT

```
>php t09/test.php | cat -e
INACTIVE$
NO DATA$
NOT SET$
NULL$
```

Act: Task 10

NAME

Trait

DIRECTORY

`t10/`

SUBMIT

`MarkII.php, Update.php`

ALLOWED

PHP

LEGEND

"Tony, don't be jealous."
"No, it's subtle, all the bells and whistles."
"Yeah, it's called being a badass."

DESCRIPTION

Create a class `MarkII` and a trait `Update`.

The script from the SYNOPSIS must produce output as shown in the CONSOLE OUTPUT section.

SYNOPSIS

```php
<?php

/*
  Task 10 (test.php)
  Task name: Trait
*/

spl_autoload_register(function ($class_name) {
    include $class_name . '.php';
});

$mark_II = new MarkII();
echo $mark_II->makeBoom() ."\n";

class WarMachine extends MarkII {
```

```
    use Update;
}

$wm = new WarMachine();
print_r($wm->makeBoom());
```

CONSOLE OUTPUT

```
>php t10/test.php | cat -e
2 x Repulsors$
Array$
($
    [0] => 2 x Repulsors$
    [1] => M134 7.62mm Minigun$
    [2] => 2 x FN F2000 Tacticals$
    [3] => Sidewinder "Ex-Wife" Self-Guided Missile$
    [4] => M24 Shotgun$
    [5] => Milkor MGL 40mm Grenade Launcher$
)$
```

Share

PUBLISHING

Last but not least, the final stage of your work is to publish it. This allows you to share your challenges, solutions, and reflections with local and global audiences. During this stage, you will discover ways of getting external evaluation and feedback on your work. As a result, you will get the most out of the challenge, and get a better understanding of both your achievements and missteps.

To share your work, you can create:

- a text post, as a summary of your reflection
- charts, infographics or other ways to visualize your information
- a video, either of your work, or a reflection video
- an audio podcast. Record a story about your experience
- a photo report with a small post

Helpful tools:

- Canva - a good way to visualize your data
- QuickTime - an easy way to capture your screen, record video or audio (macOS)
- ScreenToGif - screen, webcam, and sketchboard recorder with an integrated editor (Windows)

Examples of ways to share your experience:

- Facebook - create and share a post that will inspire your friends
- YouTube - upload an exciting video
- GitHub - share and describe your solution
- Telegraph - create a post that you can easily share on Telegram
- Instagram - share photos and stories from ucode. Don't forget to tag us :)

Share what you've learned and accomplished with your local community and the world. Use #ucode and #CBLWorld on social media.