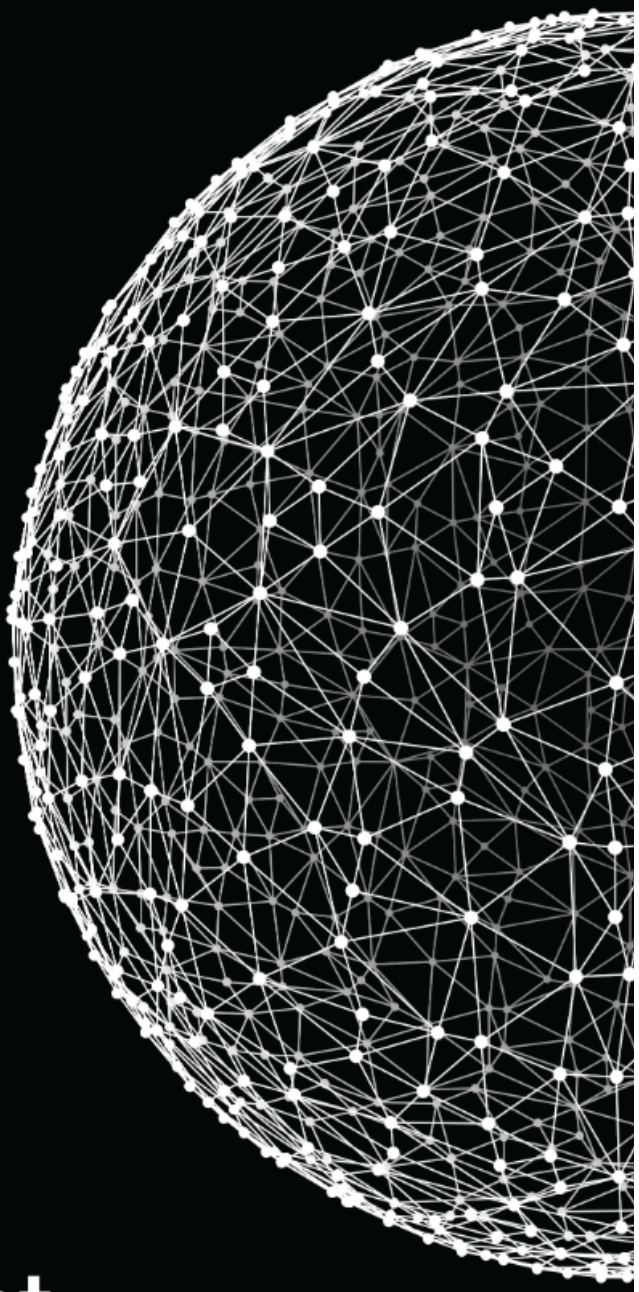
# Sprint 06

## Half Marathon Web

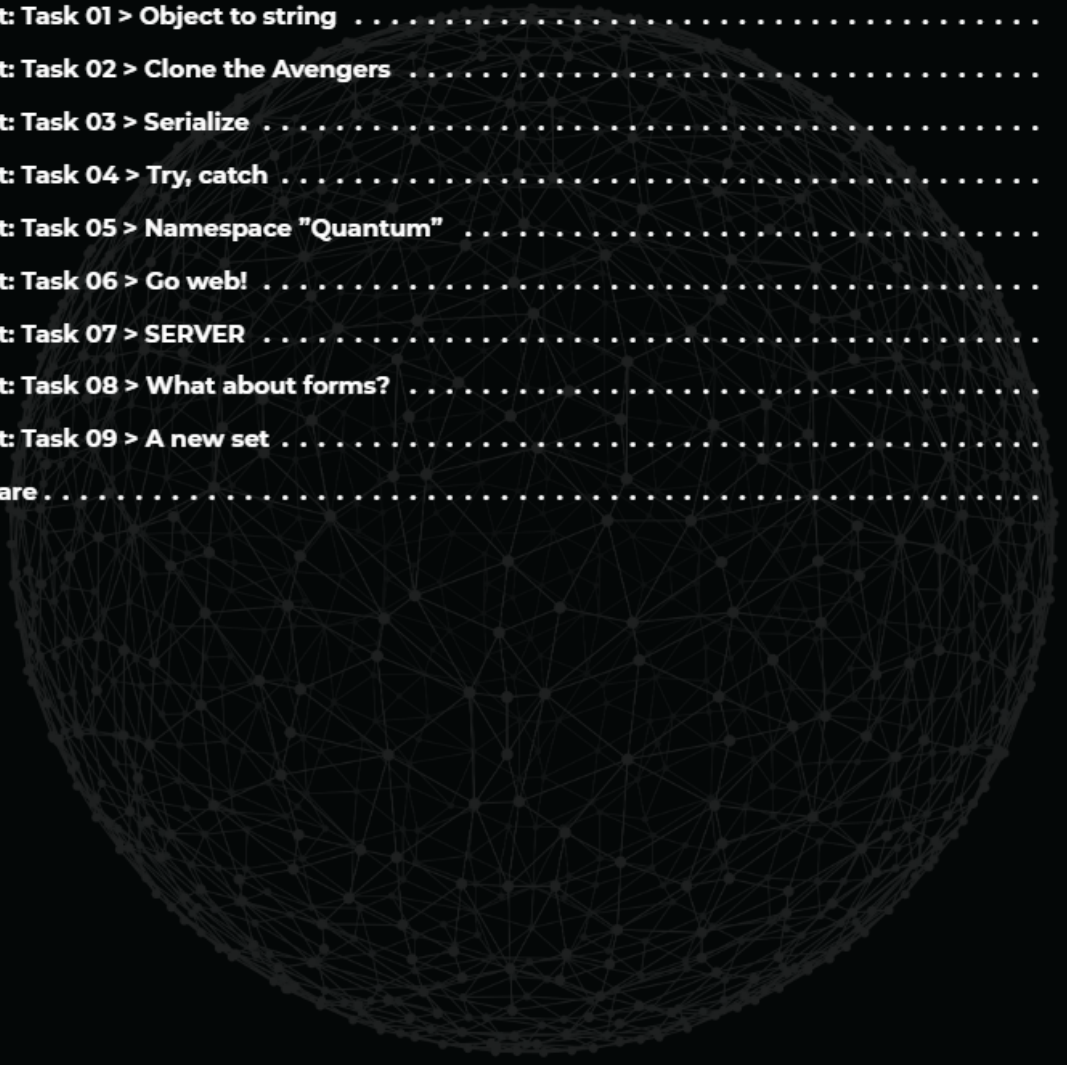
January 13, 2021



ucode connect

# Contents





ucode connect

# Engage

## DESCRIPTION

Hopefully, you've enjoyed PHP so far. Let's continue learning new topics in this Sprint.

Here, you will encounter the concept of namespaces. It's a way to group related classes, interfaces, functions, and constants. Namespaces are one of the methods to encapsulate items. They are present in many languages, not just PHP. Thus, it's crucial to understand how to use the words `namespace` and `use`.

Another key topic is `superglobals array` that helps operate user-entered data. You will also familiarize yourself with HTML forms, which are widely used in web services. And, since user interaction is quite error-prone, in this Sprint you will learn how to handle these errors in the future.

## BIG IDEA

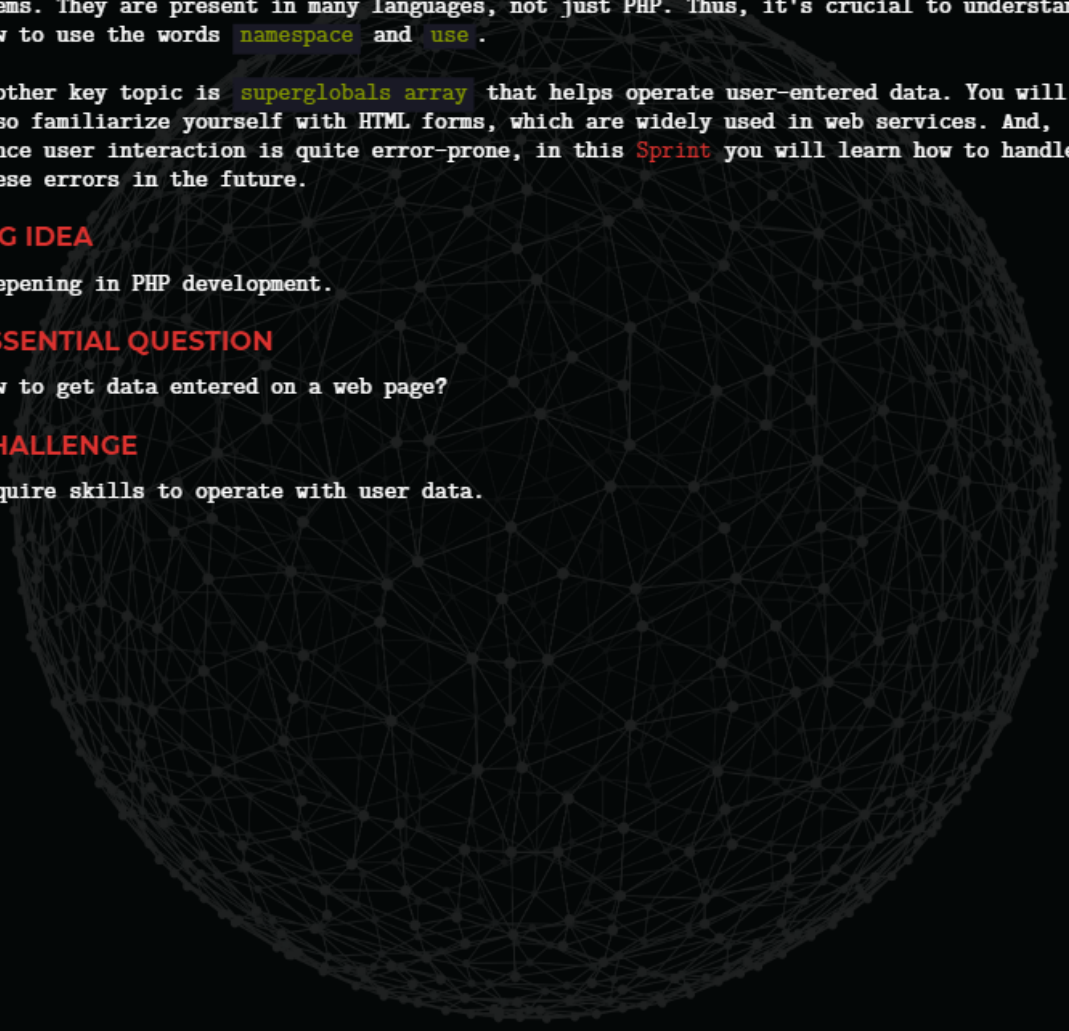
Deepening in PHP development.

## ESSENTIAL QUESTION

How to get data entered on a web page?

## CHALLENGE

Acquire skills to operate with user data.



# 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 `magic methods`? What are they used for?
- Which `magic methods` do you know?
- Explain the purpose of object cloning.
- For what do you need a `serialize` function in PHP?
- What is an exception?
- How to handle exceptions in PHP?
- Which types of errors exist in PHP?
- What are namespaces? What are they used for?
- What do you know about name resolution rules?
- Where are all class and function definitions placed without any namespace definitions?
- What is a `superglobals array` in PHP?
- Which `supeglobals arrays` do you know?
- How can an HTML form and a PHP script interact?

## 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.

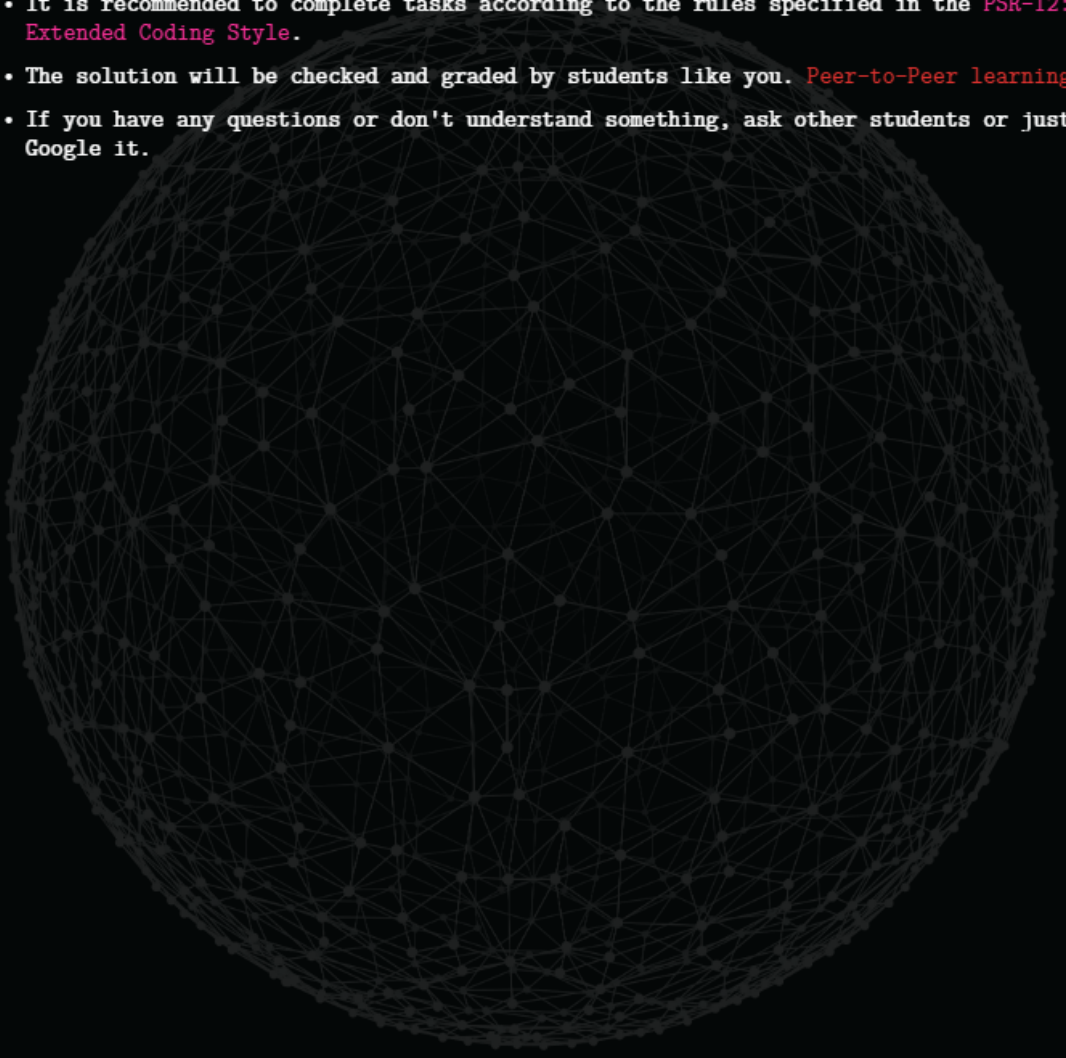
- Research how to structure an HTML form.
- Find on the internet some HTML registration forms templates that you like.
- Discuss with your peers why magic methods are called magic.
- Create your personal cheat sheet with all basic PHP concepts that you have learned so far.
- Clone your git repository, issued on the challenge page in the LMS.
- Try to implement your thoughts in code.

## 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

LinkedList

## DIRECTORY

`t00/`

## SUBMIT

`LLItem.php, LList.php`

## ALLOWED

PHP

## DESCRIPTION

Create a linked list item class called `LLItem`, and a linked list class called `LList`.

Properties of `LLItem` include:

- data
- next - reference to the next element or null

Methods of `LList` (implement `IteratorAggregate`) include:

- `getFirst()`
- `getLast()`
- `add(value)` - creates a new LLItem with given value and appends it to the list
- `addArr(array)` - appends all values (as LLItems) of the given array to the list
- `remove(value)` - deletes the first element that equals the given value
- `removeAll(value)` - deletes all elements that are equal to the given value
- `contains(value)` - checks whether a value is present in the LList
- `clear()` - clears LList
- `count()` - gets number of items in the LList
- `toString()` - prints all element values, separated by a ","
- `getIterator()` - returns an iterator for element values (implement `Iterator`)

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

## SYNOPSIS

```php
<?php

/*
  Task 00 (test.php)
  Task name: LinkedList
*/

function autoload($pClassName)
{
    include(__DIR__ . "/" . $pClassName . ".php");
}
spl_autoload_register("autoload");

$list = new LList();
$list->addArr([100, 1, 2, 3, 100, 4, 5, 100]);

$list->removeAll(100);
$list->add(10);

echo  $list->contains(10) . "\n"; // 1

echo $list->toString() . "\n"; // 1, 2, 3, 4, 5, 10

$sum = 0;
$iter = $list->getIterator();
foreach ($iter as $v)
    $sum += $v;
echo "$sum\n"; // 25
$list->clear();
echo $list->toString() . "\n";
```

## CONSOLE OUTPUT

```
>php t00/test.php | cat -e
1$
1, 2, 3, 4, 5, 10$
25$
$
```

## SEE ALSO

Linked list

# Act: Task 01

## NAME

Object to string

## DIRECTORY

`t01/`

## SUBMIT

`Avenger.php`

## ALLOWED

PHP

## LEGEND

"There was an idea to bring together a group of remarkable people,
to see if we could become something more."

## DESCRIPTION

So, let's create some remarkable people.

Create a class `Avenger`, with the following public properties:

- `name`
- `alias`
- `gender`
- `age`
- `power: array`

In everyday life, the Avengers are normal people without powers. If you introduce an Avenger, their name, gender, and age will be revealed (don't reveal the alias and superpowers!).

Sometimes, an Avenger must be able to show what they can do. In this case, use the Avenger object as a function, and invoke all their powers and show them with the Avenger's alias at the top.

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

## SYNOPSIS

```php
<?php

/*
  Task 01 (test.php)
  Task name: Object to string
*/

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

$first_avenger = new Avenger("Tony Stark", "Iron Man", "man", 38,
                             ["intelligence", "durability", "magnetism"]);
$second_avenger = new Avenger("Natasha Romanoff", "Black Widow", "woman", 35,
                             ["agility", "martial arts"]);

echo "*** calling \$first_avenger() ***\n";
$first_avenger();

echo "*** calling echo \$second_avenger ***\n";
echo $second_avenger;

echo "*** calling \$second_avenger() ***\n";
$second_avenger();
```

## CONSOLE OUTPUT

```
>php t01/test.php | cat -e
*** calling $first_avenger() ***$
IRON MAN$
inteligence$
durability$
magnetism$
$
*** calling echo $second_avenger ***$
name: Natasha Romanoff$
gender: woman$
age: 35$
*** calling $second_avenger() ***$
BLACK WIDOW$
agility$
martial arts$
$
```

# Act: Task 02

## NAME

Clone the Avengers

## DIRECTORY

`t02/`

## SUBMIT

`Team.php`

## ALLOWED

PHP

## LEGEND

"I get emails from a raccoon, so nothing sounds crazy."

## DESCRIPTION

There are so many villains in this universe who want to destroy the Avengers. It will be great to have an additional team or two.

Use the class `Avenger` from the previous task and add an `hp` property to it.

Create a class `Team` with the following properties and methods:

- `id`
- `avengers` – an array of `Avenger` objects
- `battle($damage): void` – method that subtracts `damage` from `hp` of each team member
- `calculate_losses($cloned_team): void` – compares two teams and prints the number of losses in the current team

Create a function `battle($damage)` in which the value of the `hp` variable will be decreased by the given `damage` amount. If the `hp` of an Avenger is 0 or less, then they should be deleted from the Team. In the test code, we clone the team before a battle in order to calculate the losses afterwards. Implement the function `calculate_losses($cloned_team)`, which prints out information about how many Avengers we have lost in the last battle.

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

## SYNOPSIS

```php
<?php

/*
  Task 02 (test.php)
  Task name: Clone the Avengers
*/

require_once(__DIR__ . "/Team.php");
require_once(__DIR__ . "/Avenger.php");

$arr = array();

$arr[0] = new Avenger("Tony Stark", "Iron Man", "man", 38,
                          ["intelligence", "durability", "magnetism"], 120);
$arr[1] = new Avenger("Natasha Romanoff", "Black Widow", "woman", 35,
                          ["agility", "martial arts"], 75);
$arr[2] = new Avenger("Carol Danvers", "Captain Marvel", "woman", 27,
                          ["durability", "flight", "interstellar travel"], 95);

$team = new Team(1, $arr);

echo "Battle 1\n";
$cloned_team = clone $team;
$damage = 25;
$team->battle($damage);
$team->calculate_losses($cloned_team);

echo "\nBattle 2\n";
$cloned_team = clone $team;
$damage = 80;
$team->battle($damage);
$team->calculate_losses($cloned_team);
```

## CONSOLE OUTPUT

```
>php t02/test.php | cat -e
Battle 1$
We haven't lost anyone in this battle!$
$
Battle 2$
In this battle we lost 2 Avenger(s).$
```

# Act: Task 03

## NAME

Serialize

## DIRECTORY

`t03/`

## SUBMIT

`Ant.php`

## ALLOWED

PHP

## LEGEND

"I'm gonna have to shrink between the molecules to get in there."

## DESCRIPTION

This is a task for Ant-Man and his 'army of ants'. But, first, we need to do a census of the fighters.

Create a class called `Ant` with the following properties:

- `name`
- `role_in_army`
- `date_of_entry` (into the army)
- `number_of_fights`
- `number_of_legs`

Create some instances of the class and try to `echo` them to the console. You wouldn't be able to do this. Figure out how to do this without using the `__toString()` method.

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

## SYNOPSIS

```php
<?php

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

include_once ("Ant.php");

$ant = new Ant("Anthony", "sergeant", "2015-07-16", 1, 4);

$serialized = serialize($ant);
echo $serialized . "\n\n";

$unserialized = unserialize($serialized);
echo $unserialized;
```

## CONSOLE OUTPUT

```
>php t03/test.php | cat -e
O:3:"Ant":5:{s:4:"name";s:7:"Anthony";s:12:"role_in_army";s:8:"sergeant";s:13:"date_of_entry";
s:10:"2015-07-16";s:16:"number_of_fights";i:1;s:14:"number_of_legs";i:4;}$
$
name: Anthony$
role_in_army: sergeant$
date_of_entry: 2015-07-16$
number_of_fights: 1$
number_of_legs: 4$
```

# Act: Task 04

## NAME

Try, catch

## DIRECTORY

`t04/`

## SUBMIT

`EatException.php, Product.php, Ingestion.php`

## ALLOWED

PHP

## LEGEND

"[arguing over which Avenger is strong enough to wield the Infinity Gauntlet]
Thor: Do you know what is coursing through my veins right now?
James Rhodes: Cheez Whiz?"

## DESCRIPTION

Since we are helping the Avengers, there is another character who needs help.
Let's help Thor lose weight!

Create an exception class named `EatException` with a message "No more junk food, dumpling".
Create a class `Product` with:

- `name`
- `kcal_per_portion`

You should determine whether it is junk food or not by the amount of calories per portion.
If the amount is more than 200 - it's junk food. Otherwise - healthy food.

Create a class `Ingestion` with:

- `id`
- `meal_type: array (breakfast, lunch, dinner)`
- `day_of_diet`
- `products: array`
- `get_from_fridge($product): void - if it's junk food - throw the exception`

Test your code with the script given in the SYNOPSIS. The CONSOLE OUTPUT will differ for every call, because the calories amount is a random number, but the logic must work the same. Test with more cases.

## SYNOPSIS

```php
<?php

/*
  Task 04 (test.php)
  Task name: Try, catch
*/

require_once("EatException.php");
require_once("Ingestion.php");
require_once("Product.php");

$namesProducts = [
  'Nutella',
  'Chicken',
  'Coca-Cola',
  'Biscuit',
  'Brocolli',
  'Tomatoes',
  'Apple',
  'Potato',
  'Pizza',
  'Beer'
];

$stock = new Ingestion('breakfast', 1);

foreach ($namesProducts as $name) {
  $stock->setProduct(new Product($name, rand(40, 500)));
}

$allProducts = $stock->getProducts();
foreach ($namesProducts as $product) {
  $count = rand(1, 5);
  try {
    echo "***\nGetting " . $allProducts[$product]->getName() . " that has ";
    echo $allProducts[$product]->getKcal() . " calories.\n";
    $stock->get_from_fridge($product);
    echo "You're doing great, " . $product . " is good!\n";
  } catch (EatException $e) {
    echo "Caught exception: ". $e->getMessage() . "! ";
    echo "Throw " . $product . " away!\n";
  }
}
```

## CONSOLE OUTPUT

```
>php t04/test.php | cat -e
***$
Getting Nutella that has 413 calories.$
```

```
Caught exception: No more junk food, dumpling! Throw Nutella away!$
***$
Getting Chicken that has 476 calories.$
Caught exception: No more junk food, dumpling! Throw Chicken away!$
***$
Getting Coca-Cola that has 77 calories.$
You're doing great, Coca-Cola is good!$
***$
Getting Biscuit that has 186 calories.$
You're doing great, Biscuit is good!$
***$
Getting Brocolli that has 352 calories.$
Caught exception: No more junk food, dumpling! Throw Brocolli away!$
***$
Getting Tomatoes that has 416 calories.$
Caught exception: No more junk food, dumpling! Throw Tomatoes away!$
***$
Getting Apple that has 90 calories.$
You're doing great, Apple is good!$
***$
Getting Potato that has 340 calories.$
Caught exception: No more junk food, dumpling! Throw Potato away!$
***$
Getting Pizza that has 397 calories.$
Caught exception: No more junk food, dumpling! Throw Pizza away!$
***$
Getting Beer that has 203 calories.$
Caught exception: No more junk food, dumpling! Throw Beer away!$
```

# Act: Task 05

## NAME

Namespace "Quantum"

## DIRECTORY

`t05/`

## SUBMIT

`index.php, Quantum/index.php, Normal/index.php`

## ALLOWED

PHP

## LEGEND

"Last night, we powered up the tunnel for the first time.
It was overloaded and it shut down. But for a split second, the doorway to the quantum realm was opened."

## DESCRIPTION

Remember how Scott Lang got stuck in quantum space? And who can remember how much time he spend in there? We know that there are no such concepts as time and space in the quantum dimension. Imagine that you fell into a quantum space in 1 of January in 1939 (we let you guess why exactly this year). Let's assume that 7 normal years are considered as 1 year in the quantum space.

So, create a function `calculate_time(): datetime` in the Space\Normal namespace, which returns to you how much time has passed since 1939 in normal space in format «years-months-days». And the function `calculate_time(): array` in the Space\Quantum namespace, which returns the same time by the standards of quantum space. We know that there are no such concepts as time and space in the quantum dimension. It's not as easy as it seems. Print your results to the console.

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

## SYNOPSIS

```
Space\Normal\calculate_time()
Space\Quantum\calculate_time()
```

```php
<?php

/*
  Task 05 (test.php)
  Task name: Namespace "Quantum"
*/

require_once __DIR__ . '/Normal/index.php';
require_once __DIR__ . '/Quantum/index.php';

$time = Space\Normal\calculate_time();
echo "In real life you were absent for " . $time->format("%Y") . " years, " .
    $time->format("%m") . " months, " . $time->format("%d") . " days\n";

$time = Space\Quantum\calculate_time();
echo "\nIn quantum space you were absent for " . $time[0] . " years, " .
    $time[1] . " months, " . $time[2] . " days\n";
```

## CONSOLE OUTPUT

```
>php t05/test.php | cat -e
In real life you were absent for 81 years, 7 months, 30 days$
$
In quantum space you were absent for 11 years, 11 months, 30 days$
```

# Act: Task 06

## NAME

Go web!

## DIRECTORY

`t06/`

## SUBMIT

`Quantum/index.php, Normal/index.php`

## ALLOWED

PHP, HTML

## LEGEND

"Hi...uhhh...is anyone home? This is Scott Lang.
We met a few years ago... at the airport...in Germany...I got really big."

## DESCRIPTION

Were you able to cope with the previous task? Great, because we have another one.

Now you should display the same information on the web page. On the page Normal/index.php you should display how much time has passed since 1939 in normal space. And on the page Quantum/index.php display the same time by the standards of quantum space.

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

## CONSOLE OUTPUT

```
>php t06/Quantum/index.php | cat -e
<!DOCTYPE html>$
<html>$
$
<head>$
  <meta charset="utf-8">$
  <title>Quantum space</title>$
</head>$
$
<body>$
  <p>In quantum space you were absent for 11 years, 11 months, 30 days!</p></body>$
$
</html>$
```

# Act: Task 07

## NAME

SERVER

## DIRECTORY

`t07/`

## SUBMIT

`index.php`

## ALLOWED

`$_SERVER`

## LEGEND

"Heilmdail: Be warned, I shall uphold my sacred oath to protect this realm as its gatekeeper. If your return threatens the safety of Asgard, my gate will remain shut and you will be left to perish on the cold waste of Jotunheim."

## DESCRIPTION

Heimdail is an Asgardian gifted with sensory abilities far superior to the capabilities of other Ases, which allows him to see almost everything that happens in the Nine Worlds. Like Heimdail, the global variable `$_SERVER` also sees almost everything that's happening on the web page.

`$_SERVER` - it's an array. Discover it and show on the page this information:

- `a name of file of the executed script`
- `arguments passed to the script`
- `IP address of the server`
- `a name of host that invoke current script`
- `a name and a version of the information protocol`
- `a query method`
- `User-Agent information`
- `IP address of the client`
- `a list of parameters passed by URL`

# Act: Task 08

## NAME

What about forms?

## DIRECTORY

`t08/`

## SUBMIT

`index.php`

## ALLOWED

PHP, HTML

## LEGEND

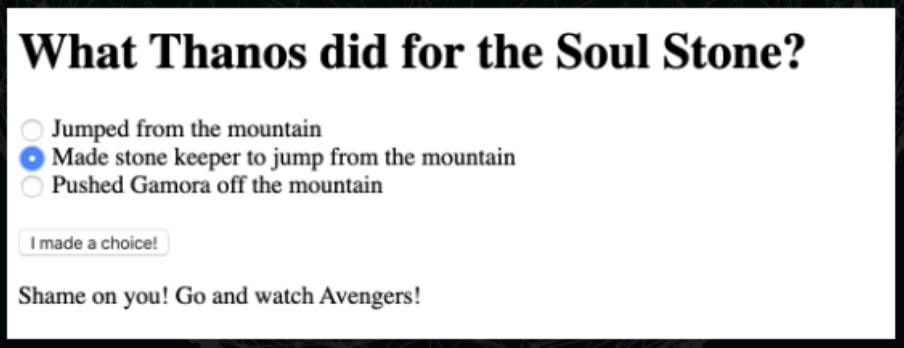
"The hardest choices require the strongest wills."

## DESCRIPTION

Do you remember what Thanos did to get the Soul Stone?

Let's create a little quiz. Ask any question on the web page which is related to the Avengers. Make different types of answer. Your option fields should be rounded.

## EXAMPLE



# Act: Task 09

## NAME

A new set

## DIRECTORY

`t09/`

## SUBMIT

`index.php`

## ALLOWED

PHP, HTML

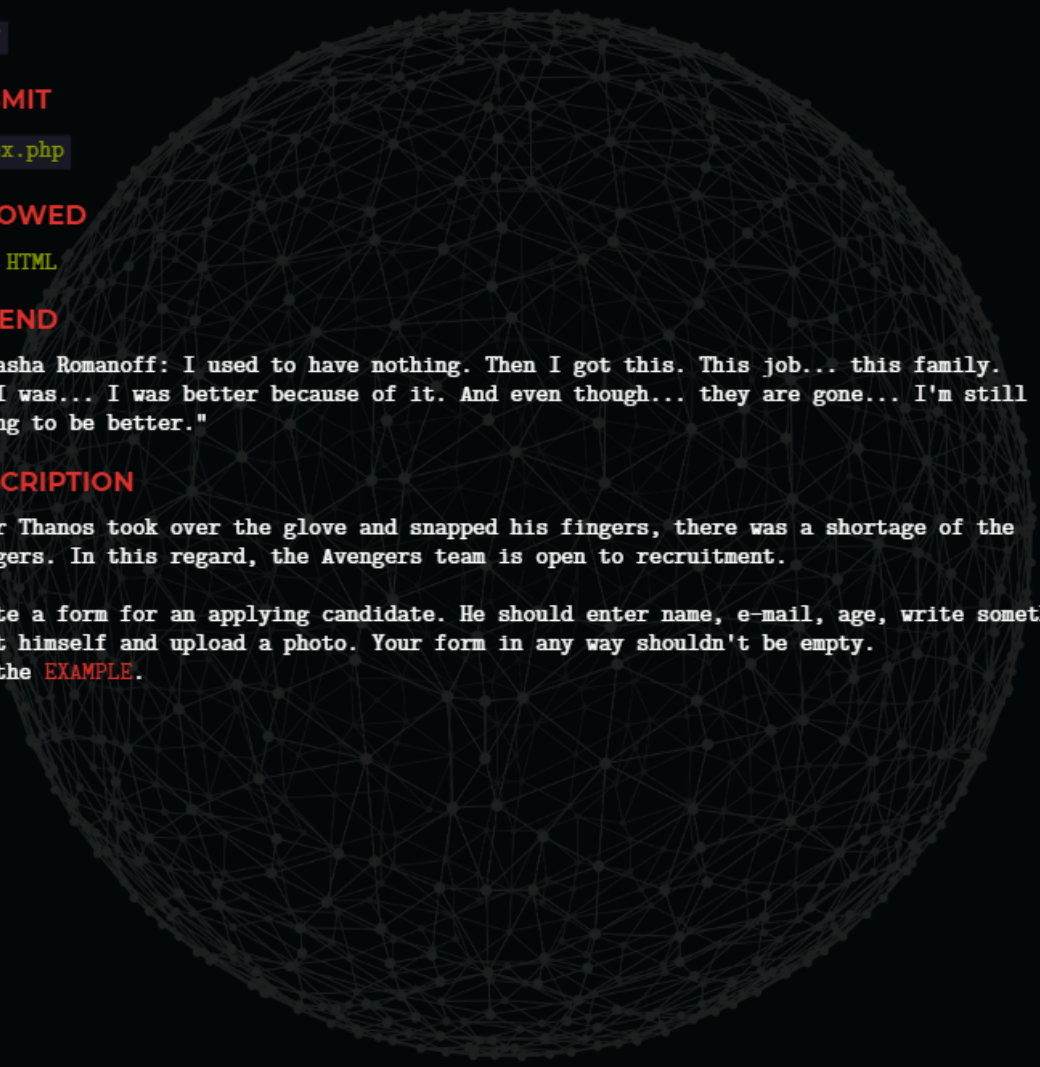
## LEGEND

"Natasha Romanoff: I used to have nothing. Then I got this. This job... this family. And I was... I was better because of it. And even though... they are gone... I'm still trying to be better."

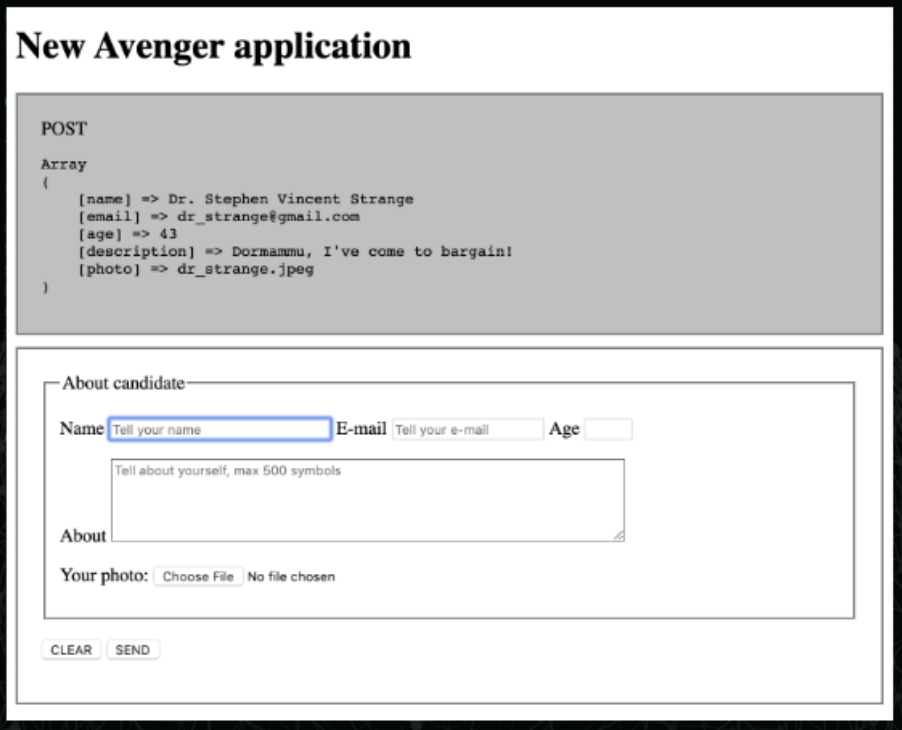
## DESCRIPTION

After Thanos took over the glove and snapped his fingers, there was a shortage of the Avengers. In this regard, the Avengers team is open to recruitment.

Create a form for an applying candidate. He should enter name, e-mail, age, write something about himself and upload a photo. Your form in any way shouldn't be empty.
See the EXAMPLE.



EXAMPLE


New Avenger application
POST
Array
(
[name] => Dr. Stephen Vincent Strange
[email] => dr_strange@gmail.com
[age] => 43
[description] => Dormammu, I've come to bargain!
[photo] => dr_strange.jpeg
)
About candidate
Name
Tell your name
E-mail
Tell your e-mail
Age
Tell about yourself, max 500 symbols
About
Your photo:
Choose File
No file chosen
CLEAR
SEND

# 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

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.