Level 2

Arrays

Simple & Associative Arrays

Why an Array?

Variables alone will not scale. We need a better way to keep our data.

index.php

```php
<?php
// We could keep going with variables
$meteor_1 = 'Hoba';
$meteor_2 = 'Cape York';
$meteor_3 = 'Campo del Cielo';
$meteor_4 = 'Canyon Diablo';
…
$meteor_42 = 'Prefect';
```

Arrays, a Map

An array maps values to keys, like an address for setting and recalling.

Key	Value
0	Hoba
1	Cape York
2	Campo del Cielo
3	Canyon Diablo

Creating an Array

Let's create an empty array to hold our meteorite data.

index.php

```php
<?php
// Create our array, using the php array function
$meteors = array();

// Create our array, using a shortcut available since 5.4
$meteors = [];
```

The array function has good readability

Here, the brackets define an empty array

Array With Values

We can create an array with one or more key value pairs using the same function.

index.php

```php
<?php
// Create our array with a single value
$meteors = array('Hoba');
$meteors = ['Hoba'];

// Create array with multiple values
$meteors = array('Hoba','Cape York');

// Echo the array
echo($meteors);
```

echo will not show the data within the array

Output

```
Array
```

Array With Values

We can create an array with one or more key value pairs using the same function.

index.php

```php
<?php
// Create our array with a single value
$meteors = array('Hoba');
$meteors = ['Hoba'];

// Create array with multiple values
$meteors = array('Hoba','Cape York');

// Let's take a look at our array
with an internal function
print_r($meteors);
```

print_r will echo human-readable output

Output

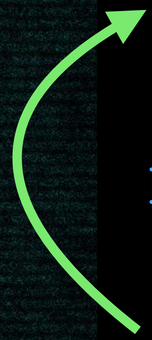
```
Array (
[0] => Hoba
[1] => Cape York
)
```

Adding More Data to Our Array

We can append new values by placing square brackets after the array variable.

index.php

```php
<?php
…
// Let's add two more items
$meteors[] = 'Campo del Cielo';
$meteors[] = 'Canyon Diablo';

print_r($meteors);
```



Empty brackets after the variable name indicate a new item in the array

Output

```
Array (
[0] => Hoba
[1] => Cape York
[2] => Campo del Cielo
[3] => Canyon Diablo
)
```

How Can We Access This Data?

Placing the key, or index, inside the square bracket gives us access to the value.

index.php

```php
<?php
$meteors = array(
    'Hoba',
    'Cape York',
    'Campo del Cielo',
    'Canyon Diablo'
    );

echo $meteors[0];
```

Remember: Array keys are 0 indexed

Output

```
Hoba
```

How Can We Access This Data?

Placing the key, or index, inside the square bracket gives us access to the value.

index.php

```php
<?php
$meteors = array(
    'Hoba',
    'Cape York',
    'Campo del Cielo',
    'Canyon Diablo'
    );

echo $meteors[0];
echo $meteors[1];
echo $meteors[3];
```

Output

```
Hoba
Cape York
Canyon Diablo
```

Modifying an Existing Item

Placing the key inside also allows us access to modify the value.

index.php

```php
<?php
…
$meteors[0] = 'Los Angeles';

print_r($meteors);
```

Choose your key to modify

Then modify the value

Output

```
Array (
[0] => Los Angeles
[1] => Cape York
[2] => Campo del Cielo
[3] => Canyon Diablo
)
```

Storing Even More Data in an Array

What if we want to store more information about the meteorite?

Name	Weight	Location	Year
Hoba	600000000	-19.58333, 17.91667	1920
Cape York	58200000	76.13333, -64.93333	1818
Campo del Cielo	50000000	-27.46667, -60.58333	1576
Canyon Diablo	30000000	35.05, -111.03333	1891

Associative vs. Index Arrays

Associative arrays allow us to use strings as the key.

index.php

```php
<?php
// Create an associative array
$meteors = array(
    'Hoba' => 600000000,
    'Cape York' => 58200000,
);

print_r($meteors);
```

This array operator associates keys with values

The name is our key

Output

```
Array (
[Hoba] => 600000000
[Cape York] =>
58200000
)
```

Accessing an Item in the Array

Instead of the numerical index, we now use the string key for access.

index.php

```php
<?php

// Access our data.
echo $meteors['Hoba'];
echo $meteors['Cape York'];
```

Output

```
600000000
58020000
```

Appending a New Item

Using a string key, we can add a new item as well.

index.php

```php
<?php
// Add new meteorite data.
$meteors['Canyon Diablo'] = 30000000;

print_r($meteors);
```

Place the key inside of square brackets

Then set your value

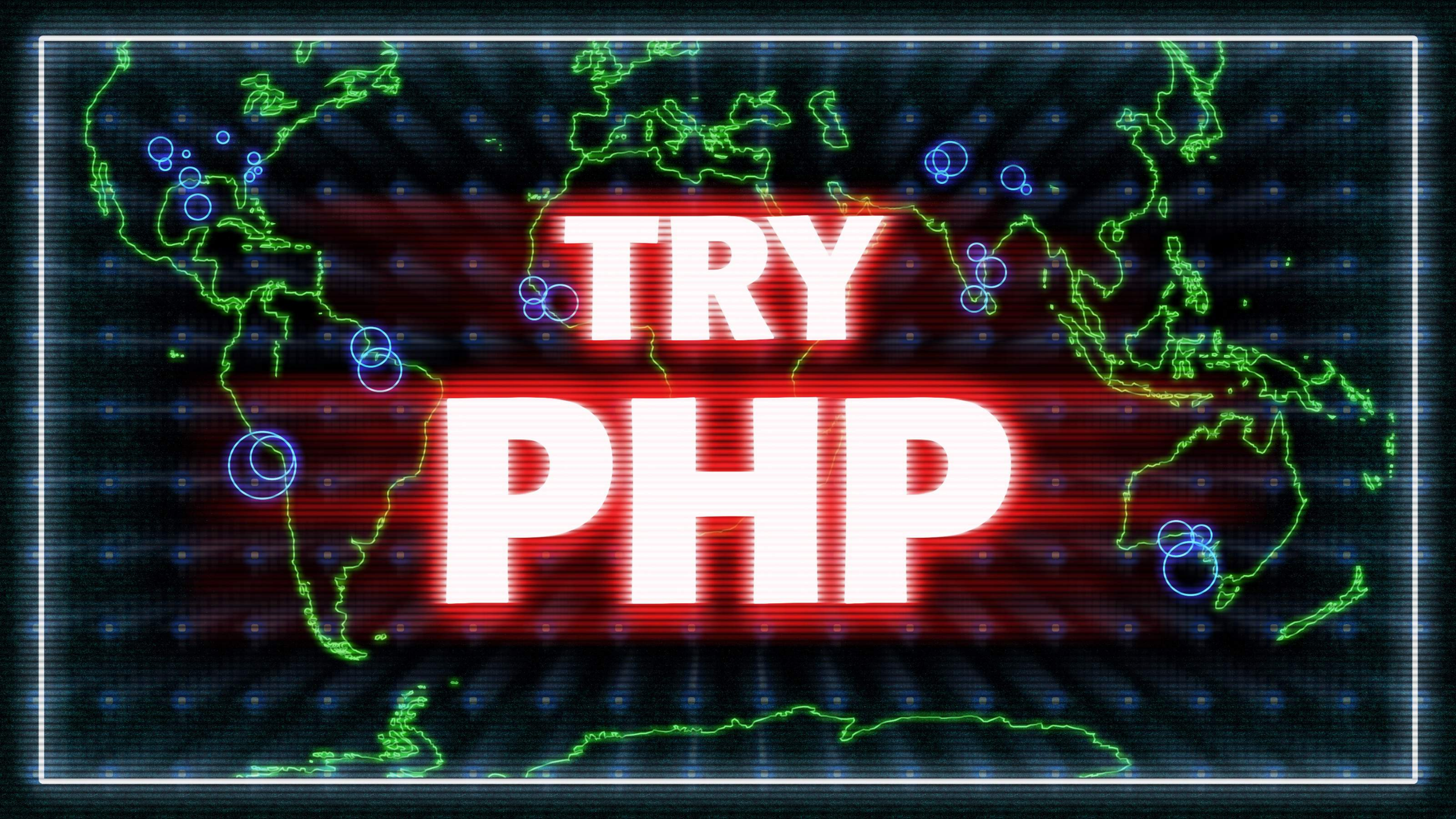
Output

```
Array(
    ...
    [Canyon Diablo]
        => 30000000
)
```

What Have We Learned?

Let’s have a quick review.

- Numerical indexed arrays
- Associative arrays
- Array creation with values
- Modification of array data


TRY
PHP

Level 2

Arrays

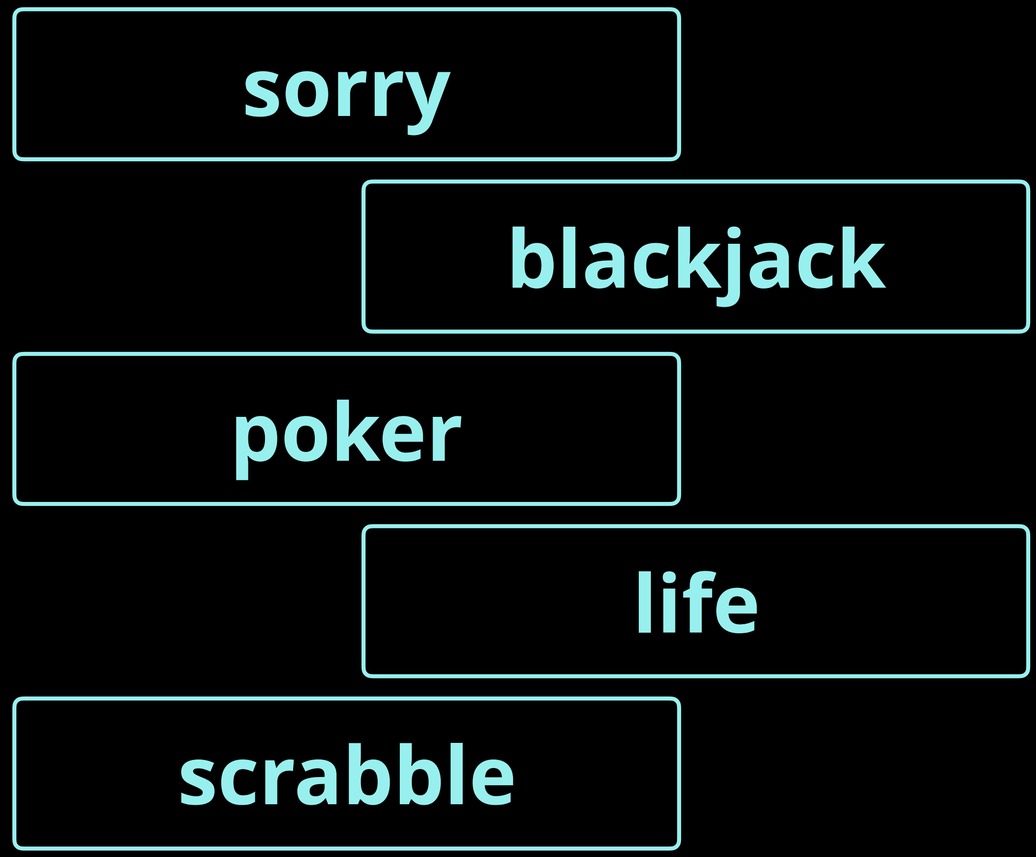
Multidimensional Arrays & Array Functions

An Array of Games

Arrays can help us organize data.

index.php

```php
<?php

$games = array(
    'sorry',
    'blackjack',
    'poker',
    'life',
    'scrabble',
);
```



Groups of Games

How can we better organize our list of games?

- sorry
- blackjack
- poker
- life
- scrabble

Imagining Two Groups

Splitting our games into two groups can help us sort and recall the data.

Tabletop Games	*Card Games*
sorry	blackjack
life	poker
scrabble	

Another Dimension

index.php

```php
<?php
$games = array(
    'tabletop' => 'sorry'
);
```

Tabletop Games

sorry

Another Dimension

index.php

```php
<?php
$games = array(
    'tabletop' => array()
);
```

Another Dimension

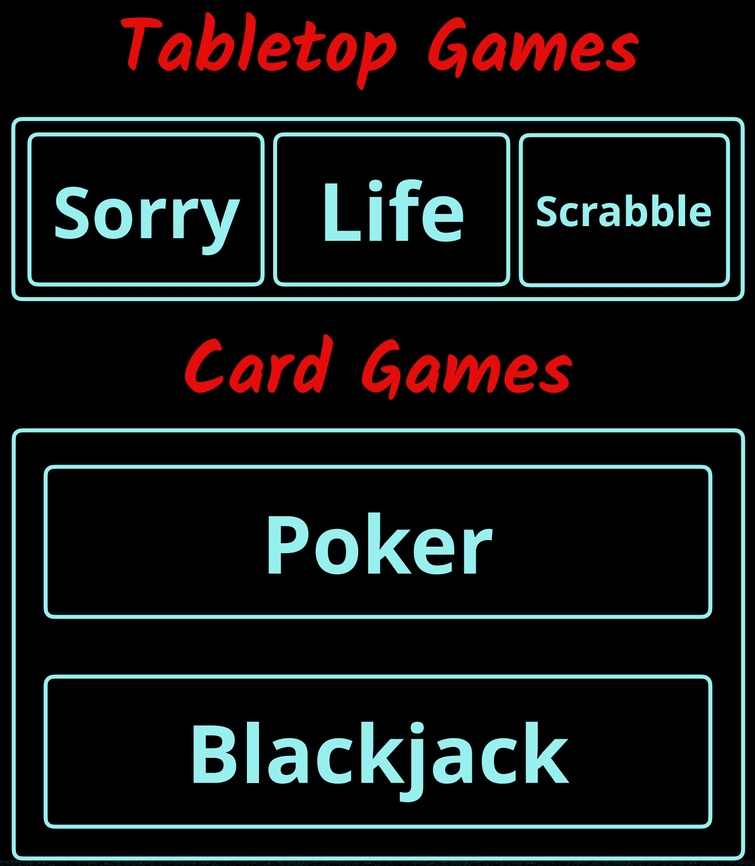
index.php

```php
<?php
$games = array(
    'tabletop' => array(
        'sorry',
        'life',
        'scrabble',
    ),
);
```

Another Dimension

index.php

```php
<?php
$games = array(
    'tabletop' => array(
        'sorry',
        'life',
        'scrabble',
    ),
    'card' => array(
        'poker',
        'blackjack',
    ),
);
```



Array Inspection

If we print_r our $games array, you can see the multidimensional structure.

```php
<?php
print_r($games);
```

Output

```
Array(
  [tabletop] => Array(
    [0] => sorry
    [1] => life
    [2] => scrabble
  [card] => Array(
    [0] => poker
    [1] => blackjack
)
```

Accessing Data

By using the array's key, we can see the array value.

index.php

```php
<?php

print_r($games['tabletop']);
```

Output

```
Array (
[0] => sorry
[1] => life
[2] => scrabble
)
```

Accessing Data

By using the array's key, we can see the array value.

index.php

```php
<?php

print_r($games['tabletop']);

print_r($games['card']);
```

Output

```
Array (
[0] => poker
[1] => blackjack
)
```

Accessing Data

By using the array's key, we can see the array value.

index.php

```php
<?php

print_r($games['tabletop']);

print_r($games['card']);

echo $games['tabletop'][0];
```

Output

```
sorry
```

Modifying the Data

Instead of single item access, we can use the same methods to change a value.

index.php

```php
<?php

$games['card'][0] = 'rummy';

print_r($games['card']);
```

Output

```
Array (
[0] => rummy
[1] => blackjack
)
```

Array Functions: count

This function lets us count all the items in an array.

index.php

```php
<?php
$people = array(
    'David',
    'Jennifer',
    'Falken',
    'Joshua',
);

echo count($people);
```

Output

```
4
```

Array Functions: implode

implode **joins all elements of the array into a string.**

index.php

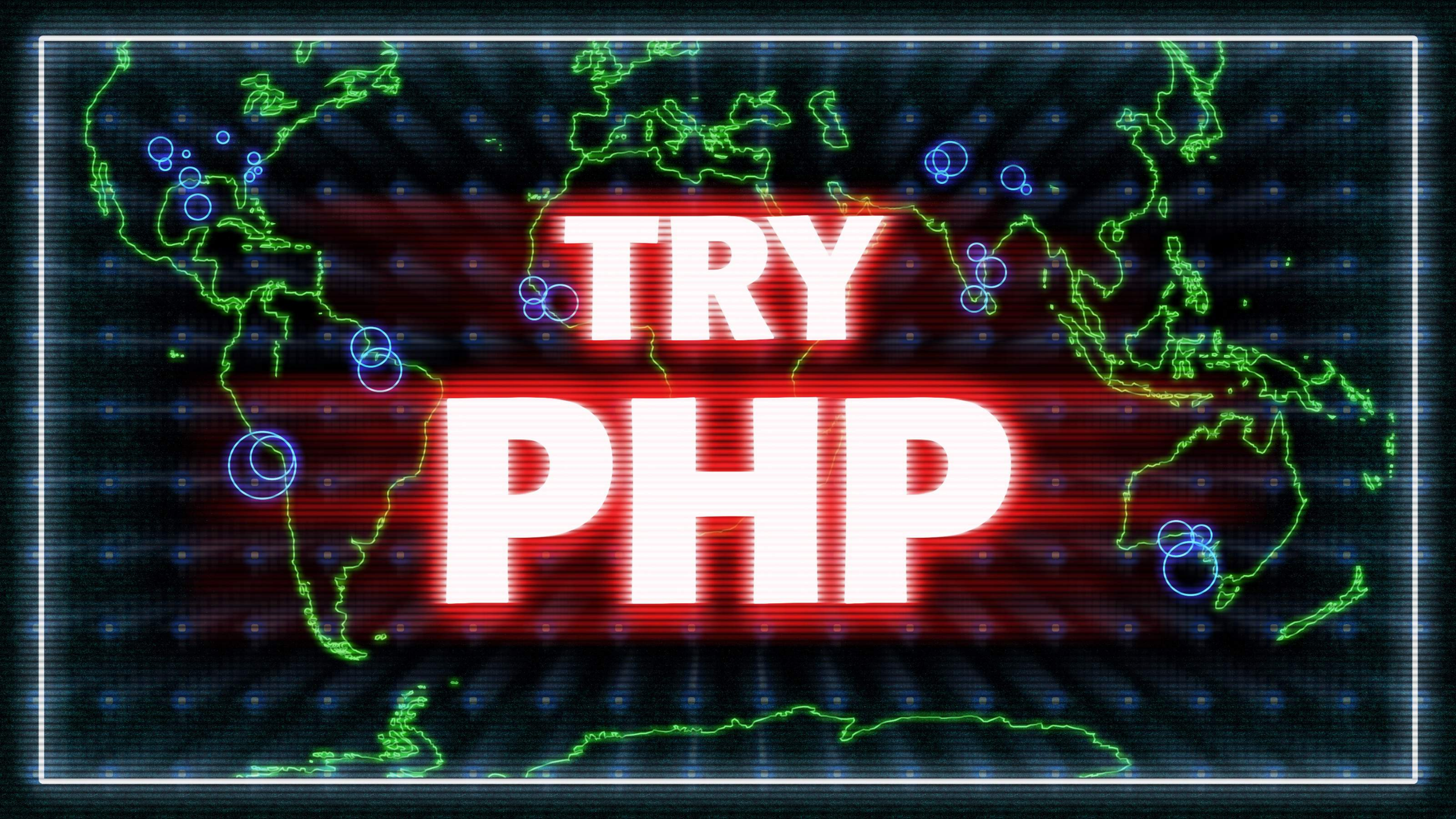
```php
<?php
$people = array(
    'David',
    'Jennifer',
    'Falken',
    'Joshua',
);

echo implode(' ', $people);
```

the character that separates the combined array values

the array to combine

Output

```
'David Jennifer Falken Joshua'
```


TRY
PHP

Level 3

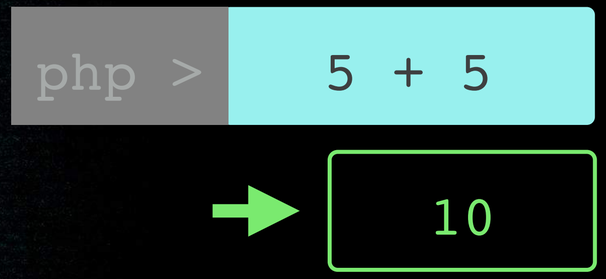
Conditionals & Operators

What If? Now What? What Else?

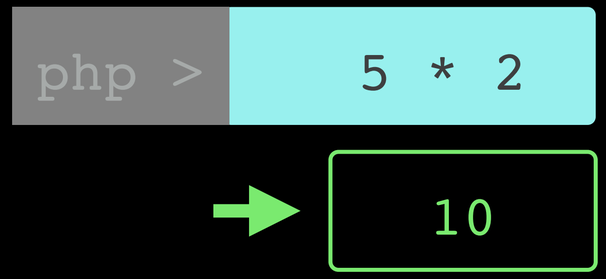
Arithmetic Operators

These are some of the arithmetic operators available to us in PHP.

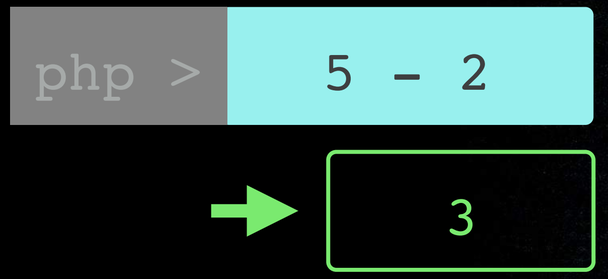
Addition



Multiplication



Subtraction



Division

Exponent

Comparison Operators

These are some of the comparison operators available to us in PHP.

Equal

```
php > 5 == '5'
```

```
true
```

Greater Than or Equal To

```
php > 5 >= 5
```

```
true
```

Not Equal To

```
php > 5 != 2
```

```
true
```

Greater Than

```
php > 5 > 2
```

```
true
```

Less Than

```
php > 2 < 5
```

```
true
```

Identical

```
php > 5 ==='5'
```

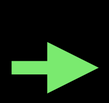
```
false
```

Identical Comparison Operator

To be identical, the items must be of the same type and value.

Identical

```
php > 5 ==='5'
```



```
false
```

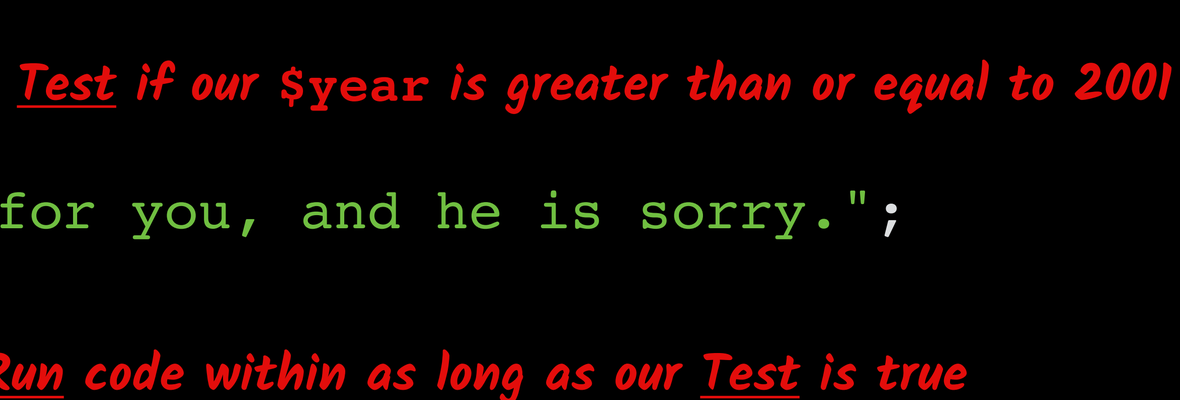
```
5    // is integer data
'5'  // is string data
```

Control Flow

The if statement allows us to execute code based on a condition.

index.php

```php
<?php
$year = 2016;

if ($year >= 2001) {
    echo "Hal can't do that for you, and he is sorry.";
}
```



Default Condition

The else statement allows us to run code when the if returns false.

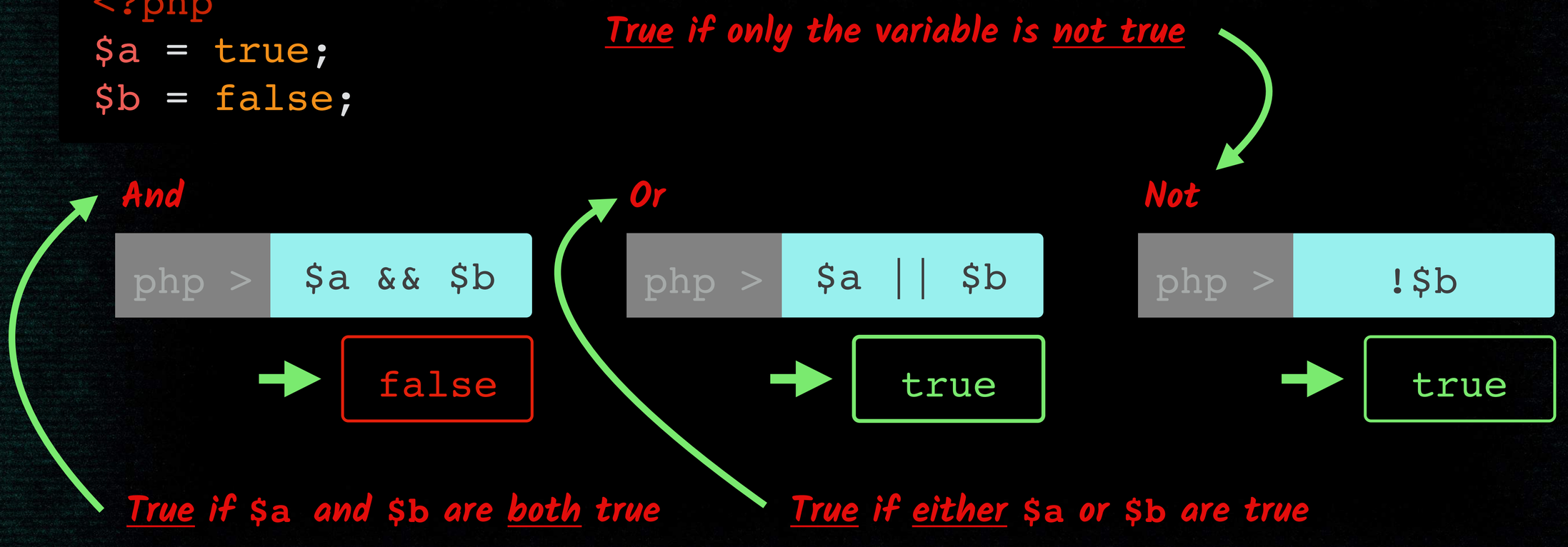
index.php

```php
<?php
$year = 2016;

if ($year >= 2001) {
    echo "Hal can't do that for you, and he is sorry.";
} else {
    echo "You still have time. Destroy the machines!";
}
```

Logical Operators

These are some of the logical operators available to us in PHP.

```php
<?php
$a = true;
$b = false;
```



Testing Multiple Conditions

Using the logical operator and, we can test to see if multiple conditions are true.

index.php

```php
<?php
$year = 1984;

if ($year >= 1994 && $year < 2001){
    echo "Skynet is growing stronger every day.";
} else {
    echo "You still have time. Destroy the machines!";
}
```

Multiple if Statements

The elseif statement allows us to have multiple conditions.

index.php

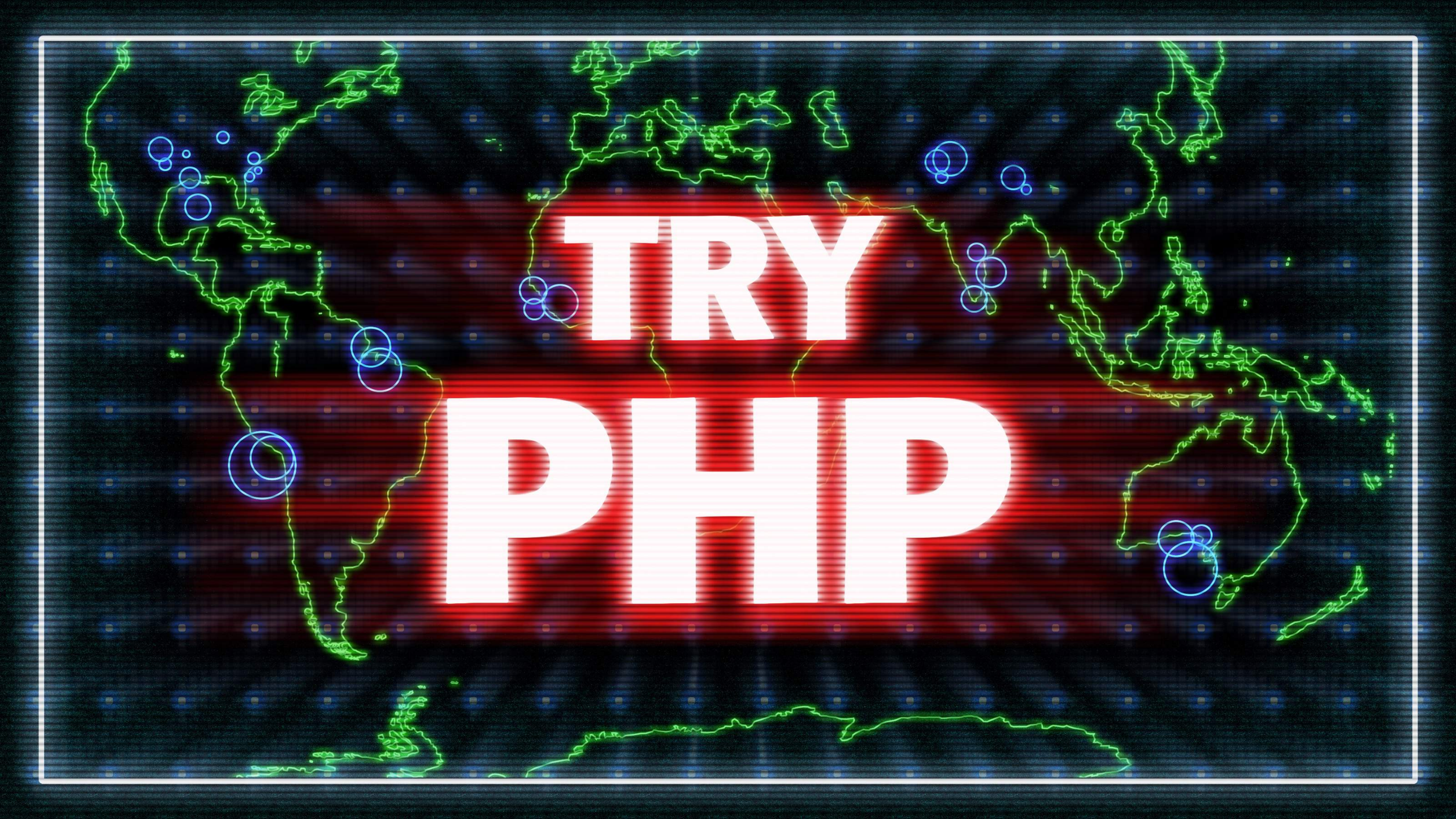
```php
<?php
$year = 1984;

if ($year >= 2001) {
    echo "Hal can't do that for you, and he is sorry.";
} elseif ($year >= 1984) {
    echo "Eurasia has fallen! Rejoice with Big Brother.";
} else {
    echo "You still have time. Destroy the machines!";
}
```

Test this if the first condition is false

What Have We Learned?

Let’s have a quick review.

- Comparison operators
- Arithmetic operators
- if, if-else, else comparisons
- Logical operators


TRY
PHP

Level 4

Loops

Cycle Through All the Data

Don’t Repeat Yourself

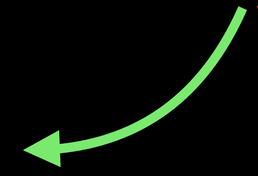
The DRY (or “Don’t Repeat Yourself”) method helps us keep our code efficient.

```php
<?php
    $value = 1*12;
    echo "1 times 12 is $value";
    $value = 2*12;
    echo "2 times 12 is $value";
    $value = 3*12;
    echo "3 times 12 is $value";
    $value = 4*12;
    echo "4 times 12 is $value";
    $value = 5*12;
    echo "5 times 12 is $value";
```

while **Loops**

Now let's initialize, test, and increment.

Initialize an integer variable and set it to 1



```php
<?php
$i = 1;

while( $i <= 12 ) {
    $value = $i * 12;
    echo "$i times 12 is $value";
    $i++;
}
```

while **Loops**

Now let's initialize, test, and increment.

Initialize an integer variable and set it to 1

Test if our variable is less than or equal to 12

```php
<?php
$i = 1;

while( $i <= 12 ) {
    $value = $i * 12;
    echo "$i times 12 is $value";
    $i++;
}
```

while **Loops**

Now let's initialize, test, and increment.

Initialize an integer variable and set it to 1

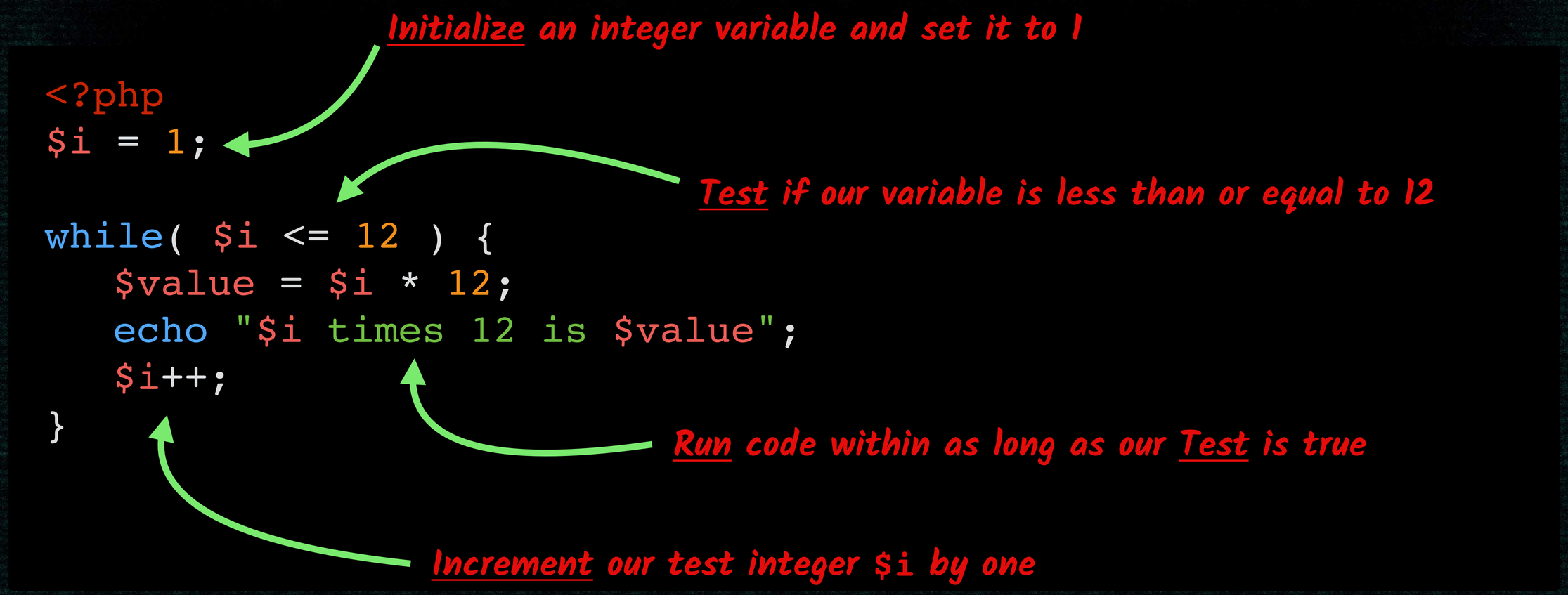
```php
<?php
$i = 1;

while( $i <= 12 ) {
    $value = $i * 12;
    echo "$i times 12 is $value";
    $i++;
}
```

Test if our variable is less than or equal to 12

Run code within as long as our Test is true

while **Loops**

Now let's initialize, test, and increment.



while **Loops**

Now let's initialize, test, and increment.

```php
<?php
$i = 1;

while( $i <= 12 ) {
    $value = $i * 12;
    echo "$i times 12 is $value";
    $i++;
}
```

Output

```
1 times 12 is 12
2 times 12 is 24
3 times 12 is 36
...
10 times 12 is 120
11 times 12 is 132
12 times 12 is 144
```

Using a for Loop

Now let's initialize, test, and increment.

```php
<?php

for( $i = 1; $i <= 12; $i++) {
    $value = $i * 12;
    echo "$i times 12 is $value";
}
```

Initialize an integer variable and set it to 1

Test if our variable is less than or equal to 12

Increment our integer variable `$i` by one

`$i++` is the same as `$i = $i + 1`

Using a for Loop

Now let's initialize, test, and increment.

```php
<?php

for( $i = 1; $i <= 12; $i++) {
    $value = $i * 12;
    echo "$i times 12 is $value";
}
```

Output

```
1 times 12 is 12
2 times 12 is 24
3 times 12 is 36
...
10 times 12 is 120
11 times 12 is 132
12 times 12 is 144
```

The Simple Meteorite Array

How else could we extract each item in the array other than direct access?

```php
<?php
$meteors = array(
    'Hoba',
    'Cape York',
    'Campo del Cielo',
    'Canyon Diablo',
    );
```

Looping Access to the Array

The foreach and as will allow us to cycle through each item in our array.

```php
<?php
$meteors = array(
    'Hoba',
    'Cape York',
    'Campo del Cielo',
    'Canyon Diablo',
    );

foreach($meteors as $meteor) {
    echo $meteor;
}
```

On each pass through our `foreach` loop, the data in `$meteor` will update with the next item in the collection.

The value, our meteorite names

Output

Hoba
Cape York
Campo del Cielo
Canyon Diablo

Associative Meteorite Array

What would happen if we ran this array through our existing foreach loop?

```php
<?php
$meteors = array(
    'Hoba' => 600000000,
    'Cape York' => 58200000,
    'Campo del Cielo' => 50000000,
    'Canyon Diablo' => 30000000,
    );
```

Looping Through an Associative Array

What would happen if we ran this array through our existing foreach loop?

```php
<?php
$meteors = array(
    'Hoba' => 600000000,
    'Cape York' => 58200000,
    'Campo del Cielo' => 50000000,
    'Canyon Diablo' => 30000000,
    );

foreach($meteors as $meteor) {
    echo $meteor;
}
```

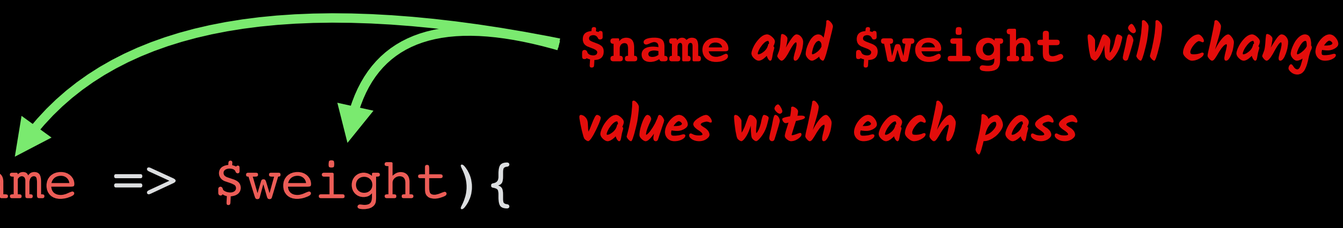
The value is our meteorite weight!

Output

How Can We Access the Key and Value?

We can use the array operator => to set up the key and value variables.

```php
<?php
$meteors = array(
    'Hoba' => 600000000,
    'Cape York' => 58200000,
    'Campo del Cielo' => 50000000,
    'Canyon Diablo' => 30000000,
    );


foreach($meteors as $name => $weight){

}
```



How Can We Access the Key and Value?

We can use the object operator => to set up the key and value variables.

```php
<?php
$meteors = array(
    'Hoba' => 600000000,
    'Cape York' => 58200000,
    'Campo del Cielo' => 50000000,
    'Canyon Diablo' => 30000000,
    );

foreach($meteors as $name => $weight){
    echo "$name weighs $weight grams.";
}
```

Output

```
Hoba weighs 600000000 grams.
...
Canyon Diablo weighs 30000000 grams.
```

A Complete Picture

```php
<?php
$meteors = array(
    'Hoba' => 600000000,
    'Cape York' => 58200000,
    'Campo del Cielo' => 50000000,
    'Canyon Diablo' => 30000000,
    );
$epic = 600000000; // 600 million grams
$huge = 50000000; // 50 million grams
?>
```

A Complete Picture

```php
<?php
$meteors = array(
    'Hoba' => 600000000,
    'Cape York' => 58200000,
    'Campo del Cielo' => 50000000,
    'Canyon Diablo' => 30000000,
    );
$epic = 600000000; // 600 million grams
$huge = 50000000; // 50 million grams
foreach ($meteors as $name => $weight) {

}
?>
```

A Complete Picture

```php
<?php
$meteors = array(
    'Hoba' => 600000000,
    'Cape York' => 58200000,
    'Campo del Cielo' => 50000000,
    'Canyon Diablo' => 30000000,
    );
$epic = 600000000; // 600 million grams
$huge = 50000000; // 50 million grams
foreach ($meteors as $name => $weight) {
    if ($weight >= $epic) {
        echo 'You have found an epic meteorite!<br>';
        echo 'Your meteorite\'s name is ' . $name . '<br>';
    }
}
?>
```

A Complete Picture

```php
<?php
$meteors = array('Hoba' => 600000000, ...);
$epic = 600000000; // 600 million grams
$huge = 50000000; // 50 million grams
foreach ($meteors as $name => $weight) {
    if ($weight >= $epic) {
        echo 'You have found an epic meteorite!<br>';
        echo 'Your meteorite\'s name is ' . $name . '<br>';
    } elseif ($weight >= $huge) {
        echo 'You have found a huge meteorite!<br>';
        echo 'Your meteorite\'s name is ' . $name . '<br>';
    }
}
?>
```

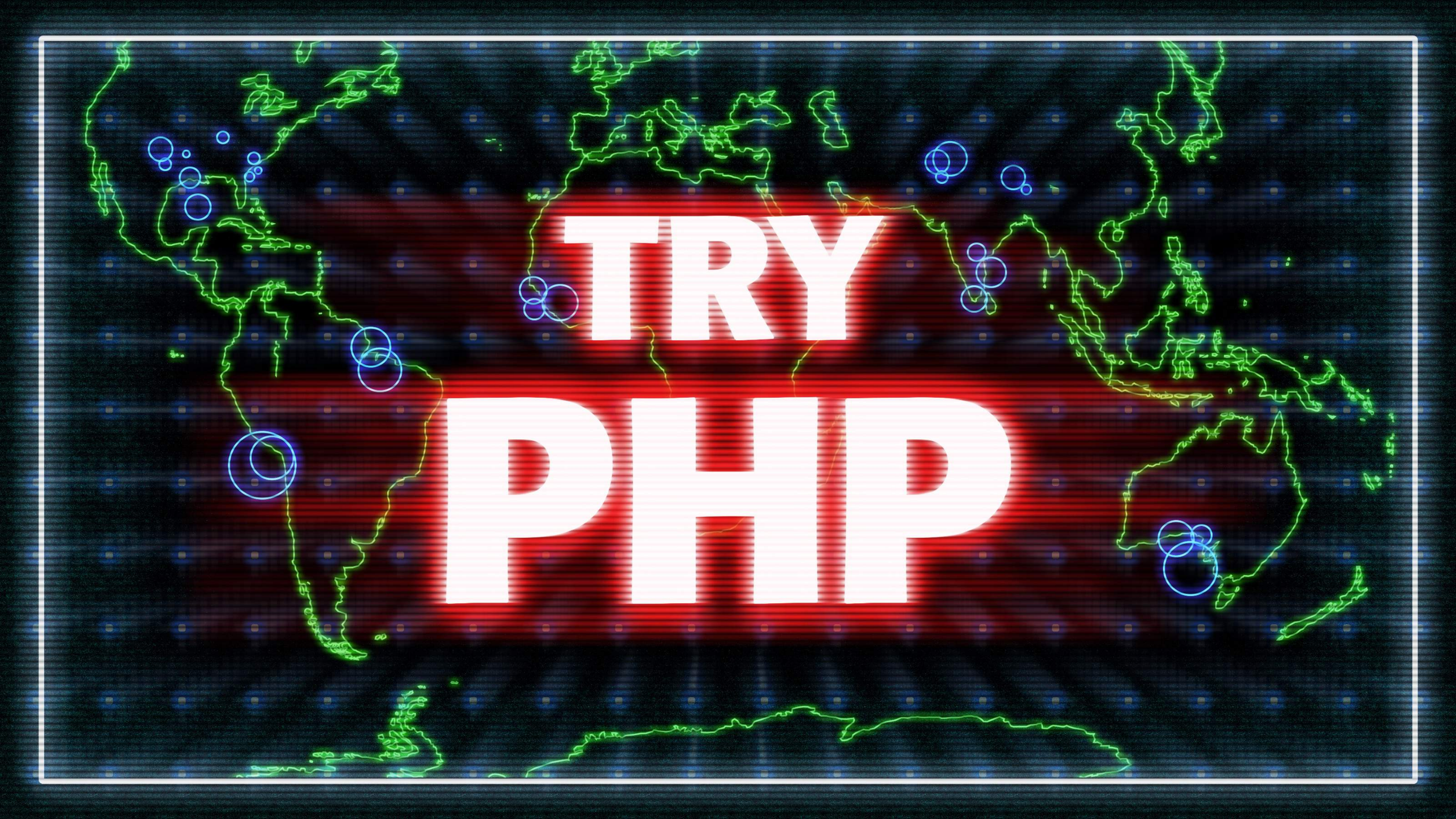
A Complete Picture

```php
<?php
$meteors = array('Hoba' => 600000000, ...);
$epic = 600000000; // 600 million grams
$huge = 50000000; // 50 million grams
foreach ($meteors as $name => $weight) {
    if ($weight >= $epic) {
        echo 'You have found an epic meteorite!<br>';
        echo 'Your meteorite\'s name is ' . $name . '<br>';
    } elseif ($weight >= $huge) {
        echo 'You have found a huge meteorite!<br>';
        echo 'Your meteorite\'s name is ' . $name . ‘<br>';
    } else {
        echo 'You have found a meteorite, awesome!<br>';
        echo 'Your meteorite\'s name is ' . $name . '<br>';
    }
}
?>
```

What Have We Learned?

Let’s have a quick review.

- while **loop**
- for **loop**
- foreach **loop**
- foreach **with key/value**
- **Combining loops and conditionals**


TRY
PHP