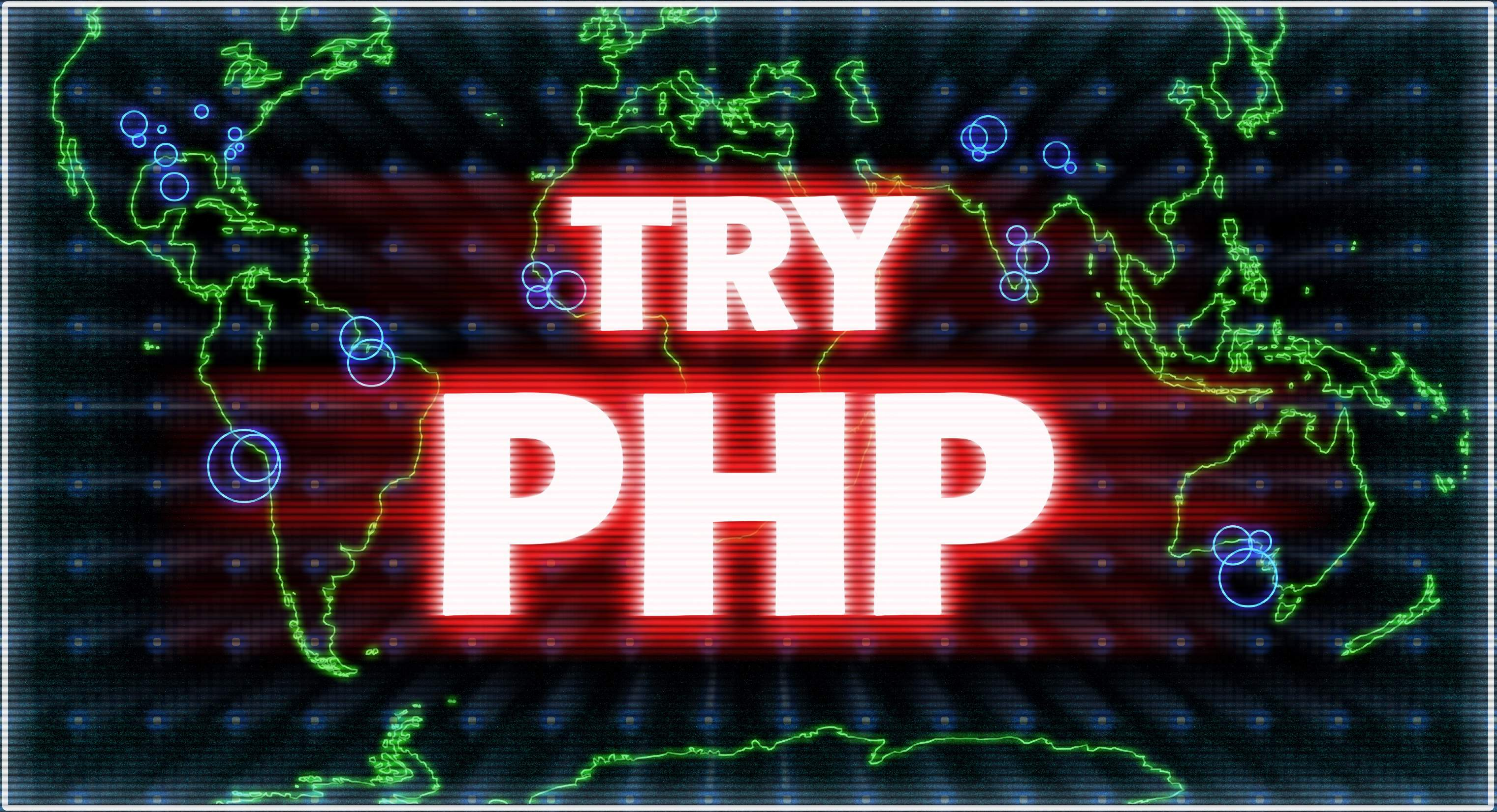
TRY PHP



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";
```

Assign the product of 1 and 12 to a variable.

echo our product

while **Loops**

Now let's initialize, test, and increment.

Initialize an integer variable and set it to 1

```
<?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.

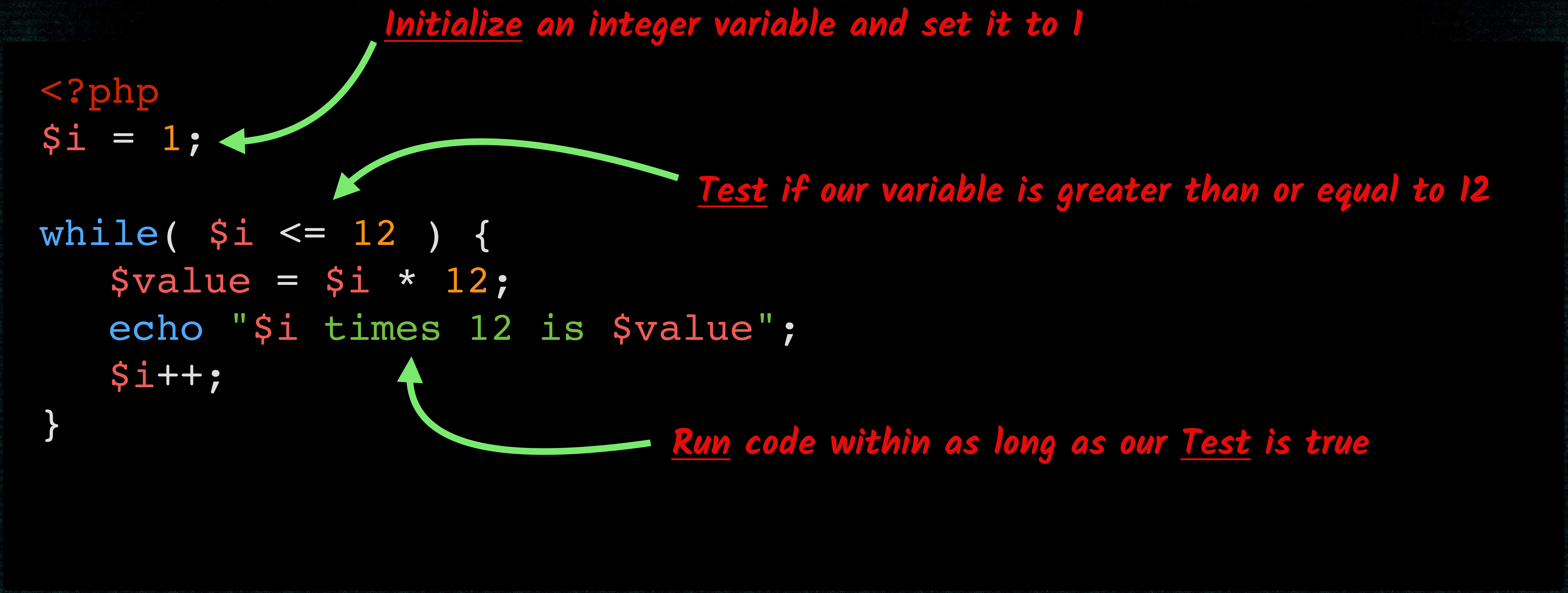
Initialize an integer variable and set it to 1

Test if our variable is greater than or equal to 12

```
<?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.



```
<?php
$i = 1;

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

Initialize an integer variable and set it to 1
Test if our variable is greater than or equal to 12
Run code within as long as our Test is true

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 greater than or equal to 12

Run code within as long as our Test is true

Increment our test integer $i by one

while **Loops**

Now let’s initialize, test, and increment.

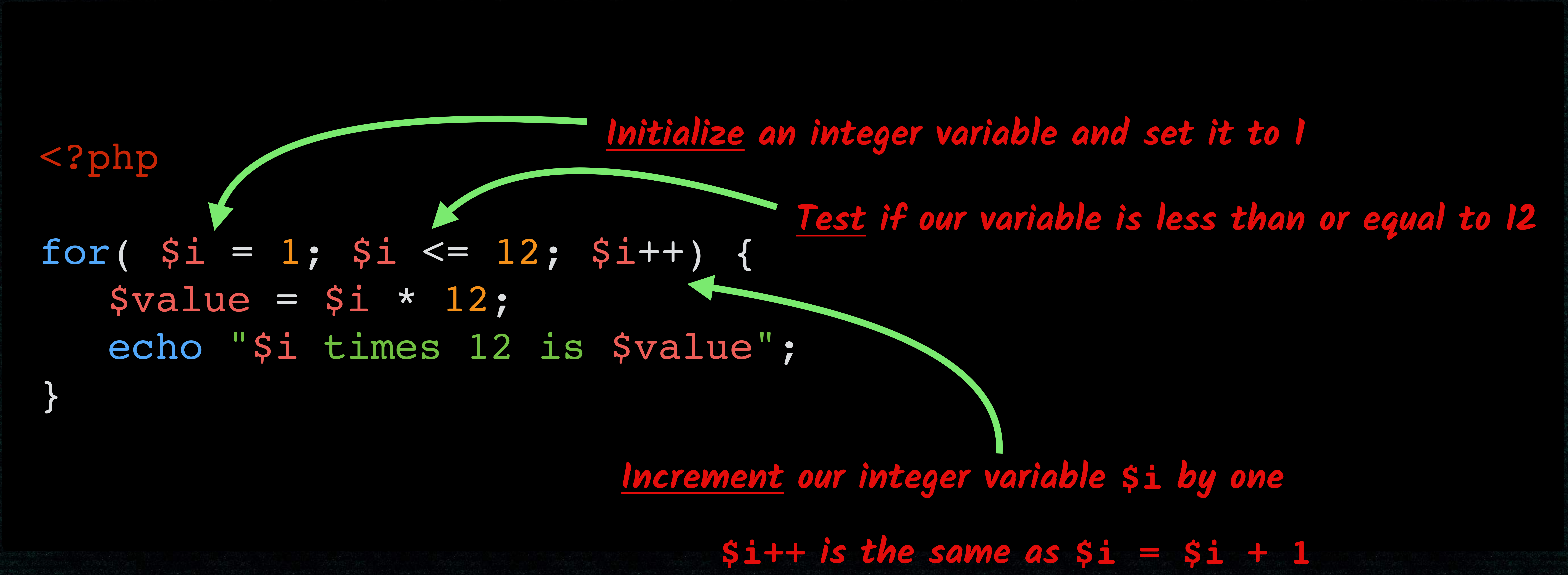
```php
<?php
$i = 1;

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

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.

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

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

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

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;
}
```

Output

```
600000000
58200000
50000000
30000000
```

The value is our meteorite weight!

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){

}
```

$name and $weight will change values with each pass

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>';
    }
}
?>
```

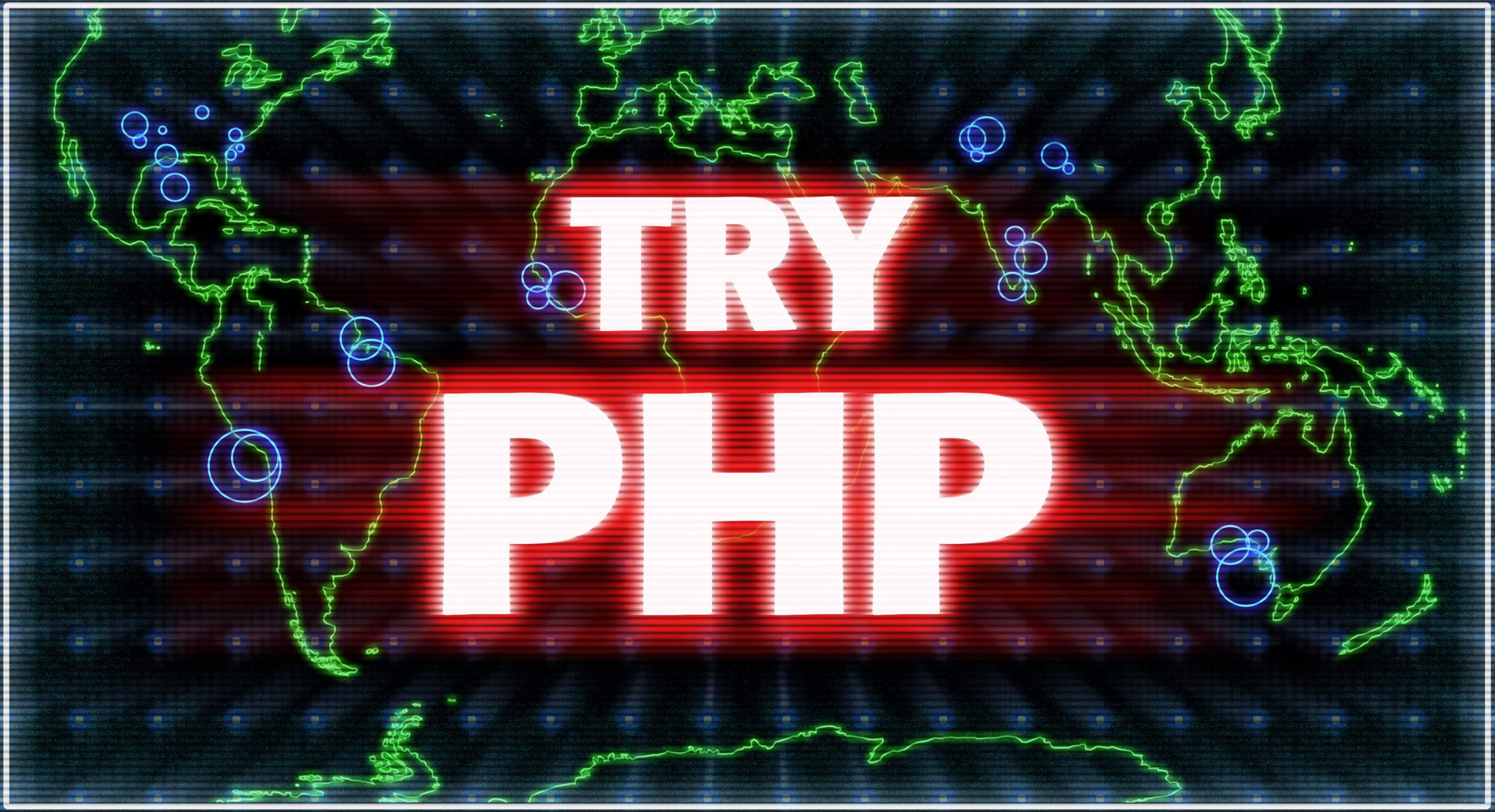
A Complete Picture

```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