

# CSY2028

## Web Programming

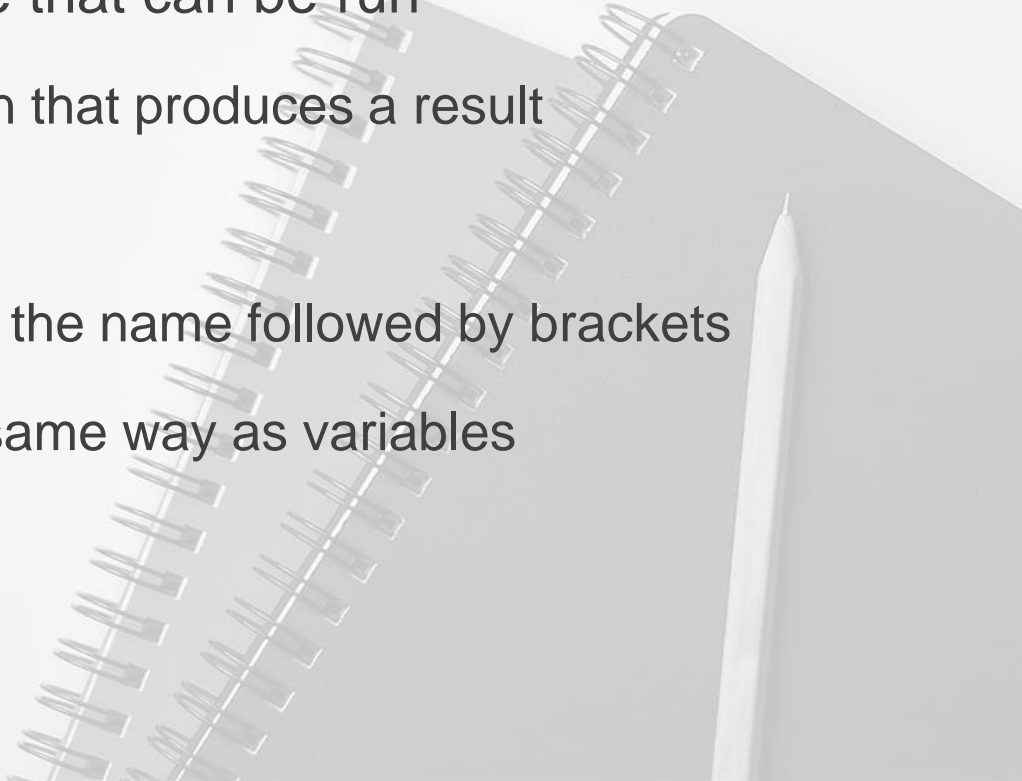


# Topic 4 - Functions

- Inbuilt PHP Functions
- Custom Functions
- Arguments
- Return Values



# PHP Inbuilt Functions

- A function is a piece of code that can be run
  - Usually it will perform an action that produces a result
  - Every function has a name
  - You can run the function using the name followed by brackets
  - Functions can be used in the same way as variables
- 

# PHP Inbuilt Functions

- The rand() function can be used to generate a random number

```
<?php  
echo rand();  
?>
```

```
Output:  
185902316
```

- Each time you run the script it will generate a random number

# PHP Inbuilt Functions

- The pi() function calculates mathematical PI to 14 decimal places

```
<?php  
echo pi();  
?>
```

Output:  
3.1415926535898

# PHP Inbuilt Functions


- Functions can take arguments
- The rand() function will generate a random number from 0 – 2147483647

```
<?php  
echo rand();  
?>
```

Output:  
727321813

- You can give functions arguments these are values which change the way the function works

# PHP Inbuilt Functions

- Each function has arguments that are specific to that function
  - The rand() function optionally takes two arguments:
    - Minimum Value
    - Maximum Value
  - Both arguments are integers
- 

# Arguments

- You can supply a min/max value to the rand function by providing two numbers inside the brackets:

```
<?php  
echo rand(1, 10);  
?>
```

Output:  
6

- Each argument is separated by a comma
- Different functions take different numbers of arguments
- Those arguments are specific to that function



# str\_replace

- The str\_replace function is used to replace part of a string with another string.
- It takes three arguments:
  - A string to look for
  - A string to replace it with
  - The original string

```
<?php  
echo str_replace('NAME', 'Bob', 'Hello NAME');  
?>
```

Output:  
Hello Bob

# Arguments

```
<?php  
echo str_replace('NAME', 'Bob', 'Hello NAME');  
?>
```

Output:  
Hello Bob

- This can be read as:
- Take the string 'Hello NAME'
  - Look for 'NAME' in that string
  - Replace it with 'Bob'

# Date Function

- The date function can be used to display the current date based on the system clock
- It takes a single argument:
  - A specially formatted string that represents a date format.
  - For example: d/m/Y means day/month/year

```
<?php  
echo date('d/m/Y');  
?>
```

Output:  
28/10/2015

# Date Function

- Changing the string supplied to the date function will change its output.
  - H:i is used to display the time in Hours:Minutes format

```
<?php  
echo date('H:i');  
?>
```

Output:  
13:44

- For a full list of supported formats, see:  
<https://www.php.net/manual/en/function.date.php>

# Other Inbuilt Functions

```
<?php  
echo strtoupper('Hello World');  
?>
```

Output:  
HELLO WORLD

```
<?php  
echo strtolower('Hello World');  
?>
```

Output:  
hello world

```
<?php  
echo str_pos('Hello World', 'o');  
?>
```

Output:  
4

```
<?php  
echo strlen('Hello World');  
?>
```

Output:  
11

# Using functions with variables

- Any argument can be replaced with a variable:

```
<?php
$format = 'd/m/Y';
echo date($format);
?>
```

Output:  
28/10/2015

- The result can be stored in a variable and used later

```
<?php
$format = 'd/m/Y';
$result = date($format);
echo '<p>' . $result . '</p>';
?>
```

Output:  
<p>28/10/2015</p>

# Combining arguments

- One or more arguments can be replaced with a variable

```
<?php
$name = 'Bob';
echo str_replace('NAME', $name, 'Hello NAME');
?>
```

Output:  
Hello Bob

# Using functions with variables

- You can store the result of one function in a variable then use it in an argument for another function

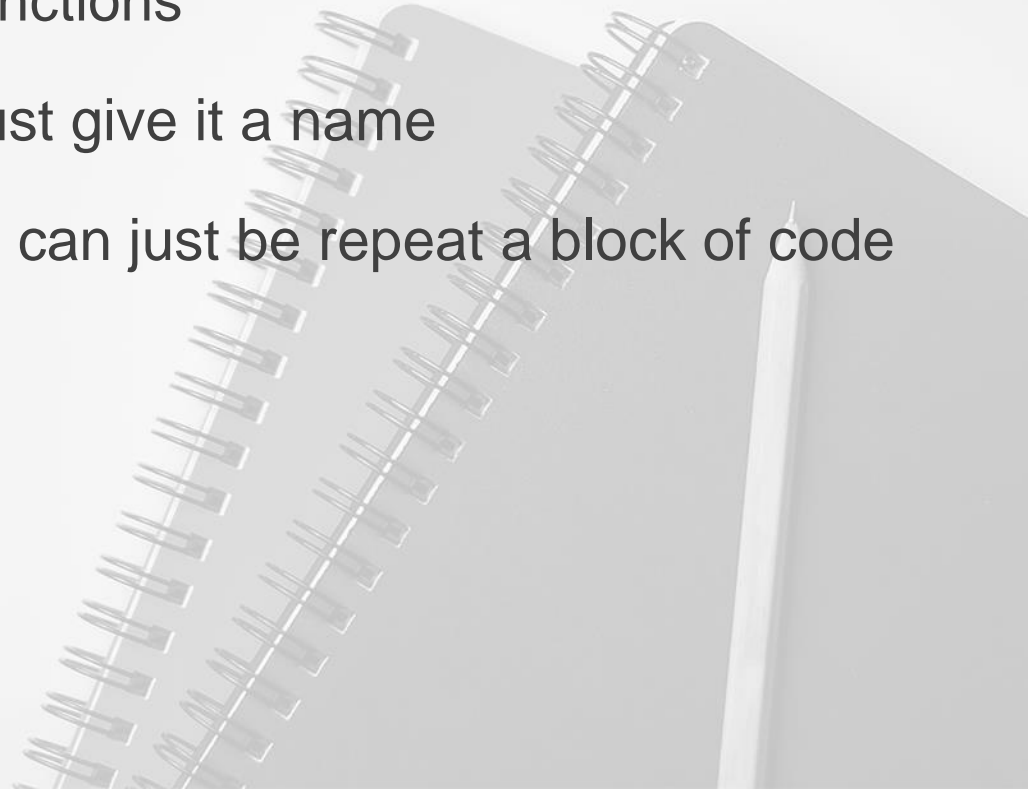
```
<?php
$time = date('H:i');
echo str_replace('{TIME}', $time, 'The time is {TIME}');
?>
```

Output:  
The time is 15:48



# Custom Functions

- You can create your own functions
- To create a function you must give it a name
- At its most basic, a function can just be repeat a block of code throughout your program



# Custom Functions

- A custom function can be declared using the function keyword

```
<?php  
function name() {  
    //Code here  
}  
?>
```

# Custom Functions

- Functions can contain one or more lines of code:

```
<?php
function hello() {
    echo '<p>Hello World</p>';
}
?>
```

# Custom Functions

- Declaring a function alone will have no effect
- After the function is defined, you must call it
- You can call a custom function like an inbuilt function

```
<?php  
function hello() {  
    echo '<p>Hello World</p>';  
}  
  
hello();  
?>
```

Output:  
<p>Hello World</p>

# Custom Functions

- You can call a function as many times as you like once it's been defined:

```
<?php
function hello() {
    echo '<p>Hello World</p>';
}

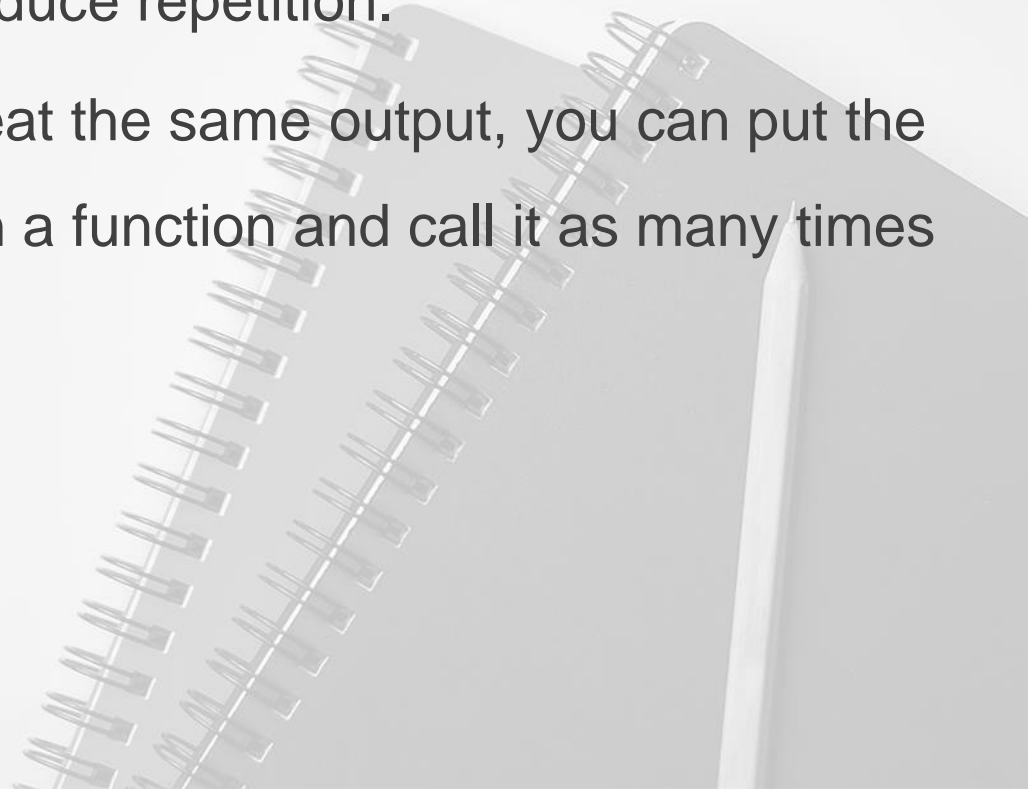
hello();
hello();
hello();
?>
```

Output:

```
<p>Hello World</p>
<p>Hello World</p>
<p>Hello World</p>
```

# Custom Functions

- You can use functions to reduce repetition.
- Anywhere you need to repeat the same output, you can put the HTML you want to repeat in a function and call it as many times as you like



# Custom Functions

```
<?php
function navigation() {
    echo '<ul>';
    echo '  <li><a href="index.php">Home</a></li>';
    echo '  <li><a href="about.php">About</a></li>';
    echo '  <li><a href="contact.php">Contact</a></li>';
    echo '</ul>';
}

?>
<!DOCTYPE html>
<html>
  <head>
    <title>My Site</title>
  </head>
  <body>
    <nav>
      <?php navigation(); ?>
    </nav>
    <main>
      <p>Lorem ipsum...</p>
    </main>
    <footer>
      <?php navigation(); ?>
    </footer>
  </body>
</html>
```

```
<!DOCTYPE html>
<html>
<head>
  <title>My Site</title>
</head>
<body>
  <nav>
    <ul>
      <li><a href="index.php">Home</a></li>
      <li><a href="about.php">About</a></li>
      <li><a href="contact.php">Contact</a></li>
    </ul>
  </nav>
  <main>
    <p>Lorem ipsum...</p>
  </main>
  <footer>
    <ul>
      <li><a href="index.php">Home</a></li>
      <li><a href="about.php">About</a></li>
      <li><a href="contact.php">Contact</a></li>
    </ul>
  </footer>
</body>
</html>
```

# Program Flow

- When a method is called, the PHP interpreter will jump in position from line to line.
- Normally code is run sequentially

```
<?php  
echo '<p>line 1</p>';  
echo '<p>line 2</p>';  
echo '<p>line 3</p>';  
echo '<p>line 4</p>';  
echo '<p>line 5</p>';  
  
?>
```

Interpreter Position



Output:



# Program Flow

```
<?php
```

```
echo '<p>line 1</p>';  
echo '<p>line 2</p>';  
echo '<p>line 3</p>';  
echo '<p>line 4</p>';  
echo '<p>line 5</p>';
```

```
?>
```

Interpreter Position



Output:  
<p>line 1</p>

# Program Flow

```
<?php
```

```
echo '<p>line 1</p>';
```

```
echo '<p>line 2</p>';
```

```
echo '<p>line 3</p>';
```

```
echo '<p>line 4</p>';
```

```
echo '<p>line 5</p>';
```

```
?>
```

Interpreter Position



Output:

```
<p>line 1</p>
```

```
<p>line 2</p>
```

# Program Flow

```
<?php
```

```
echo '<p>line 1</p>';  
echo '<p>line 2</p>';  
echo '<p>line 3</p>';  
echo '<p>line 4</p>';  
echo '<p>line 5</p>';
```

```
?>
```

Interpreter Position



Output:

```
<p>line 1</p>  
<p>line 2</p>  
<p>line 3</p>
```

# Program Flow

```
<?php  
echo '<p>line 1</p>';  
echo '<p>line 2</p>';  
echo '<p>line 3</p>';  
echo '<p>line 4</p>';  
echo '<p>line 5</p>';  
?>
```

Interpreter Position

Output:  
<p>line 1</p>  
<p>line 2</p>  
<p>line 3</p>  
<p>line 4</p>

# Program Flow

```
<?php  
echo '<p>line 1</p>';  
echo '<p>line 2</p>';  
echo '<p>line 3</p>';  
echo '<p>line 4</p>';  
echo '<p>line 5</p>';  
?>
```

Interpreter Position



Output:  
<p>line 1</p>  
<p>line 2</p>  
<p>line 3</p>  
<p>line 4</p>  
<p>line 5</p>

# Program Flow

- When using functions the program flow jumps around. It is not linear in the same way

```
<?php  
  
function myFunction() {  
    echo '<p>myFunction line 1</p>';  
    echo '<p>myFunction line 2</p>';  
}  
  
echo '<p>line 1</p>';  
echo '<p>line 2</p>';  
myFunction();  
echo '<p>line 3</p>';  
echo '<p>line 4</p>';  
echo '<p>line 5</p>';  
  
?>
```

Interpreter Position

Output:

# Program Flow

```
<?php  
function myFunction() {  
    echo '<p>myFunction line 1</p>';  
    echo '<p>myFunction line 2</p>';  
}  
  
echo '<p>line 1</p>';  
echo '<p>line 2</p>';  
myFunction();  
echo '<p>line 3</p>';  
echo '<p>line 4</p>';  
echo '<p>line 5</p>';  
?>
```

Interpreter Position

Output:

# Program Flow

```
<?php  
function myFunction() {  
    echo '<p>myFunction line 1</p>';  
    echo '<p>myFunction line 2</p>';  
}  
  
echo '<p>line 1</p>';  
echo '<p>line 2</p>';  
myFunction();  
echo '<p>line 3</p>';  
echo '<p>line 4</p>';  
echo '<p>line 5</p>';  
?>
```

Interpreter Position

Output:



# Program Flow

```
<?php  
  
function myFunction() {  
    echo '<p>myFunction line 1</p>';  
    echo '<p>myFunction line 2</p>';  
}  
  
echo '<p>line 1</p>';  
echo '<p>line 2</p>';  
myFunction();  
echo '<p>line 3</p>';  
echo '<p>line 4</p>';  
echo '<p>line 5</p>';  
  
?>
```

Interpreter Position

Output:  
<p>line 1</p>

# Program Flow

```
<?php  
function myFunction() {  
    echo '<p>myFunction line 1</p>';  
    echo '<p>myFunction line 2</p>';  
}  
  
echo '<p>line 1</p>';  
echo '<p>line 2</p>';  
myFunction();  
echo '<p>line 3</p>';  
echo '<p>line 4</p>';  
echo '<p>line 5</p>';  
?>
```

Interpreter Position

Output:  
<p>line 1</p>  
<p>line 2</p>

# Program Flow

```
<?php  
  
function myFunction() {  
    echo '<p>myFunction line 1</p>';  
    echo '<p>myFunction line 2</p>';  
}  
  
echo '<p>line 1</p>';  
echo '<p>line 2</p>';  
myFunction();  
echo '<p>line 3</p>';  
echo '<p>line 4</p>';  
echo '<p>line 5</p>';  
  
?>
```

Interpreter Position

Output:  
<p>line 1</p>  
<p>line 2</p>

# Program Flow

```
<?php  
  
function myFunction() {  
    echo '<p>myFunction line 1</p>';  
    echo '<p>myFunction line 2</p>';  
}  
  
echo '<p>line 1</p>';  
echo '<p>line 2</p>';  
myFunction();  
echo '<p>line 3</p>';  
echo '<p>line 4</p>';  
echo '<p>line 5</p>';  
  
?>
```

Interpreter Position

Output:  
<p>line 1</p>  
<p>line 2</p>  
<p>myFunction line 1</p>

# Program Flow

```
<?php  
function myFunction() {  
    echo '<p>myFunction line 1</p>';  
    echo '<p>myFunction line 2</p>';  
}  
  
echo '<p>line 1</p>';  
echo '<p>line 2</p>';  
myFunction();  
echo '<p>line 3</p>';  
echo '<p>line 4</p>';  
echo '<p>line 5</p>';  
  
?>
```

Interpreter Position

Output:

```
<p>line 1</p>  
<p>line 2</p>  
<p>myFunction line 1</p>  
<p>myFunction line 2</p>
```

# Program Flow

```
<?php
function myFunction() {
    echo '<p>myFunction line 1</p>';
    echo '<p>myFunction line 2</p>';
}

echo '<p>line 1</p>';
echo '<p>line 2</p>';
myFunction();
echo '<p>line 3</p>';
echo '<p>line 4</p>';
echo '<p>line 5</p>';

?>
```

Interpreter Position

Output:

```
<p>line 1</p>
<p>line 2</p>
<p>myFunction line 1</p>
<p>myFunction line 2</p>
<p>line 3</p>
```

# Program Flow

```
<?php
function myFunction() {
    echo '<p>myFunction line 1</p>';
    echo '<p>myFunction line 2</p>';
}

echo '<p>line 1</p>';
echo '<p>line 2</p>';
myFunction();
echo '<p>line 3</p>';
echo '<p>line 4</p>';
echo '<p>line 5</p>';

?>
```

Interpreter Position



Output:

```
<p>line 1</p>
<p>line 2</p>
<p>myFunction line 1</p>
<p>myFunction line 2</p>
<p>line 3</p>
<p>line 4</p>
```

# Program Flow

```
<?php
function myFunction() {
    echo '<p>myFunction line 1</p>';
    echo '<p>myFunction line 2</p>';
}

echo '<p>line 1</p>';
echo '<p>line 2</p>';
myFunction();
echo '<p>line 3</p>';
echo '<p>line 4</p>';
echo '<p>line 5</p>';

?>
```

Interpreter Position

Output:

```
<p>line 1</p>
<p>line 2</p>
<p>myFunction line 1</p>
<p>myFunction line 2</p>
<p>line 3</p>
<p>line 4</p>
<p>line 5</p>
```



# Functions

- You can call a function more than once

```
<?php
function myFunction() {
    echo '<p>myFunction line 1</p>';
    echo '<p>myFunction line 2</p>';
}

echo '<p>line 1</p>';
echo '<p>line 2</p>';
myFunction();
echo '<p>line 3</p>';
echo '<p>line 4</p>';
myFunction();
echo '<p>line 5</p>';

?>
```

Interpreter Position

Output:

# Functions

```
<?php
function myFunction() {
    echo '<p>myFunction line 1</p>';
    echo '<p>myFunction line 2</p>';
}

echo '<p>line 1</p>';
echo '<p>line 2</p>';
myFunction();
echo '<p>line 3</p>';
echo '<p>line 4</p>';
myFunction();
echo '<p>line 5</p>';

?>
```

Interpreter Position

Output:

# Functions

```
<?php
function myFunction() {
    echo '<p>myFunction line 1</p>';
    echo '<p>myFunction line 2</p>';
}

echo '<p>line 1</p>';
echo '<p>line 2</p>';
myFunction();
echo '<p>line 3</p>';
echo '<p>line 4</p>';
myFunction();
echo '<p>line 5</p>';

?>
```

Interpreter Position

Output:

# Functions

```
<?php
function myFunction() {
    echo '<p>myFunction line 1</p>';
    echo '<p>myFunction line 2</p>';
}

echo '<p>line 1</p>';
echo '<p>line 2</p>';
myFunction();
echo '<p>line 3</p>';
echo '<p>line 4</p>';
myFunction();
echo '<p>line 5</p>';

?>
```

Interpreter Position

Output:  
<p>line 1</p>

# Functions

```
<?php
function myFunction() {
    echo '<p>myFunction line 1</p>';
    echo '<p>myFunction line 2</p>';
}

echo '<p>line 1</p>';
echo '<p>line 2</p>';
myFunction();
echo '<p>line 3</p>';
echo '<p>line 4</p>';
myFunction();
echo '<p>line 5</p>';

?>
```

Interpreter Position

Output:  
<p>line 1</p>

# Functions

```
<?php
function myFunction() {
    echo '<p>myFunction line 1</p>';
    echo '<p>myFunction line 2</p>';
}

echo '<p>line 1</p>';
echo '<p>line 2</p>';
myFunction();
echo '<p>line 3</p>';
echo '<p>line 4</p>';
myFunction();
echo '<p>line 5</p>';

?>
```

Interpreter Position

Output:  
<p>line 1</p>  
<p>line 2</p>

# Functions

```
<?php
function myFunction() {
    echo '<p>myFunction line 1</p>';
    echo '<p>myFunction line 2</p>';
}

echo '<p>line 1</p>';
echo '<p>line 2</p>';
myFunction();
echo '<p>line 3</p>';
echo '<p>line 4</p>';
myFunction();
echo '<p>line 5</p>';

?>
```

Interpreter Position

Output:  
<p>line 1</p>  
<p>line 2</p>  
<p>myFunction line 1</p>

# Functions

```
<?php
function myFunction() {
    echo '<p>myFunction line 1</p>';
    echo '<p>myFunction line 2</p>';
}

echo '<p>line 1</p>';
echo '<p>line 2</p>';
myFunction();
echo '<p>line 3</p>';
echo '<p>line 4</p>';
myFunction();
echo '<p>line 5</p>';

?>
```

Interpreter Position

Output:

```
<p>line 1</p>
<p>line 2</p>
<p>myFunction line 1</p>
<p>myFunction line 2</p>
```



# Functions

```
<?php
function myFunction() {
    echo '<p>myFunction line 1</p>';
    echo '<p>myFunction line 2</p>';
}

echo '<p>line 1</p>';
echo '<p>line 2</p>';
myFunction();
echo '<p>line 3</p>';
echo '<p>line 4</p>';
myFunction();
echo '<p>line 5</p>';

?>
```

Interpreter Position

Output:

```
<p>line 1</p>
<p>line 2</p>
<p>myFunction line 1</p>
<p>myFunction line 2</p>
<p>line 3</p>
```

# Functions

```
<?php
function myFunction() {
    echo '<p>myFunction line 1</p>';
    echo '<p>myFunction line 2</p>';
}

echo '<p>line 1</p>';
echo '<p>line 2</p>';
myFunction();
echo '<p>line 3</p>';
echo '<p>line 4</p>';
myFunction();
echo '<p>line 5</p>';

?>
```

Interpreter Position

Output:

```
<p>line 1</p>
<p>line 2</p>
<p>myFunction line 1</p>
<p>myFunction line 2</p>
<p>line 3</p>
<p>line 4</p>
```

# Functions

```
<?php
function myFunction() {
    echo '<p>myFunction line 1</p>';
    echo '<p>myFunction line 2</p>';
}

echo '<p>line 1</p>';
echo '<p>line 2</p>';
myFunction();
echo '<p>line 3</p>';
echo '<p>line 4</p>';
myFunction();
echo '<p>line 5</p>';

?>
```

Interpreter Position

Output:

```
<p>line 1</p>
<p>line 2</p>
<p>myFunction line 1</p>
<p>myFunction line 2</p>
<p>line 3</p>
<p>line 4</p>
```

# Functions

```
<?php
function myFunction() {
    echo '<p>myFunction line 1</p>';
    echo '<p>myFunction line 2</p>';
}

echo '<p>line 1</p>';
echo '<p>line 2</p>';
myFunction();
echo '<p>line 3</p>';
echo '<p>line 4</p>';
myFunction();
echo '<p>line 5</p>';

?>
```

Interpreter Position

Output:

```
<p>line 1</p>
<p>line 2</p>
<p>myFunction line 1</p>
<p>myFunction line 2</p>
<p>line 3</p>
<p>line 4</p>
<p>myFunction line 1</p>
```

# Functions

```
<?php
function myFunction() {
    echo '<p>myFunction line 1</p>';
    echo '<p>myFunction line 2</p>';
}

echo '<p>line 1</p>';
echo '<p>line 2</p>';
myFunction();
echo '<p>line 3</p>';
echo '<p>line 4</p>';
myFunction();
echo '<p>line 5</p>';

?>
```

Interpreter Position

Output:

```
<p>line 1</p>
<p>line 2</p>
<p>myFunction line 1</p>
<p>myFunction line 2</p>
<p>line 3</p>
<p>line 4</p>
<p>myFunction line 1</p>
<p>myFunction line 2</p>
```

# Functions

```
<?php
function myFunction() {
    echo '<p>myFunction line 1</p>';
    echo '<p>myFunction line 2</p>';
}

echo '<p>line 1</p>';
echo '<p>line 2</p>';
myFunction();
echo '<p>line 3</p>';
echo '<p>line 4</p>';
myFunction();
echo '<p>line 5</p>';

?>
```

Interpreter Position

Output:

```
<p>line 1</p>
<p>line 2</p>
<p>myFunction line 1</p>
<p>myFunction line 2</p>
<p>line 3</p>
<p>line 4</p>
<p>myFunction line 1</p>
<p>myFunction line 2</p>
<p>line 5</p>
```

# Exercise 1

1. Write a function called “diceRoll” that generates a random number between 1 and 6 and says “You rolled a [number]” in a `<p>` tag
2. Use the function to roll 5 dice on each page load
3. **Optional** Create a function called drawSquare which creates a HTML element on the page with a random CSS class that selects one of 10 different coloured CSS squares.
  - Hint: You will need to include the css file with a `<link>` tag

# Solutions

```
<?php
function diceRoll() {
    $num = rand(1, 6);
    echo '<p>You rolled a ' . $num . '</p>';
}

diceRoll();
diceRoll();
diceRoll();
diceRoll();
diceRoll();
?>
```

```
<link rel="stylesheet" href="style.css" />

<?php
function drawSquare() {
    $num = rand(1, 10);
    echo '<div class="square" . $num . "></div>';
}

for ($i = 0; $i < 10; $i++) {
    drawSquare();
}
?>
```



# Arguments

- Custom functions can also take arguments
- To do this, you must define your function to take one or more arguments
- An argument is a variable which you can use in the function
- When a function has an argument it must be supplied a value when it is called which will be used as that argument

```
<?php
function hello($name) {
    echo '<p>Hello ' . $name . '</p>';
}

hello('Bob');
hello('Sue');
hello('World');
?>
```

Output:

```
<p>Hello Bob</p>
<p>Hello Sue</p>
<p>Hello World</p>
```

# Arguments

```
<?php
function hello($name) {
    echo '<p>Hello ' . $name . '</p>';
}

hello('Bob');
hello('Sue');
hello('World');
?>
```

Output:

```
<p>Hello Bob</p>
<p>Hello Sue</p>
<p>Hello World</p>
```

Declare a variable name  
in the function header

The variable is then  
Available to use in the code  
of the function

The value of the \$name  
variable will be set to whatever  
Is "passed in" when the  
function is called

# Arguments

```
<?php
function hello($name) {
    echo '<p>Hello ' . $name . '</p>';
}
hello('Bob');
hello('Sue');
hello('World');
?>
```

Output:



Interpreter Position

# Arguments

```
<?php
function hello($name) {
    echo '<p>Hello ' . $name . '</p>';
}
hello('Bob');
hello('Sue');
hello('World');
?>
```

Interpreter Position



Output:

# Arguments

```
<?php
function hello($name) {
    echo '<p>Hello' . $name . '</p>';
}
hello('Bob');
hello('Sue');
hello('World');
?>
```

Interpreter Position



Output:

# Arguments

```
<?php
function hello($name) {
    echo '<p>Hello ' . $name . '</p>';
}
hello('Bob');
hello('Sue');
hello('World');
?>
```



Interpreter Position

Output:

# Arguments

```
<?php
function hello($name) {
    echo '<p>Hello ' . $name . '</p>';
}
hello('Bob');
hello('Sue');
hello('World');
?>
```

Interpreter Position

\$name is set to 'Bob'

Output:

# Arguments

```
<?php
function hello($name) {
    echo '<p>Hello ' . $name . '</p>';
}
hello('Bob');
hello('Sue');
hello('World');
?>
```

Interpreter Position

\$name is set to 'Bob'

Output:

<p>Hello Bob</p>



# Arguments

```
<?php
function hello($name) {
    echo '<p>Hello ' . $name . '</p>';
}
hello('Bob');
hello('Sue');
hello('World');
?>
```



Interpreter Position

Output:  
<p>Hello Bob</p>

# Arguments

```
<?php
function hello($name) {
    echo '<p>Hello ' . $name . '</p>';
}
hello('Bob');
hello('Sue');
hello('World');
?>
```

Output:  
<p>Hello Bob</p>



Interpreter Position

\$name is set to 'Sue'

# Arguments

```
<?php
function hello($name) {
    echo '<p>Hello ' . $name . '</p>';
}
hello('Bob');
hello('Sue');
hello('World');
?>
```

Output:  
<p>Hello Bob</p>  
<p>Hello Sue</p>



Interpreter Position

\$name is set to 'Sue'

# Arguments

```
<?php
function hello($name) {
    echo '<p>Hello ' . $name . '</p>';
}
hello('Bob');
hello('Sue');
hello('World');
?>
```

Output:  
<p>Hello Bob</p>  
<p>Hello Sue</p>



Interpreter Position

# Arguments

```
<?php
function hello($name) {
    echo '<p>Hello ' . $name . '</p>';
}
hello('Bob');
hello('Sue');
hello('World');
?>
```

Interpreter Position

\$name is set to 'World'

Output:

```
<p>Hello Bob</p>
<p>Hello Sue</p>
```

# Arguments

```
<?php
function hello($name) {
    echo '<p>Hello ' . $name . '</p>';
}
hello('Bob');
hello('Sue');
hello('World');
?>
```

Interpreter Position

\$name is set to 'World'

Output:

```
<p>Hello Bob</p>
<p>Hello Sue</p>
<p>Hello World</p>
```

# Arguments

- A function can have more than one argument

```
<?php
function add($num1, $num2) {
    $result = $num1 + $num2;

    echo $num1 . ' + ' . $num2 . ' = ' . $result;
}

add(5, 10);
add(99, 100);
?>
```

Output:  
5 + 10 = 15  
99 + 100 = 199

# Return Values

- With inbuilt functions, you call them and use their result. For example:

```
<?php
$format = 'd/m/Y';
$result = date($format);
echo '<p>' . $result . '</p>';
?>
```

- This is called returning a value
- The date function returns a value which can then be stored inside a variable (here: \$result)



# Return Values

- Custom functions can also return values. Currently the add function prints directly to the screen:

```
<?php
function add($num1, $num2) {
    $result = $num1 + $num2;

    echo $num1 . ' + ' . $num2 . ' = ' . $result;
}

add(5, 10);
add(99, 100);

?>
```

# Return Values

- If the echo is removed from the function there is no way to access the result:

```
<?php  
  
function add($num1, $num2) {  
    $result = $num1 + $num2;  
}  
  
add(5, 10);  
add(99, 100);  
  
?>
```

# Return Values

- However, you can return a value which can be accessed where the function was called

```
<?php
function add($num1, $num2) {
    $result = $num1 + $num2;
    return $result;
}

$result1 = add(5, 10);
$result2 = add(99, 100);

echo '<p>Result 1 is ' . $result1 . '</p>';
echo '<p>Result 2 is ' . $result2 . '</p>';
?>
```

**Output:**  
Result 1 is 15  
Result 2 is 199

# Functions

- It's usually a bad idea to echo from within a function as you might want to do something else with the result (Write it to a file ,write it to a database, etc) rather than displaying it on the screen
- Return a value rather than printing it to the screen

```
<?php
function add($num1, $num2) {
    $result = $num1 + $num2;
    echo $result;
}

add(5, 10);
?>
```



```
<?php
function add($num1, $num2) {
    $result = $num1 + $num2;
    return $result;
}

echo add(5, 10);
?>
```



## Exercise 2

1. Write a function called `p` which generates a paragraph:

```
echo p('Paragraph 1');  
echo p('Paragraph 2');
```

Output:

```
<p>Paragraph 1</p>  
<p>Paragraph 2</p>
```

2. Rename the `p` function `html` and make it take two arguments:

```
echo html('h1', 'Heading');  
echo html('p', 'Paragraph 1');  
echo html('li', 'List item 1');  
echo html('li', 'List item 2');
```

Output:

```
<h1>Heading</h1>  
<p>Paragraph 1</p>  
<li>List item 1</li>  
<li>List item 2</li>
```

# Exercise 2

3. Adjust your code from exercise so that valid HTML is produced and list items appear inside `<ul>` tags

```
Output:  
<h1>Heading</h1>  
<p>Paragraph 1</p>  
<ul>  
<li>List item 1</li>  
<li>List item 2</li>  
</ul>
```

4. Add a third argument to include a class name for a HTML element

```
echo html('p', 'Paragraph 1', 'red');  
echo html('p', 'Paragraph 2', '');
```



```
Output:  
<p class="red">Paragraph 1</p>  
<p>Paragraph 2</p>
```

# Exercise 2 Solutions

1)

```
<?php
function p($text) {
    return '<p>' . $text . '</p>';
}

echo p('Paragraph 1');
echo p('Paragraph 2');
?>
```

2)

```
<?php
function html($tag, $text) {
    return '<' . $tag . '>' . $text . '</' . $tag . '>';
}

echo html('h1', 'Heading');
echo html('p', 'Paragraph 1');
echo html('li', 'List item 1');
echo html('li', 'List item 2');
?>
```

# Exercise 2 Solutions

3)

```
<?php
function html($tag, $text) {
    return '<' . $tag . '>' . $text . '</' . $tag . '>';
}

echo html('h1', 'Heading');
echo html('p', 'Paragraph 1');

$list = html('li', 'List item 1') . html('li', 'List item 2');

echo html('ul', $list);

?>
```



# Exercise 2 Solutions

4)

```
<?php
function html($tag, $text, $class) {
    if ($class != '') {
        return '<' . $tag . ' class="' . $class . '"' . $text . '</' . $tag . '>';
    }
    else {
        return '<' . $tag . '>' . $text . '</' . $tag . '>';
    }
}

echo html('p', 'Paragraph 1', 'red');
echo html('p', 'Paragraph 2', '');

?>
```

# Optional Exercises

1. Without using a for/while loop or any kind of string repeat function and a single PHP file, print out the text “All work and no play makes jack a dull boy” to the screen 200 times in a list
    - Grade C: 15 lines of code
    - Grade B: 10 lines of code
    - Grade A: 7 lines of code
    - Grade A\*: 4 lines of code (VERY difficult)
- 