For Loop
Used when you know in advance how many times you want to execute a statement or a block of statements.

```
for($x = 0; $x <= 5; $x++) {
    echo "The number is: $x <br>";
}
```

While Loop
It executes a block of code as long as the specified condition is true.

```
$x = 1;
while($x <= 5) {
    echo "The number is: $x <br>";
    $x++;
}
```

Do...While Loop
Similar to the while loop, but the condition is tested after the execution of the loop's statements.

```
$x = 1;
do {
    echo "The number is: $x <br>";
    $x++;
} while ($x <= 5);
```

Foreach Loop
Used for looping through arrays. For each loop iteration, the value of the current array element is assigned to the value variable and the array pointer is moved by one, until it reaches the last array element.

```
$colors = array("red", "green", "blue", "yellow");
foreach ($colors as $value) {
```

```
    echo "$value <br>";
}
```

```
$age = array("Peter" => "35", "Ben" => "37", "Joe" => "43");
foreach($age as $key => $value) {
    echo "$key is $value years old. <br>";
}
```

Break

It is used to exit a loop prematurely. Once break is encountered, the loop terminates and the program resumes execution after the loop.

```
for ($i = 0; $i < 10; $i++) {
    if ($i == 3) {
        break;
    }
    echo $i . "<br>";
}
```

Continue

The continue statement is used to skip the rest of the current loop iteration and proceed to the evaluation of the loop condition for the next iteration.

```
for ($i = 0; $i < 10; $i++) {
    if ($i == 3) {
        continue;
    }
    echo $i . "<br>";
}
```

#php

Functions

- Functions are blocks of code that perform specific tasks and can be reused throughout the program. Functions help with better code organization, reuse, and maintainability.

Advantages of Functions:

1. **Code Reusability**: Functions allow the reuse of code blocks without rewriting them, making development faster and more efficient.
2. **Modular Structure**: Functions help in breaking down complex problems into smaller, more manageable pieces, enhancing code readability.
3. **Ease of Maintenance**: Changes can be made in a single place (the function), and all the parts of the application that use the function will automatically benefit from the changes.
4. **Improved Debugging**: Functions isolate different tasks, making it easier to identify and fix bugs.

Key Points:

- **Static Properties and Methods**: Declared using the `static` keyword and are shared among all instances of the class.
- **Inheritance of Static Properties**: A child class can access static properties and methods from the parent class using `parent::`.
- **Accessing Static Properties**: You do not need to instantiate an object to access static properties or methods; they are accessed using the `::` operator.
- **Method Overloading**: Achieved using `__call()` for instance methods and `__callStatic()` for static methods to handle calls to undefined methods dynamically.
- **Encapsulation**: Control visibility using `private`, `protected`, and `public` access modifiers.
- **Polymorphism**: Implemented by allowing a child class to override a parent class method.
- **Abstraction**: Achieved using abstract classes and interfaces to provide a simplified interface to complex logic.
- **Functions**: Provide code reusability, modular structure, easier maintenance, and improved debugging.

Built-in Functions: These are the standard functions that are always available in PHP without any special requirement like.

strlen(): To get the length of a string.
array_merge(): To merge two or more arrays.
is_numeric(): To check if a variable is a number or a numeric string.

Build In Functions List
https://www.php.net/manual/en/indexes.functions.php

```php
$string = "Hello,World!";
$length = strlen($string);
echo $length;

$value="12Z";
if (is_numeric($value)) {
    echo "The value '$value' is numeric";
}
else {
    echo "The value '$value' is NOT numeric";
}
```

User-defined Functions: These are functions created by the programmer. They're defined using the function keyword.

```php
function sum() {
    $num1=20;
    $num2=30;
    echo $num1+$num2;
}
sum();
```