Loop

1. `while` Loop

- A `while` loop repeats as long as a given condition is `true`.
- Syntax:

```
$i = 1;
while ($i <= 3) {
    echo "$i<br>";
    $i++;
}
```

Alternative syntax with
`endwhile`

- Example:

```
$i = 1;
while ($i <= 3) :
    echo "$i<br>";
    $i++;
endwhile;
```

2. `do...while` Loop

- Runs the code inside the `do` block at least once, then continues while the condition is `true`.
- Example:

```
$i = 4;
do {
    echo "$i<br>";
    $i++;
} while ($i <= 3);
```

3. `for` Loop

- A loop with an initialization, condition check, and increment/decrement in a single line.
- Syntax:

```
for ($i = 1; $i <= 3; $i++) {
    echo "$i<br>";
}
```

Infinite loop with `break`

- Example:

```
$index = 1;
for (;;) {
    if ($index == 4) {
        break;
    }
    echo "$index<br>";
    $index++;
}
```

4. `foreach` Loop

- Iterates over elements in an array.
- Simple Array:

```php
$countries = ["EG", "SA", "QA", "SY"];
foreach ($countries as $country) {
    echo $country . "<br>";
}
```

Associative Array

- Example:

```php
$countries_with_discount = ["EG" => 50, "SA" => 30, "QA" => 50, "SY" => 70];
foreach ($countries_with_discount as $country => $discount) {
    echo "Country Name Is $country And Discount Is $discount <br>";
}
```

5. `break` and `continue`

- `break`: Exits the loop entirely when a condition is met.
- `continue`: Skips the current iteration and continues with the next one.
- Examples:

```php
foreach (["EG", "SA", "QA", "SY", "USA", "GER"] as $country) {
    if ($country == "USA") {
        break; // Stops the loop when country is "USA"
    }
    echo $country . "<br>";
}

foreach (["EG", "SA", "QA", "SY", "USA", "GER"] as $countr
y) {
    if ($country == "USA") {
        continue; // Skips "USA" but continues with the ne
xt iteration
    }
    echo $country . "<br>";
}
```

1. **Include vs. Require:**
   - `include` and `require` are both used to include external PHP files in a script.
   - The difference is that:
     - `include` generates a **warning** if the file is missing and lets the script continue.
     - `require` generates a **fatal error** if the file is missing and stops the script.
2. **include_once:**
   - `include_once` checks if the specified file has already been included in the current script.
   - If the file was already included, `include_once` **does not include it again**. This prevents the risk of re-declaring functions, classes, or variables.

Code Walkthrough

```
include_once("test.php"); // $a = 10;
echo $a . '<br>';
```

- The `include_once("test.php");` statement includes the `test.php` file **once**.
  - Let's assume `test.php` has the line `$a = 10;`.
  - After including it the first time, `$a` will be set to `10`.
- `echo $a . '<br>';` outputs `10`.

```
$a = 20;
include_once("test.php"); // $a = 10;
echo $a. '<br>';
echo "Continue";
```

Here, we assign a new value
`20` to `$a`.

- We call `include_once("test.php");` again, but because the file has already been included, this second call **does nothing**.
- `echo $a . '<br>';` outputs `20` because `$a` remains `20`.
- This line simply outputs `"Continue"` to indicate the script has finished running.

```
10
20
Continue
```

Assuming `test.php` defines `$a = 10;`, the output will be:

The
`include_once` is especially useful for preventing duplicate inclusions, which helps avoid re-declaration errors and ensures consistent variable states across your script.

3-Using
`require_once`

- Just like `include_once`, you can use `require_once` to include a file **only once**. This prevents re-declaring functions, variables, or classes if you try to require the file multiple times.

Example of `require` Usage

Let's say you have a file `config.php` with some important configurations that you need to run your application:

config.php:

```
<?php
$site_name = "My Website";
$db_host = "localhost";
$db_user = "root";
$db_pass = "password";
```

index.php:

```
<?php
require("config.php");

echo "Welcome to ". $site_name;
```

In this example:

1. If `config.php` is present, it will be loaded, and `$site_name` will be set to "My Website."
2. If `config.php` is missing, a fatal error will occur, and the script will **not proceed**, preventing potential issues from missing configuration data.

Assuming
`config.php` exists:

```
Welcome to My Website
```

If
`config.php` is **missing**:

```
Fatal error: require(): Failed opening required 'config.ph
p'...
```

When to Use `require` vs. `include`

- Use `*require**` for critical files that the script **cannot run without** (e.g., database configurations or essential libraries).
- Use `*include**` for files that are optional or non-essential. This way, even if they are missing, the rest of the script can continue running.