Ways to Write Code in PHP:

- **Procedural Programming**

أول نوع ظهر من أنواع البرمجة، وأساسها هو تجميع الأوامر أو الإجراءات في خطوات متسلسلة لحل مشكلة معينة، وتعتمد على الوظائف (Functions) والإجراءات (Procedures). كل وظيفة تكون مسؤولة عن جزء معين من الكود.

فكرة عامة :
كأنك عندك قائمة خطوات لحل مسألة معينة، تبدأ من الخطوة الأولى إلى النهاية. كل خطوة لها دور واضح ، وتُنفّذ خطوة بعد خطوة.
البيانات هنا تعتبر ثابتة وأي تغييرات عليها تتم داخل الوظائف.

مثال عملي :
نفترض أنك تريد كتابة برنامج يحسب مجموع قيم معينة:

```php
function calculateSum($numbers) {
    $sum = 0;
    foreach ($numbers as $number) {
        $sum += $number;
    }
    return $sum;
}
```

هنا ، كل خطوة واضحة : نمر على القيم ونحسب المجموع.

مميزات وعيوب:
السهولة : سهل في الفهم والتتبع.
التكرار : معقد بعض الشيء في البرامج الكبيرة لأن أي تغيير قد يتطلب تعديلات في أكثر من مكان.

- **Functional Programming**

تركز البرمجة الوظيفية على كتابة وظائف (Functions) مستقلة عن بعضها، بحيث لا تعتمد على التغيرات الخارجية، وتكون خالية من التأثيرات الجانبية (Side Effects). كل وظيفة تأخذ مدخلات وتنتج مخرجات دون التأثير على الحالة العامة للبرنامج.

الفكرة العامة :
هنا الأمر يشبه العمليات الرياضية؛ الوظيفة تأخذ قيمًا وتعيد قيمة دون تعديل أي شيء خارجي.
البيانات تعتبر ثابتة (Immutable) ما يعني أنها لا تتغير أثناء التنفيذ، وهذا يجعل البرنامج أكثر استقرارًا وسهولة في التصحيح.

مثال عملي :
فيما يلي مثال بسيط لحساب مجموع قيم معينة بطريقة برمجة وظيفية

```php

$numbers = [1, 2, 3, 4];
$sum = array_reduce($numbers, fn($carry, $item) => $carry + $item, 0);

```

هنا، array_reduce تقوم بتطبيق دالة جمع على كل عنصر من القائمة، ولا توجد تغييرات على البيانات الأصلية.

مميزات وعيوب:
الوضوح والمرونة : الوظائف تكون مستقلة ويمكن إعادة استخدامها بسهولة.
التعقيد في المشاريع الكبيرة : البرمجة الوظيفية قد تكون صعبة الفهم في بعض الحالات لأنها تركز على الدوال المستقلة فقط.

- **Object-Oriented Programming**

أما البرمجة الشيئية فهي طريقة مختلفة كليًا، حيث تعتبر كل كائن (Object) ممثلًا لجزء معين من البرنامج، وكل كائن لديه بيانات (Attributes) ووظائف (Methods) خاصة به. الكائنات تتفاعل مع بعضها البعض عبر الوظائف.

فكرة عامة:
البرمجة الشيئية تحاكي العالم الحقيقي ؛ كل كائن يمثل جزءًا حقيقيًا له خصائص ومهام .
تعتمد على المفاهيم الأساسية مثل:

- **التغليف (Encapsulation) : جعل الكائن يحتفظ ببياناته ووظائفه.**
- **التوريث (Inheritance) : إعادة استخدام الكائنات بطرق مختلفة.**
- **تعدد الأشكال (Polymorphism) : استخدام الكائنات بشكل متعدد عبر واجهات متعددة.**
- **التجريد (Abstraction): إخفاء التفاصيل المعقدة للكائنات وتوفير واجهة بسيطة للتفاعل معها.**

مثال عملي:
نريد إنشاء كائن يمثل مستخدم في النظام:

```php
class User {
    private $name;

    public function __construct($name) {
        $this->name = $name;
    }

    public function greet() {
        return "Hello, " . $this->name;
    }
}

$user = new User("Ahmed");
echo $user->greet();  // Output: Hello, Ahmed
```

– هنا، الـclass User يحتوي على name ودالة greet ، وهو منظم ويحاكي فكرة المستخدم.

ميزات وعيوب:

- **المرونة وإعادة الاستخدام : يمكنك بناء الكود بشكل مرن وقابل للتطوير.**
- **التعقيد : قد يكون معقدًا لبعض المبتدئين، حيث يعتمد على العديد من المفاهيم.**