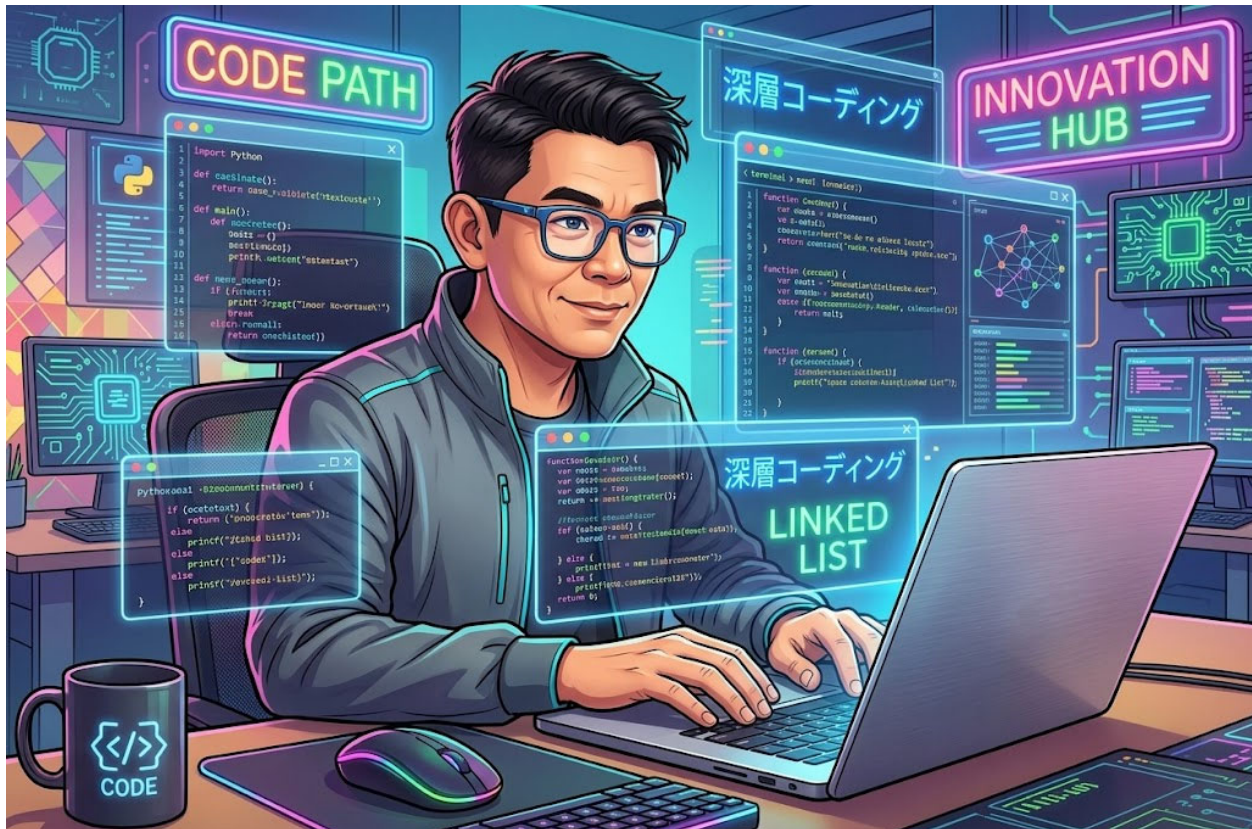


การเขียนโค้ด (Coding)



การเขียนโค้ด (Coding) คือการเขียนชุดคำสั่งหรือภาษาคอมพิวเตอร์เพื่อบอกให้ระบบทำงานตามที่เราต้องการ และที่ขาดไม่ได้คือ การเขียนโค้ดแบบทำงานเป็นทีม หรือการเขียนโค้ดสำหรับนักพัฒนาคนเดียว มีความแตกต่าง และสำคัญเพียงใด

โครงสร้างโค้ดและหลักเกณฑ์การตั้งชื่อ

การจัดโครงสร้างโค้ดและหลักเกณฑ์การตั้งชื่อ (Naming Conventions) เป็นรากฐานสำคัญที่ช่วยให้โค้ดอ่านง่าย ดูแลรักษาง่าย (Maintainability) และทำงานร่วมกันเป็นทีมได้อย่างมีประสิทธิภาพ ตามแนวทางปฏิบัติสากล ดังนี้

หลักเกณฑ์การตั้งชื่อ (Naming Conventions)

การตั้งชื่อที่ดีควร "สื่อความหมายได้ทันทีว่าสิ่งนั้นคืออะไร และทำอะไร" โดยไม่ต้องมานั่งแกะโค้ดทีหลัง

- **ตัวแปร (Variable):**
 - ใช้คำนามหรือคำคุณศัพท์ที่สื่อความหมายชัดเจน
 - หลีกเลี่ยงตัวย่อสั้นๆ ที่เข้าใจได้ยาก เช่น ใช้ customerCount แทน cc หรือ c
- **ฟังก์ชัน / เมธอด (Function / Method):**
 - เริ่มต้นด้วยคำกริยา (Verb) เสมอ เพื่อบอกถึงการกระทำ เช่น getUserData(), calculateTotalPrice(), saveCustomer()
- **คลาส / อินเตอร์เฟซ (Class / Interface):**
 - ใช้คำนาม (Noun) ที่เป็นเอกพจน์ และขึ้นต้นด้วยตัวพิมพ์ใหญ่ เช่น Customer, OrderService, PaymentGateway
- **รูปแบบการเขียนตัวพิมพ์ (Case Styles):**
 - **camelCase:** ตัวแรกพิมพ์เล็ก คำถัดไปขึ้นต้นด้วยพิมพ์ใหญ่ (นิยมใช้กับตัวแปรและฟังก์ชันทั่วไป เช่น userProfile, getTotalAmount)
 - **PascalCase:** ทุกคำขึ้นต้นด้วยพิมพ์ใหญ่ (นิยมใช้กับชื่อ Class, Interface หรือ Component)
 - **snake_case:** ใช้ตัวพิมพ์เล็กทั้งหมดและคั่นด้วยขีดล่าง _ (นิยมใช้ใน Python, Database หรือตั้งชื่อไฟล์บางประเภท)
 - **kebab-case:** ใช้ตัวพิมพ์เล็กทั้งหมดและคั่นด้วยยัติภังค์ - (นิยมใช้ใน URL, HTML/CSS หรือชื่อไฟล์)

- UPPER_SNAKE_CASE: ตัวพิมพ์ใหญ่ทั้งหมดคั่นด้วยขีดล่าง (นิยมใช้กับค่าคงที่ / Constants เช่น MAX_RETRY_COUNT, API_TIMEOUT_MS)

โครงสร้างโค้ดและการจัดระเบียบ (Code Structure)

การจัดระเบียบไฟล์ โมดูล และชุดคำสั่ง ช่วยให้โครงสร้างโปรเจกต์เป็นระเบียบและไม่ซับซ้อนจนเกินไป

- แยกความรับผิดชอบ (Separation of Concerns): แบ่งโค้ดออกเป็นส่วนๆ ตามหน้าที่ เช่น ส่วนติดต่อผู้ใช้ (UI), ส่วนจัดการข้อมูล (Business Logic / Service), และส่วนเชื่อมต่อฐานข้อมูล (Repository / Database)
- จัดระเบียบโฟลเดอร์ตาม Feature หรือ Layer:
 - จัดกลุ่มไฟล์ที่เกี่ยวข้องกันไว้ด้วยกัน หรือแยกตามประเภทของไฟล์ (Controllers, Models, Views, Utils) ให้เป็นมาตรฐานเดียวกันทั้งทีม
- ความยาวของฟังก์ชันและคลาส:
 - ฟังก์ชันหนึ่งควรทำหน้าที่เพียงอย่างเดียว (Single Responsibility Principle) และมีความยาวไม่ควรเกินหลักสิบถึงร้อยบรรทัด เพื่อให้ทดสอบ (Unit Test) และอ่านง่าย
- การจัดการคอมเมนต์ (Comments):
 - เขียนคอมเมนต์เพื่ออธิบาย "ทำไม (Why)" ถึงเขียนแบบนี้ มากกว่าอธิบาย "ทำอะไร (What)" เพราะตัวโค้ดเองควรสื่อความหมายด้วยตัวมันเองอยู่แล้ว

การเขียนโค้ดแบบทำงานเป็นทีม หรือการเขียนโค้ดสำหรับนักพัฒนาคนเดียว มีความแตกต่าง และสำคัญเพียงใด การเขียนโค้ดแบบทำงานเป็นทีมมีความซับซ้อนและเน้นมาตรฐานการสื่อสารสูงกว่าการเขียนโค้ดคนเดียว เพราะต้องใช้ระบบจัดการเวอร์ชัน เช่น Git เพื่อรวมโค้ดและลดข้อขัดแย้ง (Merge Conflict) ขณะที่การเขียนโค้ดคนเดียวมีอิสระสูงกว่าและตัดสินใจทุกอย่างได้ด้วยตนเอง

ความแตกต่างหลักระหว่างการเขียนโค้ดเป็นทีม กับเขียนคนเดียว

1. การเขียนโค้ดสำหรับนักพัฒนาเดี่ยว (Solo Developer)

- **อิสระในการตัดสินใจ:** เลือกใช้ภาษา, โครงสร้างโปรเจกต์ (Project Structure) หรือรูปแบบการเขียนได้ตามใจชอบโดยไม่ต้องขอความเห็นใคร
- **ความเร็วในการเริ่มงาน:** ไม่เสียเวลาในการประชุม หรือกำหนดข้อตกลงร่วมกับคนอื่น (Coding Standards) สามารถเขียนและแก้ปัญหาได้ทันที
- **ข้อจำกัด:**
 - แกรับความเสี่ยงคนเดียว (ถ้าป่วย งานหยุด)
 - มุมมองในการแก้ปัญหาอาจแคบลง เพราะมีแค่สมองคนเดียว
 - ถ้าโค้ดไม่เป็นระเบียบ พอทิ้งไว้กลับมาอ่านเองในอนาคตจะแก้ยากมาก

2. การเขียนโค้ดแบบทำงานเป็นทีม (Team Development)

- **มาตรฐานเดียวกัน (Coding Standards):** ต้องมีการกำหนดกฎเกณฑ์ร่วมกัน เช่น การตั้งชื่อตัวแปร, การจัดรูปแบบโค้ด (Format) เพื่อให้ทุกคนอ่านโค้ดของกันและกันได้
- **การใช้เครื่องมือควบคุมเวอร์ชัน:** ต้องใช้ Git ร่วมกับแพลตฟอร์มอย่าง GitHub หรือ GitLab เพื่อแยก Branch, ทำ Code Review และตรวจสอบความถูกต้องก่อนรวมโค้ด
- **การแบ่งงานตามความเชี่ยวชาญ:** แยกส่วนงาน เช่น คนทำ Frontend, Backend, Database หรือ DevOps ทำงานประสานกันผ่าน API

- **ความสำคัญด้านทักษะสังคม:** ต้องสื่อสารให้เข้าใจตรงกัน คุยภาษามนุษย์รู้เรื่อง และจัดการปัญหาความขัดแย้งในเชิงเทคนิคได้

ตัวอย่างโครงสร้าง

ภาษา PHP

```
<?php
```

```
// ส่วนของการประกาศตัวแปรและประมวลผล
```

```
$name = "Somchai";
```

```
$age = 25;
```

```
// คำสั่งแสดงผลออกทางหน้าจอ
```

```
echo "Hello, " . $name . "<br>";
```

```
echo "Age: " . $age;
```

```
f ($age >= 18) {
```

```
    echo "คุณเป็นผู้ใหญ่แล้วนะ";
```

```
} else {
```

```
    echo "คุณยังเป็นเด็กอยู่";
```

```
}
```

```
?>
```

ภาษา python

ส่วนของการประกาศตัวแปรและประมวลผล

name = "Somchai"

age = 25

คำสั่งแสดงผลออกทางหน้าจอ

print(f"Hello, {name}
")

print(f"Age: {age}")

if age >= 18:

print("คุณเป็นผู้ใหญ่แล้วนะ")

else:

print("คุณยังเป็นเด็กอยู่")