ฝึกปฏิบัติการเขียนโปรแกรมคอมพิวเตอร์

Programming Practice

ผู้ช่วยศาสตราจารย์สมเกียรติ ช่อเหมือน (tko@webmail.npru.ac.th)

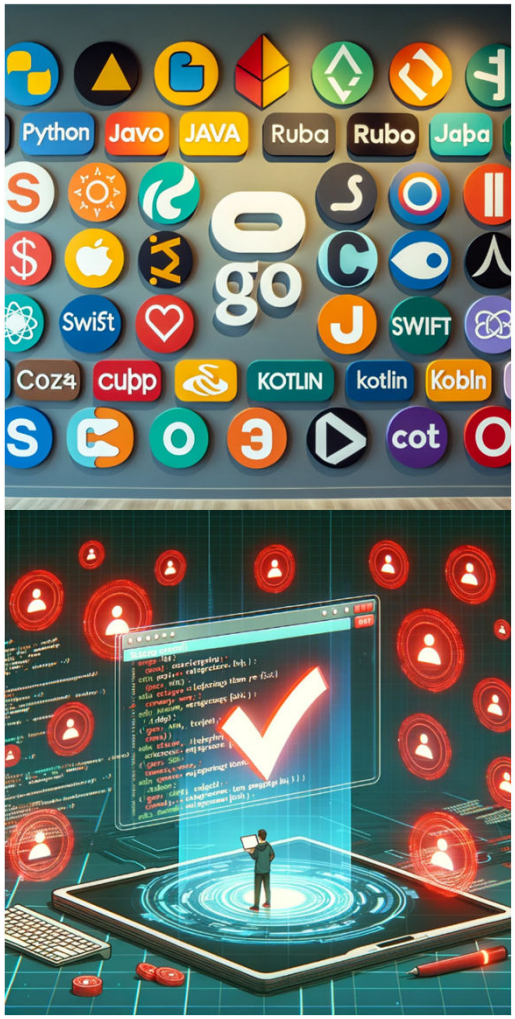
สาขาวิชาวิศวกรรมซอฟต์แวร์ คณะวิทยาศาสตร์และเทคโนโลยี

บทนำ

- การฝึกปฏิบัติเขียนโปรแกรม
- แนวคิดในการทดสอบโปรแกรม
  - การหาข้อผิดพลาดหรือบั๊ก (bugs)
  - การทดสอบหน่วย (Unit Testing)
  - การทดสอบระบบ (System Testing)
  - การทดสอบยอมรับ (Acceptance Testing)
- การเลือกข้อมูลทดสอบ (Test Data Selection)
  - ความต้องการและฟังก์ชันของโปรแกรม
  - การสร้าง Test Cases
- วิธีการทดสอบ: การทดสอบแบบ Black-box และ White-box
  - การวิเคราะห์และอธิบายโปรแกรม

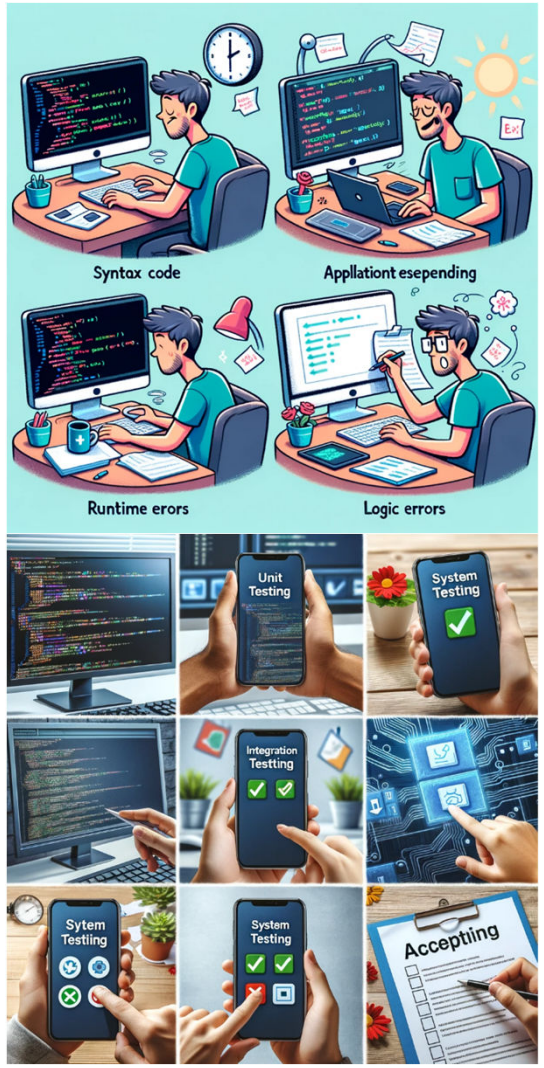
การฝึกปฏิบัติเขียนโปรแกรม

1. เรียนรู้ภาษาโปรแกรมมิ่ง
2. ทำความเข้าใจโจทย์
3. การออกแบบและวางแผน
4. เขียนโค้ด
5. ทดสอบและแก้ไขข้อผิดพลาด
6. รับรองและประเมินผล



แนวคิดในการทดสอบโปรแกรม

- **หาข้อผิดพลาด (Programming Errors)**
  - ข้อผิดพลาดของไวยากรณ์ (Syntax Errors)
  - ข้อผิดพลาดของการทำงาน (Runtime Errors)
  - ข้อผิดพลาดของตรรกะ (Logic Errors)
- **ตรวจสอบการทำงานทุกระยะของแต่ละส่วน**
  - การทดสอบระบบย่อย (Unit Testing)
  - การทดสอบการทำงานร่วมกัน (Integration Testing)
  - การทดสอบระบบ (System Testing)
  - การทดสอบการยอมรับ (Acceptance Testing)



การหาข้อผิดพลาดหรือบั๊ก (bugs)

- การตรวจสอบโค้ดอย่างละเอียด
  - การใช้งานตัวแปรไม่ถูกต้อง, โค้ดที่ไม่ถูกต้อง, หรือการใช้งานฟังก์ชันผิด
- การทดสอบโปรแกรมด้วยข้อมูลทดสอบที่หลากหลาย
  - การทดสอบรับค่านำเข้าและผลลัพธ์
- การใช้เครื่องมือตรวจสอบโค้ด
  - การตรวจจับข้อผิดพลาดและการแจ้งเตือน
- การใช้ Debuggers
  - ตรวจสอบโค้ดขณะทำงาน ตรวจสอบค่าของตัวแปร และหาสาเหตุข้อผิดพลาด

การเลือกข้อมูลทดสอบ

- โปรแกรมทำงานได้ตามความต้องการ
- แน่ใจว่าโปรแกรมทำงานได้ในสถานการณ์ต่าง ๆ

"มุมมอง" (Perspective) และ "ประสบการณ์" (Experience)

ความต้องการและฟังก์ชันของโปรแกรม

- **ความต้องการ (Requirements)**:
  - ข้อกำหนดหรือเงื่อนไขที่ทำได้
    - ความต้องการธุรกิจ (Business Requirements)
    - ความต้องการเชิงเทคนิค (Technical Requirements)
  - การสื่อสารระหว่างผู้มีส่วนได้เสียกับทีมพัฒนา
    - รับรู้ความต้องการและความคาดหวัง
- **ฟังก์ชันของโปรแกรม (Functionality)**:
  - การทำงานที่เฉพาะเจาะจงของผู้ใช้งาน
  - "โปรแกรมทำอะไรได้บ้าง"
    - แผนภาพการทำงาน (Flowchart)
    - การระบุข้อกำหนดฟังก์ชัน (Functional Specification)

การสร้างกรณีทดสอบ (Test Cases)

- **การวิเคราะห์ความต้องการ (Requirement Analysis)**
- **การกำหนดขอบเขตการทดสอบ (Test Scope Definition**
- **การออกแบบกรณีทดสอบ (Test Case Design)**:
  - ระบุเงื่อนไขการทดสอบ (Test Conditions)
  - กำหนดข้อมูลนำเข้า (Input Data)
  - กำหนดผลลัพธ์ที่คาดหวัง (Expected Results)
- **การเขียนขั้นตอนการทดสอบ (Test Procedure Specification**
- **การตรวจสอบกรณีทดสอบ (Test Case Review)**

วิธีการทดสอบ

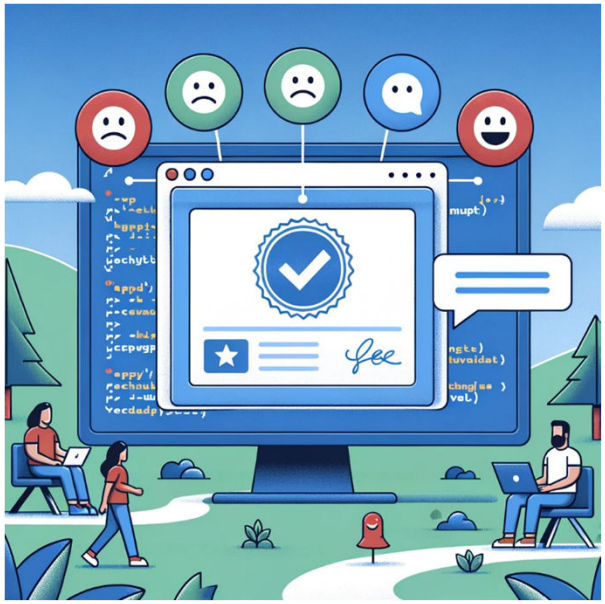
- ไม่ทราบถึงโครงสร้างหรือโค้ดภายในโปรแกรม
- ข้อมูลนำเข้า (input) และตรวจสอบผลลัพธ์ (output)

การวิเคราะห์และอธิบายโปรแกรม

- **การวิเคราะห์โปรแกรม (Program Analysis)**:
  - ศึกษาและทำความเข้าใจโครงสร้าง, การทำงาน, และส่วนประกอบต่าง ๆ
  - ค้นหาปัญหาและข้อผิดพลาด หรือส่วนที่สามารถปรับปรุงได้
  - หาเทคนิคการทดสอบ การปรับปรุงประสิทธิภาพ หรือผลกระทบ
- **การอธิบายโปรแกรม (Program Explanation)**:
  - สร้างเอกสาร, แผนภาพ, หรือแนวทางการใช้งาน
  - ช่วยให้ทีมพัฒนา, ผู้ใช้งาน, และผู้มีส่วนได้เสียเข้าใจ
  - สร้างคู่มือ วิดีโอสาธิตการใช้งาน, หรือการอบรม

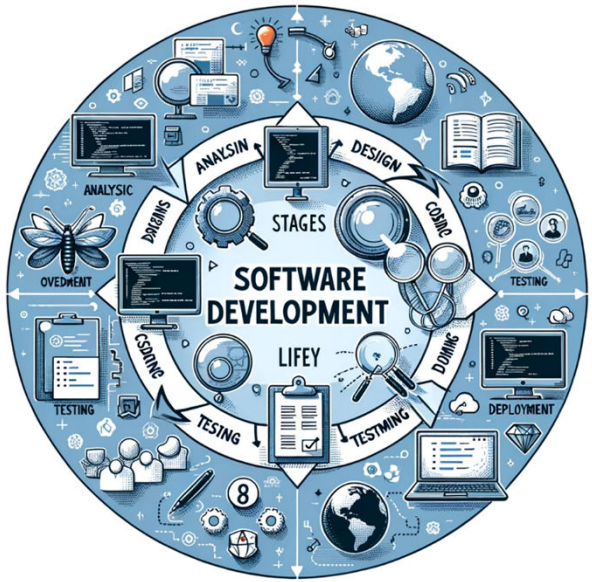
การรับรองและประเมินผล

- ยืนยันว่าได้ตรงตามข้อกำหนดหรือมาตรฐานที่กำหนดไว้
- วัดและประเมินความสามารถ, ประสิทธิภาพ, และคุณภาพการทำงาน



สรุปท้ายบท

- การฝึกปฏิบัติเขียนโปรแกรมเป็นกระบวนการที่ซับซ้อน
- รวมถึงการวิเคราะห์ การออกแบบ การทดสอบ และการประกันคุณภาพ
- การเข้าใจในแต่ละขั้นตอนและเทคนิคต่าง ๆ
- ช่วยให้การพัฒนาซอฟต์แวร์มีประสิทธิภาพ
- สร้างผลงานที่มีคุณภาพได้



ถาม-ตอบ

ขอบคุณ

- ขอบคุณสำหรับการฟังและการเข้าร่วม
- แหล่งข้อมูลเพิ่มเติมหรืออ้างอิง
- ข้อมูลติดต่อสำหรับคำถามเพิ่มเติม