ENGCE117 การเขียนโปรแกรมสำหรับวิศวกรคอมพิวเตอร์ 3(2-3-5)

(Computer Programming for Computer Engineer)

รหัสรายวิชาเดิม : ไม่มี

วิชาบังคับก่อน : ENGCC304 การเขียนโปรแกรมคอมพิวเตอร์

ศึกษาและฝึกปฏิบัติการเกี่ยวกับหลักการเขียนโปรแกรมเชิงลึกเกี่ยวกับ พอยน์เตอร์ และอาเรย์ การจองหน่วยความจำ ไฟล์อินพุต-เอาต์พุต ฟังก์ชันเรียกตัวเอง การเขียน และออกแบบโปรแกรมเชิงวัตถุ การซ่อนข้อมูล การสืบทอด การพ้องรูป คลาสนามธรรม การเขียนโปรแกรมแบบหลายเทรด การพัฒนาส่วนติดต่อกับผู้ใช้ การ ทดสอบซอฟต์แวร์ เครื่องมือที่ใช้ในการพัฒนาซอฟต์แวร์ ระบบควบคุมเวอร์ชัน

Study and practice of advanced programming concepts on pointers and arrays, memory allocation, file I/O, recursive function, object-oriented programming and design, encapsulation, inheritance, polymorphism, abstract class, multithreaded programming, graphical user interface, software testing, software development tools, version control systems.

ตารางที่ 1 แผนการสอนรายวิชา การเขียนโปรแกรมสำหรับวิศวกรคอมพิวเตอร์

สัปดาห์	หัวข้อการเรียนการสอน	กิจกรรม / การบ้าน / การประเมินผล
1	**ทบทวนและหลักการของ Python**	• ทบทวนความรู้จาก ENGCC304 (ตัวแปร, การควบคุม, ฟังก์ชัน, โครงสร้างข้อมูลพื้นฐาน) • The Zen of Python (PEP 20) และ มาตรฐานการเขียนโค้ด (PEP 8) • การจัดการสภาพแวดล้อม (Virtual Environments: venv) และการจัดการแพ็คเกจ (pip)
2	**โครงสร้างข้อมูลขั้นสูง (Advanced Data Structures)**	• ทบทวน Lists, Tuples, Dictionaries, Sets • การใช้งานขั้นสูง: List/Dict/Set Comprehensions • การจัดการ Collections (เช่น collections.namedtuple, collections.deque)

ตารางที่ 1 แผนการสอนรายวิชา การเขียนโปรแกรมสำหรับวิศวกรคอมพิวเตอร์ (ต่อ)

สัปดาห์	หัวข้อการเรียนการสอน	กิจกรรม / การบ้าน / การประเมินผล
3	**ฟังก์ชันขั้นสูงและ File I/O (Advanced Functions & File I/O)**	• การใช้งาน *args และ **kwargs • ฟังก์ชัน Lambda, map, filter, reduce • การจัดการไฟล์: with open(...), การอ่าน/เขียน JSON และ CSV
4	**ฟังก์ชันเรียกตัวเอง (Recursion) และ Generators**	• แนวคิดของฟังก์ชันเรียกตัวเอง (Recursive Functions) (Base Case, Recursive Step) • แนวคิดของ Generators และการใช้ yield (Lazy Evaluation)
5	**แนะนำการเขียนโปรแกรมเชิงวัตถุ (Introduction to OOP)**	• แนวคิดหลักของ OOP (Encapsulation, Inheritance, Polymorphism) • การสร้างคลาส (class), self, เมธอด __init__ • Class Attributes vs. Instance Attributes
6	**Quiz ครั้งที่ 1 และ ระบบควบคุมเวอร์ชัน (Version Control)**	• **Quiz ครั้งที่ 1 (เนื้อหา สัปดาห์ที่ 1-5)** • ความสำคัญของ Version Control • การใช้งาน Git เบื้องต้น (init, add, commit, push, pull, branch)
7	**OOP - การห่อหุ้มข้อมูล (Encapsulation)**	• การควบคุมการเข้าถึง: Public, _Protected, __Private (Name Mangling) • การใช้ Properties (@property) แทน Getters/Setters
8	**OOP - การสืบทอด (Inheritance)**	• แนวคิด Base Class และ Derived Class • Method Overriding และการใช้ super() • Multiple Inheritance และ Method Resolution Order (MRO)

ตารางที่ 1 แผนการสอนรายวิชา การเขียนโปรแกรมสำหรับวิศวกรคอมพิวเตอร์ (ต่อ)

สัปดาห์	หัวข้อการเรียนการสอน	กิจกรรม / การบ้าน / การประเมินผล
9	**สอบกลางภาค**	• **ดำเนินการสอบกลางภาค** (ข้อเขียนหรือภาคปฏิบัติตามความเหมาะสม)
10	**OOP - การพ้องรูป (Polymorphism) และ ABCs**	• แนวคิดเรื่อง Duck Typing ใน Python • คลาสฐานนามธรรม (Abstract Base Classes - abc module) • การใช้ @abstractmethod
11	**OOP - Magic Methods และ Decorators**	• การทำ Operator Overloading ด้วย Magic Methods (เช่น __str__, __repr__, __add__, __eq__) • แนวคิดและการสร้าง Decorators (@...)
12	**การพัฒนาส่วนติดต่อกับผู้ใช้ (GUI Development)**	• แนะนำไลบรารีสำหรับการสร้าง GUI (เช่น Tkinter หรือ PyQt) • แนวคิดเรื่อง Event-Driven Programming • การจัดการ Widgets และ Layouts • *(มอบหมายการบ้านโครงงานประยุกต์ OOP/GUI)*
13	**การทดสอบซอฟต์แวร์ (Software Testing)**	• ความสำคัญของการทดสอบซอฟต์แวร์ • แนวคิด Unit Testing • การใช้โมดูล unittest หรือ pytest ในการเขียน Test Cases
14	**Quiz ครั้งที่ 2 และ การเขียนโปรแกรมแบบหลายเทรด (Multithreading)**	• **Quiz ครั้งที่ 2 (เนื้อหา สัปดาห์ที่ 10-13)** • แนวคิดของ Concurrency • การใช้โมดูล threading สำหรับ I/O-bound tasks • การป้องกัน Race Condition ด้วย threading.Lock

ตารางที่ 1 แผนการสอนรายวิชา การเขียนโปรแกรมสำหรับวิศวกรคอมพิวเตอร์ (ต่อ)

สัปดาห์	หัวข้อการเรียนการสอน	กิจกรรม / การบ้าน / การประเมินผล
15	**Concurrency - Global Interpreter Lock (GIL) และ Multiprocessing**	• ทำความเข้าใจ Global Interpreter Lock (GIL) และข้อจำกัดของ Threading • การใช้โมดูล multiprocessing สำหรับ CPU-bound tasks
16	**Concurrency - Asyncio และเครื่องมือพัฒนา (Dev Tools)**	• แนะนำ asyncio และ async/await สำหรับ High-level I/O Concurrency • การใช้ Debugger (pdb หรือ Debugger ใน IDE)
17	**การสังเคราะห์แนวคิด Concurrency และทบทวน**	• การเปรียบเทียบเชิงลึก: Threading vs. Multiprocessing vs. Asyncio • การวิเคราะห์ปัญหา: I/O-bound vs. CPU-bound • กรณีศึกษา: "ควรใช้อะไรเมื่อไหร่?" (When to use what?)
18	**สอบปลายภาค**	• **ดำเนินการสอบปลายภาค** (ภาคทฤษฎีหรือปฏิบัติตามลักษณะรายวิชา)

ตารางที่ 2 การประเมินผลรายวิชา การเขียนโปรแกรมสำหรับวิศวกรคอมพิวเตอร์

องค์ประกอบการประเมินผล	รายละเอียด	สัดส่วน (%)
งานเดี่ยวและแบบฝึกหัดรายสัปดาห์	• แบบฝึกหัดการเขียนโปรแกรม ผังงาน และโจทย์เชิงปฏิบัติ • พิจารณาจากความครบถ้วน ความถูกต้อง ความคิดสร้างสรรค์ และการส่งงานตรงเวลา	20%
แบบทดสอบย่อยและกิจกรรมในชั้นเรียน	• แบบทดสอบสั้น (2 ครั้ง) ในหัวข้อสำคัญ • กิจกรรมกลุ่ม: วิเคราะห์โค้ดและแก้ไขปัญหา	20%
สอบกลางภาค	• การสอบวัดผลความเข้าใจเนื้อหาครึ่งแรกของรายวิชา	25%
สอบปลายภาค	• การสอบวัดผลความเข้าใจเนื้อหาครึ่งหลังของรายวิชา	25%
คุณลักษณะนิสัยและวินัยในการเรียนรู้	• การเข้าชั้นเรียนตรงเวลาและสม่ำเสมอ • ความรับผิดชอบในการส่งงาน • การมีส่วนร่วมในชั้นเรียนและการทำงานกลุ่ม • ความมีวินัยและความรับผิดชอบต่อตนเองและผู้อื่น	10%
รวม		100%