การประชุมวิชาการทางวิศวกรรมไฟฟ้า ครั้งที่ 48
The 48th Electrical Engineering Conference (EECON-48)
วันที่ 19-21 พฤศจิกายน 2568 ณ โรงแรมฟูราม่า จังหวัดเชียงใหม่

MAPLE-J: เฟรมเวิร์กจัดการสภาพแวดล้อมไพทอนอัตโนมัติแบบหลายแพลตฟอร์มสำหรับการใช้งานด้านการศึกษาโดยใช้ JupyterLab

MAPLE-J: A Multi-Platform Automated Python Learning Environment Framework for Educational Applications Using JupyterLab

บุญนที ศักดิ์บุญญารัตน์ และ ศิริพร ศักดิ์บุญญารัตน์*

สาขาวิชาคณิตศาสตร์และวิทยาการคำนวณ โรงเรียนมหิดลวิทยานุสรณ์ boonnatee.sak@mwit.ac.th, siriporn.sak@mwit.ac.th*

บทคัดย่อ

งานวิจัยนี้นำเสนอการพัฒนาเฟรมเวิร์กบริหารจัดการไพทอน (Python) อัตโนมัติแบบหลายแพลตฟอร์มเพื่อการเรียนการสอน โดยใช้เทคโนโลยี pyenv สำหรับการจัดการหลายเวอร์ชันของ Python และป้องกันปัญหาความไม่สอดคล้องของไลบรารี เฟรมเวิร์กประกอบด้วยสคริปต์อัตโนมัติที่ติดตั้งและกำหนดค่าสภาพแวดล้อมพร้อมใช้งาน ครอบคลุมไลบรารีสำคัญสำหรับการประมวลผลภาพและการพัฒนาโมเดล AI เช่น OpenCV, Pillow, scikit-image, scikit-learn และ TensorFlow พร้อมติดตั้งและตั้งค่า JupyterLab สำหรับใช้งานในโหมด localhost บน Windows, macOS และ Linux ผลการทดสอบแสดงว่าเฟรมเวิร์กนี้สามารถลดเวลาในการติดตั้งได้เทียบเท่าการใช้งานบนระบบ Google Colab เพิ่มความเสถียร ความสามารถในการทำซ้ำ (Reproducibility) และความสะดวกในการใช้งาน ทำให้เหมาะสำหรับการสอนการเขียนโปรแกรมและการประยุกต์ใช้ในสภาพแวดล้อมการเรียนการสอนที่หลากหลาย

คำสำคัญ: Python, pyenv, JupyterLab, เฟรมเวิร์กอัตโนมัติ, หลายแพลตฟอร์ม, การจัดการสภาพแวดล้อม

Abstract

This research presents the development of a cross-platform automated Python management framework for educational purposes. The framework utilizes the pyenv technology to manage multiple Python versions and prevent library incompatibility issues. It consists of automated scripts that install and configure a ready-to-use environment, covering essential libraries for image processing and AI model development, such as OpenCV, Pillow, scikit-image, scikit-learn, and TensorFlow. Additionally, the framework installs and configures JupyterLab for localhost operation on Windows, macOS, and Linux. Experimental results indicate that the framework can reduce installation time to a level comparable to that of Google Colab, while enhancing stability, reproducibility, and ease of use. These features make it highly suitable for teaching programming and applying it in diverse educational environments.

Keywords: Python, pyenv, JupyterLab, Automated Framework, Multi-platform, Environment Management

1. บทนำ

การเรียนการสอนด้านการเขียนโปรแกรมและการพัฒนาโมเดลปัญญาประดิษฐ์ (Artificial Intelligence, AI) ในปัจจุบันจำเป็นต้องใช้สภาพแวดล้อมการทำงานที่รองรับไลบรารีและเครื่องมือจำนวนมาก เช่น OpenCV, Pillow, scikit-image, scikit-learn และ TensorFlow ซึ่งมักมีข้อกำหนดด้านเวอร์ชันที่แตกต่างกัน การติดตั้งและตั้งค่าสภาพแวดล้อมเหล่านี้บนคอมพิวเตอร์ของผู้เรียนหรือผู้สอนอาจใช้เวลานาน และมักเกิดปัญหาความไม่สอดคล้องของไลบรารี (Library Incompatibility) หรือความแตกต่างระหว่างระบบปฏิบัติการต่าง ๆ เช่น Windows, macOS, Linux ซึ่งทำให้เกิดความล่าช้าและลดประสิทธิภาพในการเรียนการสอน

แม้ว่าจะมีเครื่องมือช่วยจัดการเวอร์ชันของ Python และไลบรารี เช่น virtualenv, conda หรือ Docker แต่การตั้งค่าให้พร้อมใช้งานในสภาพแวดล้อมการเรียนการสอนยังคงซับซ้อนสำหรับผู้ใช้ทั่วไป โดยเฉพาะผู้ที่ไม่มีพื้นฐานด้านการจัดการระบบ (system administration) ความซับซ้อนนี้ส่งผลให้ผู้สอนต้องใช้เวลามากในการช่วยแก้ไขปัญหาแทนที่จะใช้เวลาไปกับการสอนเนื้อหา และผู้เรียนอาจเสียโอกาสในการฝึกปฏิบัติจริง

ดังนั้น จึงมีความจำเป็นในการพัฒนาเฟรมเวิร์กที่สามารถติดตั้งและกำหนดค่าสภาพแวดล้อมสำหรับการเรียนการสอน Python และ AI ได้อย่างอัตโนมัติ รองรับการทำงานข้ามแพลตฟอร์ม และลดปัญหาความไม่สอดคล้องของไลบรารี เพื่อเพิ่มความเสถียร ความสามารถในการทำซ้ำ และความรวดเร็วในการเตรียมสภาพแวดล้อม ผลลัพธ์ที่ได้จะช่วยให้กระบวนการเรียนการสอนมีประสิทธิภาพมากขึ้น เหมาะสมกับการนำไปใช้ทั้งในชั้นเรียนและการเรียนรู้ด้วยตนเอง

2. วัตถุประสงค์ของการวิจัย

2.1 เพื่อพัฒนาเฟรมเวิร์กบริหารจัดการ Python แบบอัตโนมัติที่สามารถทำงานได้บนหลายแพลตฟอร์ม ได้แก่ Windows, macOS, Linux และ Raspberry Pi

2.2 เพื่อออกแบบสคริปต์อัตโนมัติที่ติดตั้งและตั้งค่าพร้อมใช้งาน รวมถึงติดตั้ง JupyterLab สำหรับโหมด localhost

2.3 เพื่อส่งเสริมประสิทธิภาพการเรียนการสอนด้านการเขียนโปรแกรมคอมพิวเตอร์ โดยลดเวลาและความซับซ้อนในการเตรียมสภาพแวดล้อม

*ผู้ประพันธ์บรรณกิจ

3. ขอบเขตการวิจัย

3.1 ด้านระบบปฏิบัติการที่รองรับ
- เฟรมเวิร์กถูกออกแบบให้สามารถทำงานได้หลายระบบปฏิบัติการหลัก ได้แก่ Windows, macOS, Linux และ Raspberry Pi

3.2 ด้านเทคโนโลยีและเครื่องมือที่ใช้
- ใช้ pyenv ในการจัดการหลายเวอร์ชันของ Python
- ใช้สคริปต์อัตโนมัติในการติดตั้งและกำหนดค่าระบบ
- ติดตั้งไลบรารีสำคัญสำหรับการเขียนโปรแกรมคอมพิวเตอร์
- ติดตั้งและตั้งค่า JupyterLab สำหรับการใช้งานในโหมด localhost

3.3 ด้านการทดสอบและประเมินผล
- ทดสอบการติดตั้งและใช้งานเฟรมเวิร์กในสภาพแวดล้อมการเรียนการสอน
- วัดผลด้านเวลาในการติดตั้ง ความเสถียร ความสามารถในการทำซ้ำ และความสะดวกในการใช้งาน
- กลุ่มตัวอย่างการทดสอบประกอบด้วยผู้สอนและผู้เรียนในรายวิชาที่เกี่ยวข้องกับการเขียนโปรแกรมคอมพิวเตอร์และวิชาโครงงาน

3.4 ด้านข้อจำกัดของการวิจัย
- ไม่ครอบคลุมการติดตั้งบนสภาพแวดล้อมที่ไม่มีสิทธิ์การติดตั้งซอฟต์แวร์ (เช่น ระบบที่จำกัดสิทธิ์ผู้ใช้)
- ไม่ครอบคลุมการติดตั้งไลบรารีเฉพาะทางอื่นนอกเหนือจากที่กำหนดในเฟรมเวิร์ก

4. ตัวแปรต้นและตัวแปรตาม

4.1 ตัวแปรต้น (Independent Variables)
- เทคโนโลยีที่ใช้จัดการ Python หลายเวอร์ชัน (pyenv)
- สคริปต์อัตโนมัติในการติดตั้งและกำหนดค่าสภาพแวดล้อม
- ไลบรารีสำคัญสำหรับการเขียนโปรแกรมคอมพิวเตอร์
- การติดตั้งและตั้งค่า JupyterLab ในโหมด localhost
- รองรับหลายแพลตฟอร์ม ได้แก่ Windows, macOS, Linux และ Raspberry Pi

4.2 ตัวแปรตาม (Dependent Variables)
- เวลาในการติดตั้งสภาพแวดล้อม (นาที)
- ความเสถียรของระบบ (System Stability)
- ความสามารถในการทำซ้ำ
- ความสะดวกในการใช้งาน (Usability)
- ประสิทธิภาพการเรียนการสอนด้านการเขียนโปรแกรมและวิชาโครงงาน

5. ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

5.1 ทฤษฎีที่เกี่ยวข้อง

การจัดการเวอร์ชันและสภาพแวดล้อมซอฟต์แวร์ pyenv เป็นเครื่องมือยอดนิยมสำหรับการจัดการหลายเวอร์ชันของ Python โดยช่วยให้สามารถติดตั้ง สลับใช้งาน และจัดการเวอร์ชันต่าง ๆ อย่างเป็นระบบ ลดความเสี่ยงจาก Python เวอร์ชันระบบ (Python System) และการติดตั้งซ้ำซ้อน pyenv-win คือ เวอร์ชันสำหรับ Windows ที่ได้รับการดัดแปลงจาก pyenv เพื่อใช้งานในแพลตฟอร์มนี้ [1][2]

ความสำคัญของการทดลองซ้ำได้ แนวคิดการทดลองในรูปแบบโค้ด (Experiments as Code, ExaC) เป็นแนวคิดสำหรับการทดลองที่สามารถทำซ้ำได้ ตรวจสอบได้ แก้ไขข้อบกพร่องได้ นำกลับมาใช้ใหม่ได้ และปรับขนาดได้ [3] โครงการ SIERRA เป็นเฟรมเวิร์กแบบโมดูลาร์ (Modular) ที่รองรับการทำแบบอัตโนมัติสำหรับงานทดลองต่าง ๆ เพิ่มความสามารถในการทำซ้ำ และลดการเขียนสคริปต์ที่ใช้แล้วทิ้ง [4] รายงานจาก National Academies of Sciences พูดถึงลำดับขั้นตอนการทำงานที่เป็นระบบและมีแบบแผนตั้งแต่ต้นจนจบกระบวนการ (Workflow) การวิจัยอัตโนมัติ (Automated Research Workflows, ARWs) ช่วยเพิ่มความมั่นคง ความสามารถในการทำซ้ำ ด้วยระบบอัตโนมัติตั้งแต่การเตรียมข้อมูลจนถึงการเผยแพร่ผล [5]

ปัจจัยสำคัญของการวิจัยเชิงคำนวณที่ทำซ้ำได้ (Reproducible Computational Research) งานวิจัยในด้านชีวสารสนเทศศาสตร์ (Bioinformatics) ได้นำเสนอเสาหลักทั้งห้าของการวิจัยเชิงคำนวณที่ทำซ้ำได้ (Five Pillars of Reproducible Computational Research) ได้แก่ 1) แนวคิดการเขียนโปรแกรมที่ผสมผสานภาษาโปรแกรมเข้ากับภาษาธรรมชาติ (Literate Programming) 2) ระบบควบคุมเวอร์ชัน (Code version control) เป็นระบบที่ใช้ในการติดตามและจัดการการเปลี่ยนแปลงของซอร์สโค้ด (และไฟล์อื่น ๆ) ในโปรเจกต์ซอฟต์แวร์อย่างเป็นระบบ ช่วยให้นักพัฒนาสามารถย้อนกลับไปดูเวอร์ชันก่อนหน้า เปรียบเทียบการเปลี่ยนแปลง และทำงานร่วมกับผู้อื่นได้อย่างราบรื่น พร้อมทั้งป้องกันการสูญหายของโค้ดและแก้ไขข้อผิดพลาดได้อย่างมีประสิทธิภาพ 3) การจัดการการพึ่งพาซอฟต์แวร์ (Software Dependency Management) คือ กระบวนการระบุ ติดตาม และควบคุมองค์ประกอบซอฟต์แวร์ต่างๆ เช่น ไลบรารี, เฟรมเวิร์ก, และมอดูล ที่แอปพลิเคชันต้องพึ่งพา เพื่อให้ทำงานได้อย่างมีประสิทธิภาพและปลอดภัย โดยมีเป้าหมายหลักเพื่อให้มั่นใจว่าส่วนประกอบเหล่านั้นเป็นเวอร์ชันที่ถูกต้องเข้ากันได้ และไม่มีช่องโหว่ด้านความปลอดภัย ซึ่งช่วยลดข้อผิดพลาด ป้องกันความเสี่ยง และปรับปรุงความเสถียรของซอฟต์แวร์โดยรวม 4) แหล่งที่มาของข้อมูล (Data Provenance) และ 5) การตรวจจับสภาพแวดล้อมเชิงคำนวณ (Computational Environment Capture) แนวคิดนี้ช่วยเตือนให้เห็นว่าการควบคุมสภาพแวดล้อม เช่น การใช้ pyenv เป็นส่วนหนึ่งที่สำคัญมากต่อความสามารถในการทำซ้ำ [6]

ระบบการทดสอบกรงในบ้านแบบอัตโนมัติ (Automated Home-Cage Testing) ในงานวิจัยพฤติกรรมศาสตร์ พบว่าการทำระบบอัตโนมัติช่วยลดข้อผิดพลาด และเพิ่มความสามารถในการทำซ้ำของผลลัพธ์ได้อย่างมีนัยสำคัญ [7]

5.2 งานวิจัยที่เกี่ยวข้อง

การพัฒนาระบบหรือเฟรมเวิร์กเพื่อสนับสนุนการจัดการสภาพแวดล้อมการเขียนโปรแกรมและเพิ่มความสามารถในการทำซ้ำได้รับความสนใจอย่างต่อเนื่องในวงการวิจัยและการศึกษา เช่น [8] ได้นำเสนอการฝึกอบรมให้นักศึกษาสาขา STEM และสังคมศาสตร์ได้เรียนรู้การจัดการข้อมูล และการทำให้ขั้นตอนการวิเคราะห์มีความสามารถในการทำซ้ำอย่างเป็นระบบ ซึ่งสอดคล้องกับการพัฒนาทักษะด้านเทคนิคของผู้เรียน ขณะที่ [9] ได้ออกแบบโครงสร้างระบบที่ผสานเทคโนโลยีคอนเทนเนอร์ (Container) การควบคุมเวอร์ชัน (Version Control) และการ

จัดเก็บข้อมูลถาวร เพื่อให้การจัดเก็บโค้ด ข้อมูล และผลลัพธ์มีความโปร่งใสและพร้อมตรวจสอบได้ในระยะยาว

ในด้านการจัดการสภาพแวดล้อมแบบ [10] ที่พัฒนาเครื่องมือผสาน Git, Docker และ Jupyter เข้าด้วยกันเพื่อให้ผู้เรียนสามารถเริ่มใช้งานได้ทันทีโดยไม่ต้องติดตั้งและกำหนดค่าด้วยตนเอง ซึ่งเป็นแนวทางที่ช่วยลดอุปสรรคทางเทคนิคในการเรียนการสอนอย่างมีประสิทธิภาพ นอกจากนี้งาน [11] ได้สำรวจสถานการณ์ของการทำซ้ำงานวิจัยในสาขาวิทยาการคอมพิวเตอร์ พบว่าการจัดการความไม่มีอิสระและการสร้างสภาพแวดล้อมที่เสถียรผ่านเครื่องมือ เช่น สภาพแวดล้อมเสมือนจริง (Virtual environments) และเทคโนโลยีคอนเทนเนอร์ (Container technologies) เช่น Docker, Vagrant เป็นปัจจัยสำคัญต่อความสำเร็จของการทำซ้ำ ซึ่งชี้ให้เห็นความจำเป็นของการพัฒนาเครื่องมือหรือเฟรมเวิร์กที่สามารถลดความซับซ้อนในการตั้งค่าและเพิ่มความน่าเชื่อถือของกระบวนการเรียนการสอนด้านการเขียนโปรแกรมและ AI

6. วิธีดำเนินการวิจัย

6.1 กลุ่มประชากร

กลุ่มประชากรในการวิจัยนี้ คือ ผู้สอนและผู้เรียนในรายวิชาการเขียนโปรแกรมคอมพิวเตอร์และโครงงานวิชาคอมพิวเตอร์ โรงเรียนมหิดลวิทยานุสรณ์ ซึ่งใช้ระบบปฏิบัติการที่แตกต่างกัน ได้แก่ Windows, macOS และ Linux โดยครอบคลุมทั้งในสภาพแวดล้อมการเรียนในห้องเรียนและการเรียนรู้ด้วยตนเอง

6.2 กลุ่มตัวอย่าง

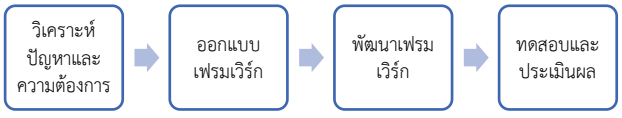
กลุ่มตัวอย่างได้มาจากการเลือกแบบเจาะจง (Purposive Sampling) จากกลุ่มประชากรดังกล่าว เพื่อให้ครอบคลุมการใช้งานจริงในสภาพแวดล้อมที่หลากหลาย โดยเป็นผู้สอน 3 คน และผู้เรียนประมาณ 144 และ11 คน จากรายวิชาการเขียนโปรแกรมคอมพิวเตอร์และโครงงานวิชาคอมพิวเตอร์ตามลำดับ

การเลือกแบบเจาะจงเหมาะสมกับงานวิจัยนี้ เนื่องจากเฟรมเวิร์กที่พัฒนามีเป้าหมายเพื่อแก้ไขปัญหาการติดตั้งและจัดการ Python ในการเรียนการสอน จึงจำเป็นต้องเลือกกลุ่มตัวอย่างที่มีลักษณะและเงื่อนไขตรงตามเกณฑ์ที่กำหนด เพื่อให้สามารถประเมินประสิทธิภาพและความเหมาะสมของเฟรมเวิร์กได้อย่างตรงจุด และลดความแปรปรวนจากปัจจัยที่ไม่เกี่ยวข้อง

กลุ่มตัวอย่างทั้งหมดใช้เครื่องคอมพิวเตอร์ที่มีระบบปฏิบัติการแตกต่างกัน ได้แก่ Windows, macOS, Linux และ Raspberry Pi เพื่อทดสอบประสิทธิภาพของเฟรมเวิร์ก

6.3 วิธีดำเนินการ

การดำเนินการวิจัย ดำเนินการตามกรอบการวิจัย ดังรูปที่ 1



รูปที่ 1 กรอบการวิจัย

6.3.1 วิเคราะห์ปัญหาและความต้องการ

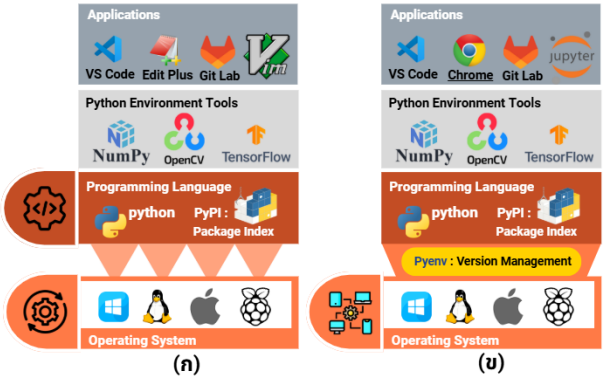
- ศึกษาปัญหาที่เกิดขึ้นในการติดตั้งและจัดการสภาพแวดล้อม Python สำหรับการเรียนการสอน
- สำรวจความต้องการของผู้สอนและผู้เรียนเกี่ยวกับเครื่องมือและไลบรารีที่จำเป็น
- วิเคราะห์ข้อจำกัดของเครื่องมือเดิม เช่น Google Colab, Repli

6.3.2 ออกแบบเฟรมเวิร์ก

เฟรมเวิร์กออกแบบให้รองรับการทำงานบนหลายระบบปฏิบัติการ ได้แก่ Windows, macOS, Linux และ Raspberry Pi พร้อมทั้งพัฒนาชุดคำสั่งติดตั้งระบบอัตโนมัติ Automated Installation Script for Pyenv-Virtualenv (AuScript-Pyenv) เพื่อใช้เปรียบเทียบระหว่างเฟรมเวิร์กที่ผู้วิจัยพัฒนา MAPLE-J กับเฟรมเวิร์กอื่น ๆ แสดงในตารางที่ 1 โดยเฟรมเวิร์กแต่ละแบบถูกประเมินตามความสามารถและวิธีการใช้งานใน 8 ด้านสำคัญ โดยใช้เกณฑ์การประเมินแบบมาตราส่วน 5 ระดับ (1 – 5 คะแนน) เพื่อสะท้อนถึงประสิทธิภาพและความเหมาะสมของแต่ละเฟรมเวิร์ก

ตารางที่ 1 ตารางแสดงรูปแบบของเฟรมเวิร์กที่ใช้ในการทดลอง

ลำดับ	ชื่อเฟรมเวิร์ก	เครื่องมือหลักที่ใช้งานของเฟรมเวิร์ก
1	G-Colab	Google Colaboratory : Colab
2	PN-Traditional	Python Native Traditional
3	PY-jupyterLAB	Pyenv และ jupyterLAB
4	MAPLE-J	AuScript-Pyenv และ jupyterLAB

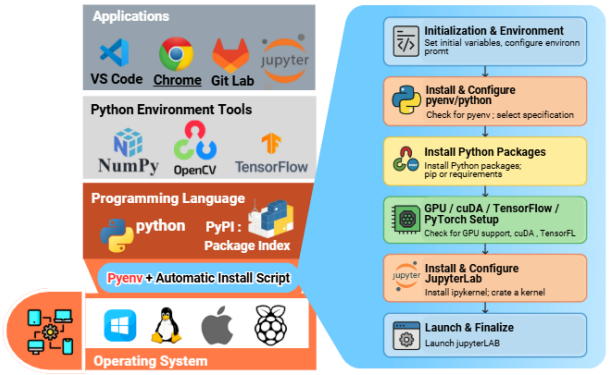


รูปที่ 2 การทำงานของเฟรมเวิร์ก ก) PN-Traditional เป็นการเรียนใช้งาน Python แบบดั้งเดิม และ ข) PY-jupyterLAB เป็นการใช้ Pyenv เข้ามาช่วยจัดการเวอร์ชั่น

ในการออกแบบเฟรมเวิร์กได้พิจารณารูปแบบการทำงานของระบบที่ใช้งานอยู่เดิมเปรียบเทียบกับเฟรมเวิร์กที่ผู้วิจัยพัฒนาขึ้นใหม่ แสดงดังรูปที่ 2 และ รูปที่ 3 ตามลำดับ

รูปที่ 2 แสดงการทำงานของเฟรมเวิร์กสองรูปแบบ ได้แก่ (ก) PN-Traditional ซึ่งเป็นการใช้งาน Python แบบดั้งเดิม โดยผู้ใช้ต้องติดตั้งและจัดการสภาพแวดล้อมด้วยตนเอง และ (ข) PY-jupyterLAB ซึ่งมีการนำ Pyenv เข้ามาช่วยจัดการหลายเวอร์ชันของ Python เพื่อเพิ่มความยืดหยุ่นในการเรียนการสอน รูปที่ 3 แสดงการทำงานของเฟรมเวิร์กที่ผู้วิจัยพัฒนาขึ้น คือ MAPLE-J โดยใช้ AuScript-Pyenv (Automated Installation Script for Pyenv-Virtualenv) เพื่อทำให้ขั้นตอนการติดตั้งและกำหนดค่าสภาพ แวดล้อมเป็นแบบอัตโนมัติ ตั้งแต่การกำหนดค่าตัวแปรเริ่มต้น การติดตั้งไลบรารี Python การตรวจสอบการรองรับ GPU/cuDNN ไปจนถึงการติดตั้ง JupyterLab และการเตรียมใช้งานจริง

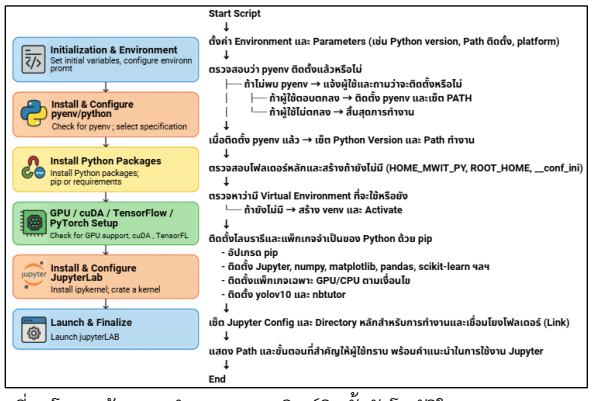
เฟรมเวิร์กนี้ช่วยลดความ ซับซ้อน เพิ่มความเสถียร และรองรับการใช้งานได้บนหลายระบบปฏิบัติการอย่างมีประสิทธิภาพ



รูปที่ 3 การทำงานของเฟรมเวิร์ก MAPLE-J

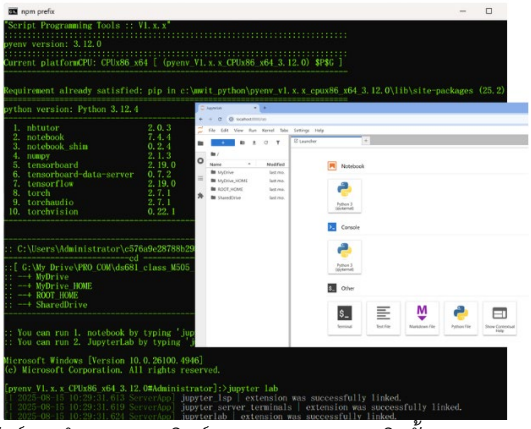
6.3.3 การพัฒนาเฟรมเวิร์ก

ในการพัฒนาเฟรมเวิร์ก MAPLE-J ผู้วิจัยได้สร้างและทดสอบสคริปต์ติดตั้งโดยใช้ Pyenv สำหรับจัดการหลายเวอร์ชันของ Python โดยออกแบบให้มีขั้นตอนการทำงานแบบอัตโนมัติ แสดงดังรูปที่ 4 ซึ่งเป็นโครงสร้างการทำงานของสคริปต์ (Automated Installation Script) ครอบคลุมตั้งแต่การกำหนดค่าพารามิเตอร์เริ่มต้น การตรวจสอบและติดตั้ง Pyenv การสร้างและจัดการ Virtual Environment การติดตั้งไลบรารีหลักด้านการประมวลผลภาพและ AI เช่น NumPy, OpenCV, TensorFlow, scikit-learn การตรวจสอบความพร้อมของ GPU/cuDNN ไปจนถึงการติดตั้งและกำหนดค่า JupyterLab ให้พร้อมใช้งาน



รูปที่ 4 โครงสร้างการทำงานของสคริปต์ติดตั้งอัตโนมัติในระบบ MAPLE-J ตั้งแต่การกำหนดค่าเริ่มต้น การจัดการเวอร์ชัน Python และการติดตั้งไลบรารี ไปจนถึงการเตรียมใช้งาน JupyterLab

การทำงานของสคริปต์ MAPLE-J สามารถติดตั้งสภาพแวดล้อม Python และไลบรารีที่จำเป็นได้อย่างครบถ้วน พร้อมเรียกใช้งาน JupyterLab ได้สำเร็จบนเครื่องผู้ใช้ โดยช่วยลดความซับซ้อน เพิ่มความเสถียร และทำให้สามารถเริ่มต้นใช้งานได้ทันทีโดยไม่ต้องกำหนดค่าด้วยตนเอง สคริปต์ถูกออกแบบให้รองรับความแตกต่างของแต่ละระบบปฏิบัติการ โดยใช้ PowerShell บน Windows และ bash shell scripts บน macOS, Linux และ Raspberry Pi OS ผลการทำงานแสดงใน รูปที่ 5



รูปที่ 5 ผลลัพธ์การทำงานของสคริปต์ MAPLE-J แสดงการติดตั้งสภาพแวดล้อมและไลบรารีหลักที่จำเป็น พร้อมการเรียกใช้งาน JupyterLab ได้สำเร็จ

6.3.4 ทดสอบและประเมินผล

เพื่อประเมินประสิทธิภาพของเฟรมเวิร์ก MAPLE-J การวิจัยนี้ได้ออกแบบเกณฑ์การประเมินโดยอ้างอิง 8 มิติหลักที่สะท้อนถึงความสามารถและวิธีการใช้งานของแต่ละเฟรมเวิร์ก ประกอบด้วย ความง่ายในการติดตั้ง การพึ่งพาอินเทอร์เน็ต การจัดการหลายเวอร์ชันของ Python การติดตั้งและจัดการแพ็กเกจ ความสามารถในการทำงานร่วมกัน ความเสถียรของระบบ ความง่ายในการเริ่มใช้งาน และความยืดหยุ่นต่อการขยาย โดยใช้มาตราส่วนการประเมินแบบ 5 ระดับ (1–5 คะแนน) เพื่อสะท้อนระดับประสิทธิภาพ ความสะดวก และความเหมาะสมของแต่ละเฟรมเวิร์ก รายละเอียดเกณฑ์การประเมินสรุปแสดงดังตารางที่ 2

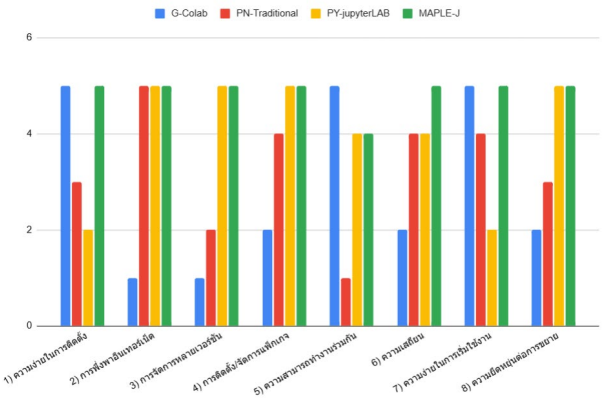
ตารางที่ 2 เกณฑ์การประเมินเฟรมเวิร์ก (1 ถึง 5 คะแนน)

ลำดับ	รายการความสามารถ และ เกณฑ์การประเมิน	
1	ความง่ายในการติดตั้ง (Ease of Installation)	
	ระดับ 1	ยากและซับซ้อนมาก
	ระดับ 2	ยากในบางขั้นตอน
	ระดับ 3	ได้แต่ต้องปรับแก้
	ระดับ 4	ง่ายเกือบทั้งหมด
	ระดับ 5	ง่ายและรวดเร็ว
2	การพึ่งพาอินเทอร์เน็ต (Internet Dependency)	
	ระดับ 1	ต้องใช้อินเทอร์เน็ตตลอดเวลา
	ระดับ 2	พึ่งพาอินเทอร์เน็ตมาก
	ระดับ 3	พึ่งพาอินเทอร์เน็ตระดับปานกลาง
	ระดับ 4	ใช้อินเทอร์เน็ตเฉพาะบางขั้นตอน
	ระดับ 5	ไม่ต้องพึ่งพาอินเทอร์เน็ต
3	การจัดการหลายเวอร์ชัน Python (Multi-Version Management)	
	ระดับ 1	ไม่รองรับ
	ระดับ 2	รองรับบางส่วน
	ระดับ 3	รองรับได้แต่ไม่เสถียร
	ระดับ 4	รองรับได้ค่อนข้างดี
	ระดับ 5	รองรับหลายเวอร์ชันได้สมบูรณ์

ลำดับ	รายการความสามารถ และ เกณฑ์การประเมิน	
4	การติดตั้ง/จัดการแพ็กเกจ (Package Installation and Management)	
	ระดับ 1	ยากมาก ต้องติดตั้งเอง
	ระดับ 2	ยากและใช้เวลานาน
	ระดับ 3	พอจัดการได้
	ระดับ 4	จัดการแพ็กเกจได้สะดวก
	ระดับ 5	จัดการแพ็กเกจได้ง่ายและอัตโนมัติ
5	ความสามารถทำงานร่วมกัน (Collaboration Capability)	
	ระดับ 1	ไม่รองรับ
	ระดับ 2	รองรับน้อยมาก
	ระดับ 3	รองรับพื้นฐาน
	ระดับ 4	รองรับได้ดีในระดับห้องเรียน
	ระดับ 5	รองรับได้ดีมากและยืดหยุ่น
6	ความเสถียรของระบบ (System Stability)	
	ระดับ 1	ไม่เสถียร มีปัญหาบ่อย
	ระดับ 2	ค่อนข้างไม่เสถียร
	ระดับ 3	ปานกลาง
	ระดับ 4	เสถียรในระดับดี
	ระดับ 5	เสถียรมาก ใช้งานได้ต่อเนื่อง
7	ความง่ายในการเริ่มใช้งาน (Ease of Getting Started)	
	ระดับ 1	เริ่มใช้งานยากมาก
	ระดับ 2	เริ่มใช้งานค่อนข้างยาก
	ระดับ 3	เริ่มใช้งานได้แต่ใช้เวลานาน
	ระดับ 4	เริ่มใช้งานได้รวดเร็ว
	ระดับ 5	เริ่มใช้งานได้ทันที
8	ความยืดหยุ่นต่อการขยาย (Flexibility to Expand)	
	ระดับ 1	ไม่สามารถขยายได้
	ระดับ 2	ขยายได้แต่ยุ่งยาก
	ระดับ 3	ขยายได้บางส่วน
	ระดับ 4	ขยายได้ค่อนข้างยืดหยุ่น
	ระดับ 5	ขยายได้ง่ายและยืดหยุ่นสูง

การทดลองได้ดำเนินการติดตั้งและใช้งานเฟรมเวิร์กทั้ง 4 รูปแบบบนหลายระบบปฏิบัติการ ได้แก่ Windows, macOS, Linux และ Raspberry Pi โดยกลุ่มตัวอย่างประกอบด้วยผู้สอนและผู้เรียนในรายวิชาที่เกี่ยวข้องกับการเขียนโปรแกรมคอมพิวเตอร์และวิชาโครงงาน ซึ่งสะท้อนถึงการใช้งานจริงในบริบทการเรียนการสอน ข้อมูลที่เก็บรวบรวม ได้แก่ ระยะเวลาในการติดตั้ง ความเสถียรของระบบ ความสามารถในการทำซ้ำ (reproducibility) และความสะดวกในการใช้งาน จากนั้นจึงทำการวิเคราะห์ผลด้วยสถิติ เพื่อให้ได้ข้อสรุปเกี่ยวกับประสิทธิภาพของแต่ละเฟรมเวิร์ก

การดำเนินการประเมินผลในลักษณะดังกล่าวไม่เพียงช่วยยืนยันความถูกต้องของการออกแบบ MAPLE-J เท่านั้น แต่ยังสะท้อนให้เห็นถึงความแตกต่างเชิงคุณภาพระหว่างเฟรมเวิร์กที่มีใช้งานอยู่เดิมกับเฟรมเวิร์กที่ผู้วิจัยพัฒนาขึ้น



รูปที่ 6 ผลการประเมินประสิทธิภาพของเฟรมเวิร์ก MAPLE-J โดยเกณฑ์การประเมิน 8 มิติหลักเทียบทุกเฟรมเวิร์ก

7. สรุปผลการวิจัย

จากผลการประเมินตามเกณฑ์ 8 มิติที่กำหนดแสดงดังรูปที่ 6 พบว่าเฟรมเวิร์ก MAPLE-J มีประสิทธิภาพสูงกว่ารูปแบบเฟรมเวิร์กอื่นในหลายด้าน โดยเฉพาะในมิติที่เกี่ยวข้องกับการจัดการหลายเวอร์ชันของ Python การติดตั้งและจัดการแพ็กเกจ ความเสถียรของระบบ ความยืดหยุ่นต่อการขยาย และความสะดวกในการเริ่มต้นใช้งาน ซึ่งสะท้อนให้เห็นถึงข้อได้เปรียบของการออกแบบสคริปต์อัตโนมัติ (AuScript-Pyenv) ในการลดความซับซ้อนและเพิ่มความเสถียรของกระบวนการติดตั้งระบบ

เมื่อเปรียบเทียบกับ Google Colab แม้จะมีความโดดเด่นในด้านความง่ายในการติดตั้งและการทำงานร่วมกัน แต่ยังมีข้อจำกัดจากการพึ่งพาอินเทอร์เน็ตอย่างต่อเนื่อง ขณะที่ PN-Traditional และ PY-jupyterLAB แม้จะมีความสามารถในการจัดการสภาพแวดล้อมได้ระดับหนึ่ง แต่ยังขาดความยืดหยุ่นและความเสถียรในบางกรณี

โดยสรุปผลการวิจัยยืนยันว่า MAPLE-J สามารถตอบสนองต่อวัตถุประสงค์ของการพัฒนาได้อย่างมีประสิทธิภาพ ทั้งในด้านการจัดการสภาพแวดล้อม การเพิ่มความเสถียร ความสามารถในการทำซ้ำ (reproducibility) และการใช้งานในบริบทการเรียนการสอนที่หลากหลายระบบปฏิบัติการ ทั้ง Windows, macOS, Linux และ Raspberry Pi ได้อย่างเหมาะสมและมีศักยภาพในการนำไปประยุกต์ใช้งานจริงในวงกว้าง

8. อภิปรายผลการวิจัย

ผลการวิจัยแสดงให้เห็นว่า เฟรมเวิร์ก MAPLE-J สามารถตอบโจทย์ปัญหาการจัดการสภาพแวดล้อมในการเรียนการสอน Python และ AI ได้อย่างมีประสิทธิภาพ โดยรองรับการทำงานบนหลายระบบปฏิบัติการ ได้แก่ Windows, macOS, Linux และ Raspberry Pi ซึ่งเป็นข้อได้เปรียบในการใช้งานจริงที่มีความหลากหลายด้านฮาร์ดแวร์และซอฟต์แวร์ เมื่อเปรียบเทียบกับเฟรมเวิร์กแบบดั้งเดิม พบว่า MAPLE-J สามารถลดเวลาในการติดตั้งได้ใกล้เคียงกับ Google Colab แต่มีความแตกต่างที่สำคัญคือสามารถทำงานได้โดยไม่พึ่งพาการเชื่อมต่ออินเทอร์เน็ตอย่างต่อเนื่อง อีกทั้งยังมีความเสถียร ความสามารถในการทำซ้ำ และความยืดหยุ่นต่อการขยายที่สูงกว่า

ผลดังกล่าวสะท้อนถึงความสำเร็จในการออกแบบ สคริปต์อัตโนมัติ (AuScript-Pyenv) ซึ่งช่วยลดความซับซ้อนของกระบวนการติดตั้ง เพิ่มความน่าเชื่อถือ และทำให้ผู้สอนสามารถมุ่งเน้นไปที่การถ่ายทอดเนื้อหามากกว่าการแก้ไขปัญหาทางเทคนิค ในขณะที่ผู้เรียนสามารถเข้าถึง

เครื่องมือที่จำเป็นได้ทันที สิ่งนี้สอดคล้องกับแนวคิด การจัดการโครงสร้างพื้นฐานด้วยโค้ด และงานวิจัยด้านการวิจัยเชิงคำนวณที่สามารถทำซ้ำได้ที่เน้นการจัดการสภาพแวดล้อมด้วยโค้ดเพื่อลดความผิดพลาดและเพิ่มมาตรฐานของกระบวนการทดลอง

9. งานวิจัยในอนาคต

ตามที่ผู้วิจัยได้ดำเนินการวิจัยตามวิธีขั้นต้น พบว่าการศึกษาวิจัยดังกล่าวสามารถพัฒนาต่อยอดในมิติต่าง ๆ ได้อย่างหลากหลาย จึงขอเสนอเป็นแนวทางการพัฒนาต่อยอดในอนาคต ดังนี้

- เพิ่มความสามารถในการปรับแต่งการติดตั้งให้ยืดหยุ่น
- รองรับการใช้งานบนคลาวด์
- รองรับเวอร์ชันใหม่ของ Python และไลบรารีอย่างต่อเนื่อง
- เพิ่มไลบรารีด้าน Machine Learning และ Data Science อื่น ๆ
- ศึกษาผลกระทบต่อผลสัมฤทธิ์ทางการเรียนในระยะยาว
- เพิ่มระบบติดตามและรายงานการติดตั้งเพื่อปรับปรุงต่ออย่างต่อเนื่อง

เอกสารอ้างอิง

[1] Real Python, “Managing Multiple Python Versions With pyenv,” RealPython.com, Accessed: Aug. 5, 2025.
[2] pyenv, “Simple Python version management: pyenv,” GitHub repository, Accessed: Aug. 5, 2025.
[3] L. Aguilar, M. Gath-Morad, J. Grübel, J. Ermatinger, H. Zhao, S. Wehrli, R. W. Sumner, C. Zhang, D. Helbing, and C. Hölscher, “Experiments as Code: A Concept for Reproducible, Auditable, Debuggable, Reusable, & Scalable Experiments,” *arXiv preprint arXiv:2202.12050*, Feb. 2022. doi: 10.48550/arXiv.2202.12050.
[4] J. Harwell, L. Lowmanstone, and M. Gini, “SIERRA: A Modular Framework for Research Automation,” *arXiv preprint arXiv:2203.04748*, Mar. 2022, doi: 10.48550/arXiv.2203.04748.
[5] National Academies of Sciences, Engineering, and Medicine, *Automated Research Workflows for Accelerated Discovery: Closing the Knowledge Discovery Loop*, Washington, DC: National Academies Press, 2022. doi: 10.17226/26532.
[6] M. Ziemann, P. Poulain, and A. Bora, "The five pillars of computational reproducibility: bioinformatics and beyond," *Briefings in Bioinformatics*, vol. 24, no. 6, bbad375, Sep. 22, 2023, doi: 10.1093/bib/bbad375.
[7] S. H. Richter, "Automated Home-Cage Testing as a Tool to Improve Reproducibility of Behavioral Research?," *Frontiers in Neuroscience*, vol. 14, p. 383, 2020, doi: 10.3389/fnins.2020.00383.
[8] L. Vilhuber, H. H. Son, M. Welch, D. N. Wasser, and M. Darisse, "Teaching for large-scale Reproducibility Verification," *arXiv preprint arXiv:2204.01540*, submitted Mar. 2022, doi: 10.48550/arXiv.2204.01540.
[9] S. Dasgupta and P. Nuyujukian, "An open framework for archival, reproducible, and transparent science," *arXiv preprint arXiv:2504.08171*, Apr. 2025. [Online]. Available: https://doi.org/10.48550/arXiv.2504.08171
[10] A. Ustyuzhanin, T. D. Head, I. Babuschkin, and A. Tiunov, "Everware toolkit. Supporting reproducible science and challenge-driven education," *arXiv preprint arXiv:1703.01200*, Mar. 2017. [Online]. Available: https://doi.org/10.48550/arXiv.1703.01200
[11] J. R. Fonseca Cacho and K. Taghva, "The state of reproducible research in computer science," in Proc. 17th Int. Conf. Information Technology–New Generations (ITNG 2020), Las Vegas, NV, USA, 2020, pp. 519–524, doi: 10.1007/978-3-030-43020-7_68.

นายบุญนที ศักดิ์บุญญารัตน์ ปัจจุบันทำงานโรงเรียนมหิดลวิทยานุสรณ์ ตำแหน่งครูชำนาญการ สาขาวิชาคณิตศาสตร์และวิทยาการคำนวณ ปฏิบัติหน้าที่หัวหน้าฝ่ายบริหารเทคโนโลยีและนวัตกรรมดิจิทัล สำเร็จการศึกษาระดับปริญญาเอก สาขาวิชาเทคโนโลยีสารสนเทศ มหาวิทยาลัยศิลปากร สาขาที่เชี่ยวชาญ คือ Computer Vision, Biomedical Image Processing, Machine learning, Data Engineer, Parallel and High Performance Computing, Network Security, IoT Robotics Microcontroller

นางศิริพร ศักดิ์บุญญารัตน์ ปัจจุบันทำงานโรงเรียนมหิดลวิทยานุสรณ์ ตำแหน่งครูชำนาญการ สาขาวิชาคณิตศาสตร์และวิทยาการคำนวณ สำเร็จการศึกษาระดับปริญญาเอก สาขาวิชาเทคโนโลยีสารสนเทศ มหาวิทยาลัยศิลปากร สาขาที่เชี่ยวชาญ คือ Data mining, Course recommendation system, Massive open online course, Machine learning, Big data