⚡ iCafeFX × SiamCafe.net

Professional AI & Trading Tutorial Series · Est. 1997

🇹🇭 SOVEREIGN THAI AI

ThaiLLM

ฉบับสมบูรณ์

AI ไทย ฟรี · ใช้ใน VSCode + Python

— EDITION 2026 —

คู่มือระดับมืออาชีพสำหรับนักพัฒนาและผู้สนใจ AI ภาษาไทย

ตั้งแต่สมัคร API Key จนถึงเขียน Python เชื่อมต่อ

พร้อมเทคนิคขั้นสูง Thinking Mode · Tool Calling · RAG · Streaming

- 📘 **9 บท + 4 ภาคผนวก**
- 💻 **โค้ด Python ใช้งานจริง**
- 🔧 **Troubleshooting ครบ**
- 📊 **เทียบ 4 โมเดลละเอียด**
- 🇹🇭 **ข้อมูลอยู่ในไทย 100%**
- ⚡ **ใช้ LANTA Supercomputer**

เขียนและเรียบเรียงโดย

อาจารย์ บอม — SiamCafe.net × iCafeFX

icafeforex.com · siamcafe.net · xmsignal.com · youtube.com/@icafefx

📺 วิดีโอประกอบคู่มือ

ดูขั้นตอนจริงแบบทำตามได้ทันที บนช่อง iCafeFX

🇹🇭 ThaiLLM ใช้ฟรี ใน VSCode + Python

สแกน QR เพื่อดูวิดีโอ

เปิดด้วยกล้องโทรศัพท์

▶ YouTube

youtu.be/MCBwEexM5C4

สิ่งที่จะได้ในวิดีโอ

- ✅ สมัคร API Key ที่ thaillm.or.th
- ✅ ติดตั้ง Continue Extension ใน VSCode
- ✅ ตั้งค่า config.yaml แบบ step-by-step
- ✅ ทดสอบ Demo Python แปลภาษา
- ✅ รีวิว 4 โมเดลและวิธีเลือกใช้งาน

🔔 Subscribe @icafefx เพื่อติดตามคลิปใหม่

youtube.com/@icafefx · icafeforex.com · siamcafe.net

⏱️	🇹🇭	💰	💻
5 นาที	**ข้อมูลในไทย**	**ฟรี**	**VSCode**
ตั้งค่าเสร็จ	100%	ไม่มีค่าใช้จ่าย	+ Python

ข้อมูลเอกสาร

รายการ	รายละเอียด
ชื่อเอกสาร	ThaiLLM ฉบับสมบูรณ์ – AI ไทย ฟรี ใช้ใน VSCode + Python
เวอร์ชัน	v2.0 Professional Edition (2026)
วันที่เผยแพร่	เมษายน 2026
ภาษา	ไทย (มีศัพท์เทคนิคภาษาอังกฤษประกอบ)
จำนวนหน้า	40+ หน้า
ระดับความยาก	เริ่มต้น ถึง ระดับกลาง (Intermediate)
กลุ่มเป้าหมาย	นักพัฒนา Python, นักศึกษา IT, ผู้สนใจ AI, ผู้ประกอบการ
ผู้เขียน	อ.บอม – ผู้ก่อตั้ง SiamCafe.net (1997) และ iCafeFX
ประกอบ	วิดีโอ YouTube บนช่อง iCafeFX

ลิขสิทธิ์และการใช้งาน

⚠️ ข้อจำกัดความรับผิด (Disclaimer)

ThaiLLM เป็นโครงการร่วมของหน่วยงานวิจัยในประเทศไทย ข้อมูลเทคนิคและวิธีใช้ในเอกสารนี้อ้างอิงจากเอกสารทางการของโครงการ ณ เดือนเมษายน 2026 รายละเอียดบางส่วน (เช่น endpoint, ชื่อโมเดล, limit) อาจมีการปรับเปลี่ยนในอนาคต ผู้อ่านควรตรวจสอบเอกสารอย่างเป็นทางการที่ **thaillm.or.th** ประกอบการใช้งานจริงเสมอ

ผู้เขียนไม่มีส่วนเกี่ยวข้องกับการพัฒนา ThaiLLM โดยตรง เอกสารฉบับนี้จัดทำขึ้นในฐานะผู้ใช้งานและนักพัฒนาอิสระที่ต้องการเผยแพร่ความรู้ให้แก่ชุมชนนักพัฒนาไทย

ช่องทางติดต่อผู้เขียน

เว็บไซต์หลัก	icafeforex.com
คอมมูนิตี้	siamcafe.net (ก่อตั้ง พ.ศ. 2540)
Forex Signals	xmsignal.com
YouTube	youtube.com/@icafefx
Facebook	fb.com/icafefx

คำนำ

สวัสดีครับ เพื่อน ๆ นักพัฒนาและผู้สนใจเทคโนโลยีทุกท่าน

ในช่วงไม่กี่ปีที่ผ่านมา โมเดลภาษาขนาดใหญ่ (Large Language Model หรือ LLM) ได้กลายเป็นเครื่องมือที่เปลี่ยนแปลงวิธีที่เราทำงาน เขียนโค้ด เรียนรู้ และสร้างสรรค์ไปอย่างสิ้นเชิง อย่างไรก็ตาม โมเดลส่วนใหญ่ที่เราคุ้นเคยอย่าง ChatGPT, Claude, หรือ Gemini ล้วนเป็นของบริษัทต่างชาติ ข้อมูลที่ส่งไปจะถูกประมวลผลในเซิร์ฟเวอร์ต่างประเทศ และมีค่าใช้จ่ายสูงเมื่อใช้งานในระดับองค์กร

ThaiLLM คือคำตอบของประเทศไทย ต่อคำถามที่ว่า "ทำไมเราไม่มี AI เป็นของเราเอง?" โครงการนี้เกิดจากความร่วมมือของหน่วยงานระดับชาติ ทั้ง เนคเทค (NECTEC) วิสเทค (VISTEC) สมาคม AIEAT ธนาคารกสิกรไทยผ่าน KBTG และสถาบันข้อมูลขนาดใหญ่ BDI โดยใช้ซูเปอร์คอมพิวเตอร์ LANTA ในประเทศไทยในการฝึกโมเดล ข้อมูลทั้งหมดไม่ต้องส่งออกนอกประเทศ

เอกสารฉบับนี้เป็นคู่มือประกอบวิดีโอ YouTube ที่ผมเผยแพร่บนช่อง iCafeFX โดยตั้งใจให้ผู้อ่านสามารถทำตามได้ทีละขั้นตอน ตั้งแต่ทำความเข้าใจว่า ThaiLLM คืออะไร วิธีสมัคร API Key การตั้งค่า VSCode ไปจนถึงการเขียน Python เชื่อมต่อและสร้างแอปพลิเคชันจริง รวมถึง **เทคนิคขั้นสูงที่ไม่ค่อยมีคนพูดถึง** เช่น Thinking Mode, Tool Calling, Long Context และการทำ RAG กับข้อมูลภาษาไทย

เอกสารนี้เหมาะสำหรับใคร?

- **นักพัฒนา Python มือใหม่ถึงกลาง** ที่ต้องการเพิ่ม AI เข้าไปในงาน
- **นักศึกษา IT / Data Science** ที่สนใจ LLM ภาษาไทย
- **ผู้ประกอบการ SME** ที่ต้องการใช้ AI โดยไม่เสียค่า API ต่างชาติ
- **ผู้ที่ใส่ใจเรื่องความเป็นส่วนตัวและ PDPA** — ข้อมูลไม่ออกนอกประเทศ
- **นักวิจัย / อาจารย์** ที่ต้องการใช้ LLM ในการสอนหรือทำวิจัย

วิธีใช้คู่มือฉบับนี้

หากคุณเพิ่งเริ่มต้น แนะนำให้อ่านจากบทที่ 1 ถึง 7 ตามลำดับ เพราะเนื้อหาต่อเนื่องกัน หากคุณมีพื้นฐานอยู่แล้ว สามารถข้ามไปที่ **บทที่ 8 (เทคนิคขั้นสูง)** ซึ่งมีข้อมูลเชิงลึกที่หาได้ยาก และภาคผนวก A ซึ่งรวมโค้ดตัวอย่างใช้งานได้จริง

ท้ายสุดนี้ ขอขอบคุณทีมนักวิจัยไทยทุกท่านที่ทุ่มเทสร้าง ThaiLLM ขึ้นมา และขอบคุณผู้อ่านทุกท่านที่ให้ความสนใจเทคโนโลยีของคนไทยด้วยกัน

อ.บอม — *iCafeFX × SiamCafe.net*
เมษายน *2026*

สารบัญ

1 ทำไม AI ไทยถึงสำคัญ

ทำความเข้าใจบริบท ความสำคัญ และจุดแข็งของ *LLM* สัญชาติไทย

1.1 ภูมิทัศน์ LLM ในปัจจุบัน

ตั้งแต่ปี 2022 ที่ OpenAI เปิดตัว ChatGPT สู่สาธารณะ วงการเทคโนโลยีก็เปลี่ยนไปอย่างสิ้นเชิง LLM (Large Language Model) กลายเป็นสาธารณูปโภคพื้นฐานใหม่ของเศรษฐกิจดิจิทัล เหมือนที่ไฟฟ้าและอินเทอร์เน็ตเคยเป็นในยุคก่อน

แต่ในขณะที่เราใช้ ChatGPT, Claude, Gemini อย่างสะดวกสบาย เราลืมถามคำถามสำคัญว่า **ข้อมูลที่เราส่งไปถูกประมวลผลที่ไหน? เก็บโดยใคร? ใช้ทำอะไรต่อ?**

100+	704	8.15	87
พันล้านโทเคนไทย	NVIDIA A100 GPUs	PFlop/s Peak	วันในการฝึก 8B model

1.2 อธิปไตยของข้อมูล (Data Sovereignty)

ประเด็นสำคัญที่ทำให้ประเทศไทยต้องมี LLM ของตัวเองไม่ใช่แค่เรื่องค่าใช้จ่าย แต่คือเรื่อง "**อธิปไตยทางข้อมูล**" (Data Sovereignty) ซึ่งครอบคลุมถึง:

- **การกำหนดเงื่อนไขการจัดเก็บ** — ข้อมูลของคนไทยควรเก็บในประเทศไทย
- **การประมวลผล** — ข้อมูลไม่ถูกนำไปฝึกโมเดลของต่างชาติโดยไม่ได้รับอนุญาต
- **การเข้าถึง** — รัฐบาลไทยและหน่วยงานที่เกี่ยวข้องสามารถตรวจสอบได้
- **ความเป็นส่วนตัว** (PDPA) — สอดคล้องกับ พ.ร.บ. คุ้มครองข้อมูลส่วนบุคคลของไทย

> 💡 **ตัวอย่างการใช้งานที่ต้องใช้ AI ในประเทศ**
>
> 1. **ธนาคาร / การเงิน** — ข้อมูลลูกค้าไม่ควรส่งไปต่างประเทศตามข้อกำหนดของ ธปท.
> 2. **โรงพยาบาล** — ข้อมูลคนไข้เป็นข้อมูลที่มีความอ่อนไหวสูง
> 3. **หน่วยงานราชการ** — ข้อมูลประชาชน / ความลับราชการ
> 4. **บริษัทเอกชนที่มีข้อมูลลูกค้าไทย** — การปฏิบัติตาม PDPA

1.3 ไทยเป็น 1 ใน 4 ชาติเอเชียที่มี LLM ของตัวเอง

ในระดับภูมิภาคเอเชีย ประเทศที่มี Foundation LLM ของตัวเองในระดับที่ใช้งานจริงได้มีเพียง **จีน ญี่ปุ่น เกาหลีใต้ และไทย** เท่านั้น การที่ประเทศไทยเข้าร่วมกลุ่มนี้ได้แสดงถึงความก้าวหน้าของวงการ AI ไทยอย่างมีนัยสำคัญ

นี่ไม่ใช่เรื่องของการแข่งขันหรือความภาคภูมิใจเท่านั้น แต่คือเรื่องของความมั่นคงทางเทคโนโลยี (Technology Sovereignty) ที่ประเทศไทยจำเป็นต้องมีในยุคที่ AI กลายเป็นโครงสร้างพื้นฐานของเศรษฐกิจ

รูปที่ *1.1* – ประเทศในเอเชียที่มี *Foundation LLM* ของตัวเอง

2 รู้จัก ThaiLLM

เบื้องหลัง โครงสร้าง สถาปัตยกรรม และทีมที่สร้างมันขึ้นมา

2.1 เบื้องหลังโครงการและพันธมิตร

ThaiLLM เกิดจากความร่วมมือของหน่วยงานระดับชาติและสถาบันการศึกษาชั้นนำของไทย ภายใต้การประสานงานของ **สำนักงานพัฒนาวิทยาศาสตร์และเทคโนโลยีแห่งชาติ (สวทช./NSTDA)** โดยได้รับการสนับสนุนทุนวิจัยจาก **กองทุนพัฒนาดิจิทัลเพื่อเศรษฐกิจและสังคม (DEF)** ของสำนักงานคณะกรรมการดิจิทัลเพื่อเศรษฐกิจและสังคมแห่งชาติ

การที่หน่วยงานต่าง ๆ มารวมตัวกันในโครงการเดียวกันนี้ สะท้อนถึงความสำคัญและขนาดของโครงการ เพราะการสร้าง LLM ต้องใช้ทั้งทรัพยากรการคำนวณจำนวนมาก ข้อมูลคุณภาพสูง และทีมนักวิจัยที่เชี่ยวชาญ ซึ่งไม่มีองค์กรเดียวที่สามารถทำได้ครบทุกด้านโดยลำพัง

9 องค์กรพันธมิตรผู้พัฒนา ThaiLLM

#	องค์กร	บทบาทในโครงการ
1	**NECTEC** — ศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ	พัฒนาโมเดล Pathumma-ThaiLLM ผู้นำทีมวิจัยหลัก และดูแล LANTA supercomputer
2	**VISTEC** — สถาบันวิทยสิริเมธี	ข้อมูลภาษาไทยคุณภาพสูง และงานวิจัยด้าน NLP
3	**AIEAT** — สมาคมผู้ประกอบการ AI ประเทศไทย	พัฒนา OpenThaiGPT และเชื่อมต่อกับภาคเอกชน
4	**AIAT** — สมาคมปัญญาประดิษฐ์ประเทศไทย	งานวิจัยและประสานงานกลุ่มนักวิจัย
5	**KBTG** — Kasikorn Business-Technology Group	พัฒนา THaLLE (ทะเล) โมเดลเฉพาะทางด้านการเงิน
6	**SCB 10X** — ธนาคารไทยพาณิชย์ 10X	พัฒนา Typhoon-S โมเดลที่เน้น Sovereign AI
7	**BDI** — สถาบันข้อมูลขนาดใหญ่	การจัดการข้อมูลและ data pipeline
8	**จุฬาลงกรณ์มหาวิทยาลัย**	งานวิจัยและพัฒนาในเชิงวิชาการ
9	**มหาวิทยาลัยมหิดล**	งานวิจัยและทดสอบโมเดลในด้านต่าง ๆ

นอกจากนี้ ยังได้รับการสนับสนุนทรัพยากรการคำนวณสำหรับระบบให้บริการ (inference) จาก บริษัท **Siam.ai** และ **บริษัท โทรคมนาคมแห่งชาติ (NT)** ซึ่งช่วยให้ ThaiLLM Playground สามารถให้บริการฟรีแก่ผู้สนใจได้

2.2 ThaiLLM-8B Base Model

หัวใจของโครงการคือ `ThaiLLM-8B` ซึ่งเป็น **Foundation Model** หรือโมเดลพื้นฐานที่โมเดลอื่น ๆ นำไปต่อยอด (fine-tune) ได้ โดยเทคนิคที่ใช้คือ **Continued Pre-Training (CPT)** จาก Qwen3-8-Base ของอาลีบาบา

> 📝 **ทำไมต้องใช้ Qwen3 เป็นฐาน?**
>
> Qwen3 เป็นโมเดล open-source คุณภาพสูงที่ปล่อยโดย Alibaba Cloud มีจุดแข็งในการรองรับภาษาเอเชีย (รวมถึงภาษาไทย) และมีสถาปัตยกรรมที่ทันสมัยรองรับ **Thinking Mode** ซึ่งเป็นคุณสมบัติที่ช่วยให้โมเดลให้คำตอบที่แม่นยำกว่า การใช้ Qwen3 เป็นฐานจึงประหยัดทรัพยากรและเวลา โดยใช้จุดแข็งของโมเดลสากลมาผนวกกับจุดแข็งทางภาษาและบริบทไทย

สถาปัตยกรรมและการฝึก

Base architecture	Qwen3-8B (Transformer decoder-only, 8 พันล้านพารามิเตอร์)
Training method	Continued Pre-Training (CPT) + Supervised Fine-Tuning (SFT) + DPO
Thai training tokens	ประมาณ 63 พันล้านโทเคน (บน base) รวมกับข้อมูลในแต่ละ variant 100B+
Context length	32,768 tokens (native), ขยายได้ถึง 128K ด้วย YaRN
Tokenizer	Qwen3 tokenizer (รองรับภาษาไทยดีที่สุดในกลุ่ม open-source)
Primary languages	ไทย 🇹🇭 + อังกฤษ 🇬🇧
License	Qwen License (ใช้งานเชิงพาณิชย์ได้ ถ้า MAU ไม่เกิน 100 ล้าน)
Parameters	นอกจากรุ่น 8B ยังมีแผนพัฒนารุ่น 30B เพื่อใช้งานระดับ Enterprise

ข้อมูลที่ใช้ฝึก

- Wikipedia ภาษาไทย (ทั้งหมด)
- บทความข่าวจากสำนักข่าวในไทย (กรณี Nation Group ร่วมมือกับ BDI)
- วรรณกรรม หนังสือ และตำราเรียน
- เอกสารราชการและนโยบายภาครัฐ
- ข้อมูลสังเคราะห์ (synthetic data) ที่สร้างโดยโมเดลขนาดใหญ่
- โค้ดตัวอย่างพร้อมคำอธิบายภาษาไทย

2.3 LANTA Supercomputer — พลังการคำนวณที่อยู่เบื้องหลัง

การฝึกโมเดลขนาด 8 พันล้านพารามิเตอร์ต้องอาศัยพลังการคำนวณมหาศาล **LANTA** คือซูเปอร์คอมพิวเตอร์ที่ทรงพลังที่สุดของไทยในปัจจุบัน ตั้งอยู่ที่ **ศูนย์ทรัพยากรคอมพิวเตอร์เพื่อการคำนวณขั้นสูง (ThaiSC)** อุทยานวิทยาศาสตร์ประเทศไทย จังหวัดปทุมธานี

346	704	20,480	10 PB
Compute Nodes	NVIDIA A100 GPUs	CPU Cores	Storage

สเปกละเอียด

คุณสมบัติ	รายละเอียด
สถาปัตยกรรม	HPE Cray EX235n (heterogeneous cluster)
Peak Performance	8.15 PFlop/s (8.15 ล้านล้านล้านคำสั่งต่อวินาที)
GPU Nodes	176 nodes × 4 GPUs = 704 × NVIDIA A100 40GB
CPU	AMD EPYC 7713 64-core 2GHz (Zen 3, 3rd Gen)
High-memory nodes	10 nodes × 4 TB RAM
Network	HPE Slingshot 200 Gbps interconnect
Storage	Cray ClusterStor E1000: 10 PB HDD + 945 TB NVMe
Cooling	Liquid cooling (ประหยัดพลังงาน 33% เทียบ air-cooled)
TOP500 Ranking	อันดับ 70 ของโลก / อันดับ 1 ในอาเซียน
เปิดใช้งาน	2023

> ✅ **ตัวเลขที่น่าทึ่ง**
>
> การฝึก LLM ขนาด 8B parameter จากศูนย์บน **GPU ตัวเดียว** จะใช้เวลา **กว่า 160 ปี** หากใช้เครื่อง workstation 8 GPUs จะใช้เวลา **ประมาณ 20 ปี** แต่ด้วย LANTA (700 GPUs) **เหลือเพียง 87 วัน** — นี่คือพลังของ supercomputer ระดับชาติ

2.4 3 รูปแบบบริการของ ThaiLLM

ThaiLLM ให้บริการ 3 รูปแบบ เพื่อรองรับผู้ใช้งานหลากหลายกลุ่ม ทุกรูปแบบให้บริการฟรีในช่วงแรก

1. **🎮 Playground (สำหรับทุกคน)**

   เว็บแอปพลิเคชันที่ `thaillm.or.th` ให้ผู้สนใจเข้าไปทดลองสนทนากับโมเดลทั้ง 4 ตัวได้ทันทีผ่าน browser เหมาะสำหรับการลองใช้งานเบื้องต้น เปรียบเทียบคำตอบระหว่างโมเดล และประเมินความเหมาะสมก่อนตัดสินใจใช้งาน API

2. **🔌 API (สำหรับนักพัฒนา)**

   REST API ที่ **compatible กับ OpenAI SDK** — หมายความว่าโค้ด Python หรือ Node.js ที่คุณเขียนไว้สำหรับ OpenAI สามารถเปลี่ยนมาใช้ ThaiLLM ได้เพียงแก้ 2 บรรทัด (base_url + api_key) ช่วยให้ integrate เข้ากับระบบเดิมได้ง่ายมาก

3

💾 Model Download (**สำหรับนักวิจัย / Enterprise**)

สำหรับผู้ที่ต้องการ fine-tune โมเดลด้วยข้อมูลเฉพาะของตัวเอง หรือต้องการ deploy on-premise เพื่อความมั่นคงปลอดภัยสูงสุด สามารถดาวน์โหลดน้ำหนัก (weights) ของ ThaiLLM-8B base และโมเดล fine-tuned ทั้ง 4 ตัวได้ที่ Hugging Face ฟรี

3 4 โมเดลให้เลือกใช้

เปรียบเทียบจุดเด่น จุดด้อย และเหมาะกับงานแบบไหน

3.1 ตารางเปรียบเทียบโมเดลทั้ง 4 ตัว

คุณสมบัติ	Pathumma	OpenThaiGPT	Typhoon-S	THaLLE
ผู้พัฒนา	NECTEC (สวทช.)	AIEAT	SCB 10X	KBTG
Model ID	Pathumma-ThaiLLM-qwen3-8b-think-3.0.0	OpenThaiGPT-ThaiLLM-8B-Instruct-v7.2	Typhoon-S-ThaiLLM-8B-Instruct	THaLLE-0.2-ThaiLLM-8B-fa
ขนาด	8B parameters	8B parameters	8B parameters	8B parameters
จุดเด่น	Thinking Mode (คิดก่อนตอบ)	2M+ Thai instruction pairs	Sovereign AI Tool Calling	เชี่ยวชาญการเงิน (ผ่าน CFA)
Context	32K (YaRN → 128K)	32K (YaRN → 128K)	32K (YaRN → 128K)	32K (YaRN → 128K)
Thinking Mode	✅ เปิด-ปิดได้	❌ ไม่มี	✅ มี	✅ ปิดเป็น default
Tool Calling	✅ รองรับ	✅ รองรับ	✅ รองรับ (hermes)	✅ รองรับ
เหมาะกับ	Reasoning, Math, Code, การคิดซับซ้อน	Chatbot ทั่วไป, การถามตอบ	Agent, RAG, Workflow ซับซ้อน	งานการเงิน, วิเคราะห์หุ้น, การลงทุน
License	Apache 2.0 / Qwen	Qwen License	Research Preview	Apache 2.0

⚠️ **หมายเหตุ: Research Preview**

โมเดลส่วนใหญ่ใน Playground ยังเป็น *Research Preview* ซึ่งหมายความว่ายังอยู่ในช่วงทดสอบ อาจมีการปรับเปลี่ยนชื่อโมเดล endpoint หรือ behavior ได้ในอนาคต สำหรับงาน Production ควรอ่านเงื่อนไขการใช้งานของแต่ละโมเดลอย่างละเอียดก่อน

3.2 Pathumma — โมเดลที่คิดก่อนตอบ

🌸 Pathumma-ThaiLLM-qwen3-8b-think

NECTEC Thinking Recommended

Model ID	`Pathumma-ThaiLLM-qwen3-8b-think-3.0.0`
ผู้พัฒนา	ศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ (NECTEC)
เน้น	Reasoning, Math, Code, Logic
จุดเด่น	มี "Thinking Mode" — โมเดลจะ "คิด" ภายใน `<think>...</think>` blocks ก่อนตอบ ทำให้คำตอบแม่นยำกว่ามากในคำถามซับซ้อน

ชื่อ "ปทุมมา" มาจากดอกปทุมมา ซึ่งเป็นดอกไม้ประจำจังหวัดเชียงราย และสื่อถึง "ความเป็นไทย" ที่ปักรากลงในวงการ AI ระดับโลก

Pathumma ใช้เทคนิค **Chain-of-Thought (CoT)** ที่ฝังไว้ในโมเดลโดยตรง ทำให้ไม่ต้องเขียน prompt ยาว ๆ เพื่อให้โมเดลคิดทีละขั้น โมเดลจะทำเองอัตโนมัติ ผลคือ:

- **คำตอบแม่นยำกว่า** ในโจทย์ที่ต้องใช้เหตุผล (ถามสวัสดี ยังไงก็ตอบเหมือนเดิม แต่ถามปัญหาคณิตศาสตร์ เห็นชัดเจน)
- **ใช้โทเคนมากกว่า** เพราะต้อง "คิด" ก่อน ทำให้ช้ากว่าและเสียทรัพยากรมากกว่า
- **เหมาะกับ** งาน debug code, แก้โจทย์คณิตศาสตร์, วิเคราะห์เชิงเหตุผล, ไม่เหมาะกับ chatbot ที่ต้องตอบเร็ว

🔐 Thinking Mode Switch (เทคนิคที่น้อยคนรู้)

คุณสามารถ **เปิด/ปิด thinking mode** ได้โดยการเติม `/think` หรือ `/no_think` ในข้อความ prompt ตัวอย่าง:

`"คำนวณ 15 × 23 /think"` → โมเดลคิดละเอียด

`"สวัสดีครับ /no_think"` → ตอบเร็วเหมือน chatbot ทั่วไป

วิธีนี้ช่วยให้คุณคุมได้ว่าจะใช้ **คุณภาพ** (thinking on) หรือ **ความเร็ว** (thinking off) ในแต่ละ query เป็นรายกรณี รายละเอียดเพิ่มเติมที่บทที่ 8

3.3 OpenThaiGPT — แชทบอทเพื่อคนไทย

🇹🇭 OpenThaiGPT-ThaiLLM-8B-Instruct-v7.2

AIEAT Chat

Model ID	`OpenThaiGPT-ThaiLLM-8B-Instruct-v7.2`
ผู้พัฒนา	สมาคมผู้ประกอบการปัญญาประดิษฐ์ประเทศไทย (AIEAT)
เน้น	การสนทนาภาษาไทยธรรมชาติ, Q&A, Chatbot
จุดเด่น	Fine-tune บน Thai instruction dataset กว่า 2 ล้านคู่, ตอบเร็ว, เข้าใจคำศัพท์ไทยในชีวิตประจำวันดี

OpenThaiGPT เป็นโครงการ open-source ที่พัฒนามาหลายรุ่น (v1.0 → v1.5 → ปัจจุบัน v7.2) โดยมีเวอร์ชันสำหรับ ThaiLLM โดยเฉพาะที่ fine-tune บน ThaiLLM-8B base เหมาะสำหรับการใช้งานทั่วไป เช่น:

- Chatbot ถาม-ตอบภาษาไทย
- สรุปบทความ / สรุปข่าว
- แปลภาษา (ไทย ↔ อังกฤษ)
- เขียนอีเมล / จดหมายธุรกิจ
- ตอบคำถามในบริบทไทย เช่น กฎหมาย วัฒนธรรม ประเพณี

จุดเด่นที่ชัดเจนคือรองรับ **Tool Calling** และมี framework **OpenThaiRAG** ที่ใช้คู่กันได้ดี สำหรับการทำระบบถาม-ตอบบนข้อมูลเฉพาะ (เช่น เอกสารบริษัท)

3.4 Typhoon-S — Sovereign AI

🌀 Typhoon-S-ThaiLLM-8B-Instruct

SCB 10X | Sovereign | Tool

Model ID	`Typhoon-S-ThaiLLM-8B-Instruct`
ผู้พัฒนา	SCB 10X (ธนาคารไทยพาณิชย์)
เน้น	Sovereign AI, Agent workflows, Tool Calling, RAG
จุดเด่น	Training dataset, training code, และ technical report เปิดเผยทั้งหมด (fully open) รองรับ tool calling แบบ hermes parser

"S" ใน Typhoon-S ย่อมาจาก "**Sovereign**" — สะท้อนปรัชญาว่าประเทศต้องมี AI ของตัวเอง และต้องสามารถ reproduce ได้ทั้งหมด ตั้งแต่ข้อมูลฝึกสอนไปจนถึงการ post-training

Typhoon มีพัฒนามาหลายเวอร์ชัน — Typhoon 1.0 (7B), Typhoon 1.5 (8B/70B), Typhoon 2 (5 sizes + multimodal), และล่าสุดคือ Typhoon-S ที่ build บน ThaiLLM-8B โดยตรง โมเดลนี้มีชื่อเสียงในการทำงานกับ framework ที่ซับซ้อน เช่น:

- **Multi-agent workflow** — หลาย AI ทำงานประสานกัน
- **RAG (Retrieval-Augmented Generation)** — ดึงข้อมูลจาก database มาตอบ
- **Function Calling** — สั่งให้ AI เรียกใช้ function ที่เราเขียน เช่น ดึงอุณหภูมิ ราคาหุ้น ฯลฯ

3.5 THaLLE (ทะเล) — AI การเงินระดับ CFA

🌊 THaLLE-0.2-ThaiLLM-8B-fa

KBTG | Finance | CFA

Model ID	`THaLLE-0.2-ThaiLLM-8B-fa`
ผู้พัฒนา	KBTG Labs (Kasikorn Business-Technology Group)
เน้น	งานการเงิน การลงทุน การวิเคราะห์
จุดเด่น	โมเดลแรกจากบริษัทไทยที่ผ่านการสอบ mock CFA (Chartered Financial Analyst), ฝึกด้วย Investment Consultant exam จาก SET

THaLLE ย่อมาจาก **T**ext **H**yperlocally **A**ugmented **L**arge **L**anguage **E**xtension พ้องเสียงกับ "ทะเล" ในภาษาไทย "fa" ต่อท้ายคือ **Financial Analyst**

จุดแข็งที่โดดเด่น:

- **เข้าใจภาษาทางการเงินอย่างลึกซึ้ง** — ศัพท์เฉพาะ เช่น NPV, IRR, P/E ratio, บอนด์ หุ้นกู้ ฯลฯ
- **ผ่านสอบเทียบเท่า CFA** ระดับมืออาชีพสากล
- **ผ่านสอบ Investment Consultant (IC) ของ SET** — P1 72%, P2 72%, P3 84%
- **ช่วยวิเคราะห์ข้อมูลการเงิน** ประเมินความเสี่ยง พัฒนาโอกาสการลงทุน

เหมาะสำหรับ: ที่ปรึกษาการเงิน, นักวิเคราะห์หุ้น, พนักงานธนาคาร, นักลงทุนรายย่อยที่ต้องการที่ปรึกษา AI ที่เข้าใจบริบทการเงินไทย

3.6 วิธีเลือกโมเดลให้เหมาะกับงาน

✅ **คู่มือเลือกโมเดลอย่างรวดเร็ว**

ต้องการคำตอบแม่นยำสูง / งานคิดซับซ้อน / คณิตศาสตร์ → Pathumma (thinking on)

แชทบอททั่วไป / ตอบเร็ว / ภาษาธรรมชาติ → OpenThaiGPT

Agent / RAG / Tool Calling / Production workflow → Typhoon-S

งานการเงิน / ลงทุน / วิเคราะห์หุ้น → THaLLE

ทดลองเปรียบเทียบทุกตัว → สลับใน config.yaml โดยเปลี่ยน model name เท่านั้น (base URL เหมือนกัน)

4 สมัคร API Key

ขั้นตอนละเอียดทีละขั้น พร้อมเทคนิคด้าน *Security*

4.1 ไปที่ thaillm.or.th

เปิด browser (Chrome / Edge / Firefox) แล้วพิมพ์ **thaillm.or.th** ในช่อง address ไม่ต้องใส่ https หน้าเว็บจะเปิดเป็นภาษาไทย หน้าตาสะอาด มีปุ่มสำคัญ 3 ปุ่ม:

- `Playground` — ทดลอง chat กับโมเดล
- `API` — สำหรับนักพัฒนา (คลิกที่มุมขวาบน)
- `Documentation` — คู่มือการใช้งาน

4.2 Login ด้วยบัญชี Google หรือ Email

กดปุ่ม **API Key** ที่มุมขวาบน ระบบจะขอให้ login ตัวเลือก:

- **Sign in with Google** (แนะนำ — รวดเร็วที่สุด)
- **Sign up with Email** (ใช้ในกรณีที่ไม่มี Google account)

สำหรับการใช้งานในองค์กร ควรสร้างบัญชีด้วย email ของบริษัทเพื่อให้ทีมอื่นสามารถจัดการ API key ต่อได้เมื่อคุณไม่อยู่

4.3 สร้าง API Key

1. **เข้าหน้า Dashboard**

   หลัง login จะเข้าสู่หน้าจัดการ API key ของคุณ หากเพิ่งสมัครใหม่ จะเห็นว่ายังไม่มี key ใด ๆ

2. **กด "Create API Key"**

   จะมี popup ให้ใส่ **ชื่อ key** (เช่น "youtube-tutorial", "my-laptop", "production-server") ใส่ชื่อที่อธิบายการใช้งาน เพื่อให้จัดการได้ง่ายภายหลัง

3. **Copy Key ทันที!**

   ระบบจะแสดง key ขึ้นมาเพียง **ครั้งเดียว** — หากปิดหน้าต่างไปโดยไม่ copy จะต้องสร้าง key ใหม่ key มีรูปแบบประมาณ:

```
tllm_sk-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

4

เก็บ key ในที่ปลอดภัย

ห้าม! paste key ลงใน chat, GitHub public repo, หรือ screenshot เด็ดขาด เพราะจะทำให้คนอื่นใช้ได้ วิธีที่ถูกต้องคือเก็บใน `.env` file (ดูหัวข้อ 4.4)

4.4 Security Best Practices — จัดการ API Key อย่างมืออาชีพ

สิ่งที่ห้ามทำเด็ดขาด

- ห้าม commit key ลง git repository (ใช้ `.gitignore` เสมอ)
- ห้าม paste key ใน chat, Line, Facebook หรือ public forum
- ห้าม hardcode key ในโค้ด production
- ห้าม share key กับเพื่อนร่วมงานผ่าน email ให้ใช้ Password Manager เช่น Bitwarden / 1Password แทน
- หาก key หลุด ให้เข้าไป **Revoke** ทันที แล้วสร้างใหม่

วิธีที่ถูกต้อง: ใช้ .env file

สร้างไฟล์ชื่อ `.env` ในโฟลเดอร์โปรเจกต์:

```
# .env (ห้าม commit ไฟล์นี้!)
THAILLM_API_KEY=tllm_sk-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
THAILLM_BASE_URL=https://api.thaillm.or.th/v1
```

และเพิ่มบรรทัดนี้ใน `.gitignore` :

```
# .gitignore
.env
.env.local
*.key
```

ใน Python ใช้ `python-dotenv` โหลดค่า:

```
# pip install python-dotenv
from dotenv import load_dotenv
import os

load_dotenv()

api_key = os.getenv("THAILLM_API_KEY")
base_url = os.getenv("THAILLM_BASE_URL")
```

การจัดการหลาย key (key rotation)

สำหรับระบบ production ควรทำ **key rotation** ทุก 90 วัน:

1. **สร้าง key ใหม่**

   ก่อนที่ key เก่าจะหมดอายุ/ถูก revoke สร้าง key ใหม่คู่ขนานกันไว้

2. **Deploy key ใหม่ไป production**

   อัปเดต environment variable ใน server / container / CI/CD

3. **ทดสอบให้แน่ใจว่า key ใหม่ทำงาน**

   เช็ค log, monitoring, healthcheck ก่อน revoke key เก่า

4. **Revoke key เก่า**

   เข้าไปที่ thaillm.or.th/api → Manage Keys → กด Revoke

— บทพิเศษ · PART II —

VSCode

ครบวงจร ฉบับมืออาชีพ

จากศูนย์ถึงมือโปร — ติดตั้ง, *UI, Extensions, Python, Debugger, Terminal, Git, AI Coding*

📚 ในบทนี้คุณจะได้เรียนรู้

- ✓ VSCode คืออะไร? ต่างจาก Visual Studio อย่างไร
- ✓ ติดตั้งบน Windows / macOS / Linux
- ✓ เข้าใจ UI ทุกส่วน — Activity Bar, Explorer, ฯลฯ
- ✓ Command Palette หัวใจของ VSCode
- ✓ Keyboard Shortcuts 60+ ตัว
- ✓ Extensions ที่ต้องมี 15 ตัว
- ✓ Python setup + Virtual Environment
- ✓ Pylance IntelliSense
- ✓ Debugger + launch.json
- ✓ Integrated Terminal
- ✓ Git Source Control
- ✓ Settings Sync & Profiles

⚡ iCafeFX × SiamCafe.net — Professional Edition

5A VSCode คืออะไร?

ทำความรู้จักเครื่องมือที่นักพัฒนากว่า *73%* ทั่วโลกเลือกใช้

5A.1 ภาพรวม

Visual Studio Code (ย่อว่า VSCode หรือ VS Code) คือ *source code editor* หรือโปรแกรมแก้ไขโค้ด ที่พัฒนาโดย **Microsoft** และเปิดตัวครั้งแรกในปี 2015 ปัจจุบันเป็นเครื่องมือที่ได้รับความนิยมสูงสุดในวงการนักพัฒนาทั่วโลก (ตาม Stack Overflow Developer Survey)

VSCode อธิบายตัวเองว่า *"a standalone source code editor that runs on Windows, macOS, and Linux"* — โปรแกรมแก้ไขโค้ดอิสระ ที่ทำงานได้บน Windows, macOS และ Linux ขนาดการดาวน์โหลดเล็กกว่า 200 MB และใช้พื้นที่ดิสก์น้อยกว่า 500 MB ทำให้ติดตั้งได้เร็วและเบากว่า IDE แบบเต็ม

> 🆓 **ฟรี 100% และ Open Source**
>
> VSCode เป็นซอฟต์แวร์ **ฟรี** สำหรับการใช้งานส่วนตัวและเชิงพาณิชย์ ไม่มีเวอร์ชัน "Pro" ที่ต้องจ่ายเงิน และ source code เปิดให้ทุกคนเข้าไปดูและแก้ไขได้ที่ github.com/microsoft/vscode ภายใต้ MIT license

5A.2 Editor vs IDE ต่างกันอย่างไร?

นักพัฒนาใหม่มักสับสนระหว่าง **Code Editor** และ **IDE** (Integrated Development Environment) VSCode อยู่ตรงกลางระหว่างทั้งสองแบบ

คุณสมบัติ	Text Editor	VSCode	Full IDE
ตัวอย่าง	Notepad, Sublime	VSCode	Visual Studio, IntelliJ
ขนาด	< 10 MB	~200 MB	2-10 GB
ความเร็วเปิด	ทันที	1-3 วินาที	10-60 วินาที
Syntax Highlight	✅ มี	✅ มีครบทุกภาษา	✅ มีครบ
Autocomplete	❌ ไม่มี	✅ IntelliSense ฉลาด	✅ ฉลาดที่สุด
Debugger	❌ ไม่มี	✅ มีผ่าน extension	✅ built-in ทรงพลัง
Git integration	❌ ไม่มี	✅ built-in	✅ built-in
Refactoring	❌ ไม่มี	✅ ดี	✅ ดีมาก
รองรับภาษา	ทุกภาษา (แสดง)	ทุกภาษา (ผ่าน ext)	เฉพาะบางภาษา
ราคา	ฟรี	ฟรี	มักเสียเงิน

🎯 **สรุปตำแหน่งของ VSCode**

VSCode = "**ผสมข้อดีของ Text Editor (เบา เร็ว ใช้ง่าย) กับ IDE (ฉลาด มี features ครบ) ไว้ด้วยกัน**" ด้วยการที่ features แต่ละอย่างถูกทำเป็น **Extension** คุณติดตั้งเฉพาะที่ใช้ ทำให้ VSCode ไม่บวมเหมือน IDE เต็มตัว

5A.3 VSCode vs Visual Studio — อย่าสับสน!

ชื่อคล้ายกันแต่เป็น **คนละโปรแกรม** เลย นี่คือจุดที่มือใหม่มักสับสนที่สุด

ประเด็น	Visual Studio Code (VSCode)	Visual Studio
ประเภท	Code Editor (น้ำหนักเบา)	Full IDE (สมบูรณ์แบบ)
OS	Windows, macOS, Linux	Windows (มี Mac แยก)
ขนาด	~200 MB	2-10 GB
เป้าหมายหลัก	Web Dev, Python, ทุกภาษา	C#, .NET, Enterprise
ความเร็ว	เปิด 1-3 วิ	เปิด 10-60 วิ
ราคา	ฟรี 100%	Community ฟรี, Pro/Ent เสียเงิน
โค้ดเปิด	Open Source (MIT)	Proprietary

5A.4 ทำไมต้องเรียน VSCode?

1 **มาตรฐานของวงการ**

บริษัท IT ส่วนใหญ่ใช้ VSCode เป็น default หากคุณทำงานในทีม การใช้เครื่องมือเดียวกันช่วยให้ทำงานร่วมกันได้ง่ายขึ้น – share settings, extensions, และทำ pair programming ได้

2 **ใช้ได้กับทุกภาษา**

Python, JavaScript, Java, C++, Go, Rust, PHP, Ruby, Swift, SQL, HTML, CSS, YAML, JSON, Markdown, ฯลฯ – เรียนครั้งเดียว ใช้ได้ตลอดอาชีพ ไม่ต้องเปลี่ยน editor เมื่อเปลี่ยนภาษา

3 **Extensions > 50,000 ตัว**

ไม่ว่าคุณจะต้องการ feature อะไร มักมี extension รองรับแล้ว: แก้ไข Docker, SSH ไป server, preview เว็บ, AI ช่วยเขียนโค้ด, custom theme, keyboard bindings แบบ Vim/Emacs ฯลฯ

4 **AI-ready (ปี 2024+)**

VSCode รองรับการทำงานกับ AI coding assistants อย่างเป็นทางการ ทั้ง GitHub Copilot (เสียเงิน), Continue (ฟรี, ใช้ ThaiLLM ได้), Cline, Cursor ทำให้เป็น **platform มาตรฐานสำหรับ AI-powered coding**

5 **Community ยักษ์ใหญ่**

ถามปัญหาใน Stack Overflow, Reddit, YouTube มีคนตอบตลอด หา tutorial ฟรีได้หลายพันคลิป บน iCafeFX เองก็มีคลิปสอน VSCode หลายคลิป

5B ติดตั้ง VSCode

Windows · macOS · Linux — ครอบคลุมทุกแพลตฟอร์ม

5B.1 ความต้องการของระบบ (System Requirements)

OS	ความต้องการขั้นต่ำ
Windows	Windows 10 / 11 (64-bit) — RAM 2 GB+, Disk 500 MB+
macOS	macOS 11 (Big Sur) ขึ้นไป — Intel หรือ Apple Silicon (M1/M2/M3/M4)
Linux	Ubuntu 20.04+, Debian 10+, Fedora 34+, glibc 2.28+ — RAM 2 GB+

> 💡 **เครื่องเก่าก็ใช้ได้**
> VSCode เป็นดาวน์โหลดขนาดเล็ก (น้อยกว่า 200 MB) และใช้พื้นที่ดิสก์น้อยกว่า 500 MB แม้เครื่องเก่า 4-8 GB RAM ก็ใช้ได้สบาย ตราบใดที่ OS ยังรองรับ

5B.2 ติดตั้งบน Windows

1

ดาวน์โหลดไฟล์ติดตั้ง

เปิด browser ไปที่ code.visualstudio.com แล้วกดปุ่มสีน้ำเงิน `Download for Windows` ระบบจะดาวน์โหลดไฟล์ `VSCodeUserSetup-x64-<version>.exe` ขนาดประมาณ 95 MB

2

เลือกประเภทการติดตั้ง

Microsoft มี 2 แบบ:

- **User Installer** (แนะนำ) — ติดตั้งเฉพาะ user ของคุณ ไม่ต้องใช้ admin
- **System Installer** — ติดตั้งทั้งเครื่อง ต้องใช้ admin password (เหมาะกับ shared computer)

3

Run Installer

Double-click ไฟล์ที่ดาวน์โหลด (VSCodeUserSetup-{version}.exe) โดย default, VSCode จะถูกติดตั้งไว้ที่ C:\Users\{Username}\AppData\Local\Programs\Microsoft VS Code

4 **เลือก "Additional Tasks" ให้ถูก** ⭐

ขั้นตอนสำคัญที่สุด — เลือก **ทุกช่องให้หมด**:

- ✅ Create a desktop icon
- ✅ Add "Open with Code" action to Windows Explorer file context menu
- ✅ Add "Open with Code" action to Windows Explorer directory context menu
- ✅ Register Code as an editor for supported file types
- ✅ **Add to PATH** (สำคัญที่สุด) — เพื่อให้เรียก `code` จาก Terminal ได้

5 **Install และ Launch**

กด Install รอ 30 วินาที – 1 นาที เสร็จแล้วกด Finish ติ๊ก "Launch Visual Studio Code" VSCode จะเปิดขึ้นมา

ทางเลือก: ติดตั้งผ่าน Winget (Windows 11)

หากคุณใช้ Windows 11 หรือ Windows 10 ที่ติดตั้ง winget แล้ว สามารถใช้คำสั่งเดียวได้:

```
# เปิด PowerShell หรือ Terminal
winget install Microsoft.VisualStudioCode
```

5B.3 ติดตั้งบน macOS

1 **ดาวน์โหลด .zip**

ไปที่ code.visualstudio.com กด Download for Mac ระบบจะเลือก Intel หรือ Apple Silicon (M1/M2/M3/M4) ให้อัตโนมัติ

2 **Extract และย้ายไป Applications**

เปิดไฟล์ `.zip` จะได้ app `Visual Studio Code.app` — ลากไปใส่โฟลเดอร์ **Applications**

3 **เปิดครั้งแรก (ข้าม gatekeeper)**

หาก macOS เตือนว่า "can't be opened because it is from an unidentified developer" ให้คลิกขวาที่ app แล้วเลือก **Open** (แทนที่จะ double-click) จากนั้นยืนยันด้วย Open อีกครั้ง

4 **เพิ่ม 'code' command ลงใน shell**

เพื่อให้เรียก VSCode จาก Terminal ได้ เปิด VSCode แล้วกด `Cmd+Shift+P` พิมพ์ `Shell Command: Install 'code' command in PATH` กด Enter ใส่รหัสผ่าน macOS

```
# ทดสอบใน Terminal
code --version
# ต้องแสดง version เช่น 1.91.0
```

5B.4 ติดตั้งบน Linux (Ubuntu/Debian)

```
# วิธีที่ 1: ดาวน์โหลด .deb แล้ว install (ง่ายที่สุด)
wget -O vscode.deb "https://code.visualstudio.com/sha/download?build=stable&os=linux-deb-x64"
sudo apt install ./vscode.deb

# วิธีที่ 2: ผ่าน APT repository (ได้ auto-update)
sudo apt-get install wget gpg
wget -qO- https://packages.microsoft.com/keys/microsoft.asc | gpg --dearmor > packages.microsoft.gpg
sudo install -D -o root -g root -m 644 packages.microsoft.gpg /etc/apt/keyrings/packages.microsoft.gpg
echo "deb [arch=amd64,arm64,armhf signed-by=/etc/apt/keyrings/packages.microsoft.gpg] https://packages.microsoft.com/repos/code stable main" | sudo tee /etc/apt/sources.list.d/vscode.list > /dev/null
sudo apt update
sudo apt install code

# วิธีที่ 3: Snap
sudo snap install code --classic
```

Fedora / RHEL / CentOS

```
sudo rpm --import https://packages.microsoft.com/keys/microsoft.asc
sudo sh -c 'echo -e "[code]\nname=Visual Studio Code\nbaseurl=https://packages.microsoft.com/yumrepos/vscode\nenabled=1\ngpgcheck=1\ngpgkey=https://packages.microsoft.com/keys/microsoft.asc" > /etc/yum.repos.d/vscode.repo'
sudo dnf install code
```

5B.5 Update และ Auto-Update

VSCode ออกเวอร์ชันใหม่ทุกเดือน และรองรับ auto-update เมื่อมีเวอร์ชันใหม่ออก โดย default VSCode จะแจ้งเตือนให้ update อัตโนมัติ

OS	วิธี Update Manual
Windows / Linux	เมนู **Help** > **Check for Updates**
macOS	เมนู **Code** > **Check for Updates**

5B.6 Portable Mode — VSCode ใน USB

VSCode รองรับ **Portable Mode** — สามารถใส่ใน USB แล้วพกพาไปใช้ที่ไหนก็ได้ โดย settings ติดมาทั้งหมด เหมาะสำหรับ:

- คอมที่ไม่มีสิทธิ์ admin

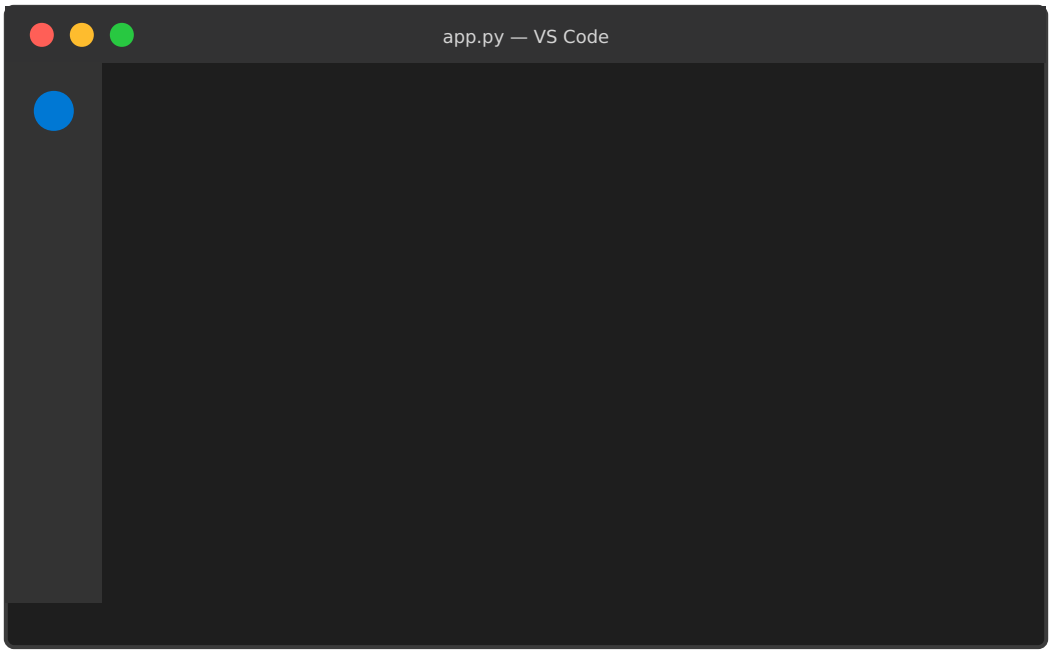
- ทำงานในร้านเน็ต / ห้องแล็บมหาวิทยาลัย
- ต้องการพา environment ติดตัวไปที่ไหนก็ได้

วิธีใช้: ดาวน์โหลด .zip (ไม่ใช่ installer) แตกไฟล์ลง USB แล้วสร้างโฟลเดอร์ `data` (หรือ `code-portable-data`) ไว้ข้าง `code.exe` ทุก config/extensions จะเก็บในโฟลเดอร์นั้น

5C เรียนรู้ UI ของ VSCode

ทำความเข้าใจทุกส่วนของหน้าจอ ใช้ได้อย่างมืออาชีพใน *15* นาที

5C.1 ภาพรวม UI ทั้งหมด



รูปที่ *5C.1* — หน้าจอ *VSCode* พร้อมคำอธิบายแต่ละส่วน *(5* พื้นที่หลัก*)*

5C.2 พื้นที่หลัก 5 ส่วน

① Activity Bar (แถบซ้ายสุด)

แถบบาง ๆ ที่ขอบซ้ายสุด มีไอคอน 6 ตัวเป็นหลัก กดสลับมุมมองได้:

ไอคอน	ชื่อ	Shortcut	ใช้ทำอะไร
📁 Explorer	File Explorer	`Ctrl+Shift+E`	ดูและจัดการไฟล์/โฟลเดอร์ในโปรเจกต์
🔍 Search	Search	`Ctrl+Shift+F`	ค้นหาข้อความในทุกไฟล์ของโปรเจกต์
⑂ Git	Source Control	`Ctrl+Shift+G`	Git: commit, push, pull, merge
🐞 Debug	Run and Debug	`Ctrl+Shift+D`	Debug โค้ด ตั้ง breakpoint
⊞ Ext	Extensions	`Ctrl+Shift+X`	ติดตั้ง / จัดการ extensions
⚙ Settings	Manage	–	ตั้งค่า, Keyboard Shortcuts, Profiles

> 💡 **ปรับแต่ง Activity Bar**
> คลิกขวาที่ไอคอนใด ๆ เพื่อซ่อน/แสดง หรือย้ายไปขอบขวาแทน (Move Activity Bar Right) — บางคนชอบซ้าย บางคนชอบขวา

② Side Bar (แถบข้าง)

พื้นที่กว้างกว่าข้าง Activity Bar จะแสดงเนื้อหาตามไอคอนที่กด เช่น กด Explorer จะเห็น tree ของไฟล์ กด Extensions จะเห็นรายการ extensions

ซ่อน/แสดง Side Bar: `Ctrl+B` (Windows/Linux) หรือ `Cmd+B` (Mac) — เหมาะเมื่อต้องการพื้นที่เขียนโค้ดเพิ่ม

③ Editor Area (พื้นที่หลัก)

พื้นที่ที่คุณเขียนโค้ดจริง ๆ features สำคัญ:

- **Tabs** — เปิดหลายไฟล์พร้อมกันได้ กดเลข `Ctrl+1/2/3` เพื่อสลับกลุ่ม
- **Split View** — แบ่งจอเป็น 2-4 ช่อง `Ctrl+\\`
- **Minimap** — แผนที่โค้ดย่อด้านขวา คลิกเพื่อเลื่อนเร็ว
- **Breadcrumbs** — แถบบนสุดบอก path ของไฟล์ + ฟังก์ชันที่อยู่

④ Panel (พื้นที่ล่าง)

มี 4 tab หลักในด้านล่าง:

Tab	Shortcut	ใช้ทำอะไร
Problems	`Ctrl+Shift+M`	รายการ errors/warnings ในโค้ด
Output	`Ctrl+Shift+U`	log ของ extensions เช่น Python, Git
Debug Console	`Ctrl+Shift+Y`	เขียน expression ดู variable ตอน debug
Terminal	`Ctrl+`` (backtick)	ใช้ command line (PowerShell/bash/zsh)

> 🔧 **Terminal Tip**
> สามารถเลือก shell อื่นได้จาก dropdown ใน terminal (PowerShell, CMD, Git Bash, WSL) ใน macOS/Linux มี zsh/bash ตาม default

⑤ Status Bar (แถบล่างสุด)

แถบสีฟ้า/ม่วงด้านล่าง แสดงข้อมูลสำคัญ (จากซ้ายไปขวา):

- **Git branch** — branch ปัจจุบัน (คลิกเพื่อสลับ)
- **Problems count** — จำนวน error/warning
- **Line/Column** — ตำแหน่ง cursor (คลิกเพื่อ Go to Line)

- **Encoding** — UTF-8, UTF-16 ฯลฯ
- **Line Ending** — LF (Unix), CRLF (Windows)
- **Language Mode** — Python, JavaScript, JSON ฯลฯ
- **Python Interpreter** (ถ้าติดตั้ง Python extension) — คลิกเพื่อเปลี่ยน version
- **Notifications** (กระดิ่ง)

5D Command Palette & Shortcuts

หัวใจของความเร็ว — เรียนรู้ *shortcut 60+* ตัวแบบจัดกลุ่ม

5D.1 Command Palette — คำสั่งเดียวที่ต้องจำ

⭐ **ถ้าจำได้แค่ shortcut เดียว จำอันนี้!**

กด Ctrl+Shift+P (Windows/Linux) หรือ Cmd+Shift+P (macOS) เพื่อเปิด Command Palette — ให้คุณเข้าถึงทุกฟังก์ชันของ VSCode ทำเกือบทุกอย่างด้วย keyboard อย่างเดียว

`Ctrl` + `Shift` + `P`

วิธีใช้ Command Palette

1. **กด shortcut เปิด**

   `Ctrl+Shift+P` จะมีกล่องโผล่ขึ้นตรงบนสุดของหน้าจอ

2. **พิมพ์สิ่งที่อยากทำ (ไทย/อังกฤษ)**

   เช่น พิมพ์ `format` จะเห็น "Format Document" / พิมพ์ `theme` จะเห็น "Color Theme"

3. **กด Enter**

   คำสั่งทำงานทันที ไม่ต้องหา menu หลายชั้น

Command Palette แบบต่าง ๆ

อาการ	เริ่มด้วย	ทำอะไร
ค้นหาคำสั่งทั่วไป	ว่างเปล่า	เมนูคำสั่งทั้งหมด (ค่า default)
เปิดไฟล์ตามชื่อ	ไม่มี `>`	Go to File (Quick Open) — `Ctrl+P`
ดู symbol/function ในไฟล์	`@`	ไปที่ฟังก์ชัน / class ในไฟล์ปัจจุบัน
ดู symbol ในทั้ง workspace	`#`	หา symbol ข้าม file ได้
กระโดดไปบรรทัด	`:`	Go to Line (เช่น `:42`)
ดู command ล่าสุด	`>`	คำสั่งที่เคยใช้
ช่วยเหลือ / Help	`?`	แสดง tips การใช้ command palette

5D.2 Keyboard Shortcuts 60+ ตัวที่ต้องรู้

จัดกลุ่มให้ง่ายต่อการจำ สำหรับ Mac ให้แทน `Ctrl`→`Cmd` และ `Alt`→`Option`

📁 ไฟล์ (File Management)

สร้างไฟล์ใหม่	`Ctrl+N`	เปิดไฟล์	`Ctrl+O`
เปิดโฟลเดอร์	`Ctrl+K Ctrl+O`	Save	`Ctrl+S`
Save As	`Ctrl+Shift+S`	Save All	`Ctrl+K S`
ปิด tab	`Ctrl+W`	เปิด tab ที่เพิ่งปิด	`Ctrl+Shift+T`

📝 แก้ไขโค้ด (Editing)

Cut/Copy/Paste	`Ctrl+X/C/V`	Undo / Redo	`Ctrl+Z / Ctrl+Y`
ย้ายบรรทัดขึ้น/ลง ⭐	`Alt+↑/↓`	ก็อปบรรทัด ⭐	`Shift+Alt+↑/↓`
ลบทั้งบรรทัด	`Ctrl+Shift+K`	แทรกบรรทัดข้างล่าง	`Ctrl+Enter`
Comment บรรทัด	`Ctrl+/`	Comment แบบ block	`Shift+Alt+A`
Indent	`Tab`	Unindent	`Shift+Tab`
Format Document ⭐	`Shift+Alt+F`	Rename Symbol	`F2`

🖱️ Multi-cursor & Selection (จัดเด่นมาก)

เลือกทั้งบรรทัด	`Ctrl+L`	เลือกคำถัดไป (เหมือน)	`Ctrl+D`
เลือก occurrence ทั้งหมด	`Ctrl+Shift+L`	เพิ่ม cursor ข้างบน/ล่าง	`Ctrl+Alt+↑/↓`
ขยาย/หด selection	`Shift+Alt+→/←`	เลือกทั้งหมด	`Ctrl+A`

🔍 Navigation (ไปไหนมาไหน)

Go to File ⭐	`Ctrl+P`	Go to Line	`Ctrl+G`

Go to Symbol	`Ctrl+Shift+O`
Go to Definition ⭐	`F12`
Peek Definition	`Alt+F12`
Find References	`Shift+F12`
Go Back	`Alt+←`
Go Forward	`Alt+→`

🔎 Search & Replace

Find ในไฟล์	`Ctrl+F`
Replace ในไฟล์	`Ctrl+H`
Find ในทุกไฟล์ ⭐	`Ctrl+Shift+F`
Replace ในทุกไฟล์	`Ctrl+Shift+H`

🪟 Window / View

Command Palette ⭐⭐⭐	`Ctrl+Shift+P`
ซ่อน/แสดง Side Bar	`Ctrl+B`
เปิด Terminal	`Ctrl+``
Split Editor	`Ctrl+\`
สลับ Editor Group	`Ctrl+1/2/3`
Zoom In / Out	`Ctrl+= / -`
Zen Mode	`Ctrl+K Z`
Full Screen	`F11`

▶️ Run & Debug

Start Debugging	`F5`
Run Without Debug	`Ctrl+F5`
Stop	`Shift+F5`
Step Over	`F10`
Step Into	`F11`
Step Out	`Shift+F11`
Toggle Breakpoint	`F9`
Continue	`F5`

🎯 ชุด Shortcut 5 ตัวที่ต้องจำให้ได้ (ทำ 80% ของงาน)

1. `Ctrl+Shift+P` — Command Palette (ทำได้ทุกอย่าง)
2. `Ctrl+P` — Go to File (เปิดไฟล์เร็ว)
3. `Ctrl+`` — Terminal (backtick เหนือ Tab)
4. `Ctrl+D` — Multi-select คำเดียวกัน
5. `F12` — Go to Definition (กระโดดไปดู function)

5D.3 Custom Shortcut

สามารถปรับแต่ง shortcut ได้ทุกอัน เปิด Keyboard Shortcuts editor ผ่าน File > Preferences > Keyboard Shortcuts หรือใช้คำสั่ง Preferences: Open Keyboard Shortcuts (Ctrl+K Ctrl+S)

```
// keybindings.json - ตัวอย่าง custom shortcut
[
  {
    "key": "ctrl+alt+t",
    "command": "workbench.action.terminal.new"
  },
  {
    "key": "ctrl+shift+r",
    "command": "workbench.action.reloadWindow"
  }
]
```

5E Extensions ที่ต้องมี

15 extensions ที่นักพัฒนามืออาชีพใช้กันทั่วโลก

5E.1 Extensions คืออะไร และติดตั้งยังไง

Extensions คือ plug-ins ที่เพิ่มความสามารถให้ VSCode — เช่น รองรับภาษาใหม่, เพิ่ม theme, connect ไปเครื่องมืออื่น

> 🛒 **Marketplace ตัวเลขปี 2026**
>
> Visual Studio Marketplace มี extensions **มากกว่า 50,000 ตัว** ทั้งจาก Microsoft และ community — ส่วนใหญ่ฟรี

วิธีติดตั้ง Extension

1. **เปิด Extensions view**

   กด `Ctrl+Shift+X` หรือคลิกไอคอน ⊞ ที่ Activity Bar

2. **ค้นหาชื่อ extension**

   พิมพ์ในช่อง Search เช่น "Python", "GitLens"

3. **ตรวจ publisher ให้ถูก**

   ระวัง extensions ปลอม! ดู publisher ให้ตรง เช่น **Python** publisher ต้องเป็น *Microsoft*

4. **กด Install**

   รอ 5-30 วินาที บาง extension จะขอให้ reload VSCode

5E.2 Top 15 Extensions ที่ต้องมี

🐍 กลุ่ม Python (ถ้าจะเขียน ThaiLLM ต้องมี)

1. Python (by Microsoft)

Publisher: Microsoft · **Downloads:** 160M+

Extension หลักสำหรับ Python ใน VSCode — ให้ IntelliSense, auto-completion, linting, debugging, unit testing, และความสามารถในการสลับระหว่าง Python environments รวมถึง virtual และ conda environments

ติดตั้งแล้วจะได้: Python extension + Pylance (language server) + Python Debugger อัตโนมัติ

2. Pylance (by Microsoft)

Publisher: Microsoft · ติดตั้งอัตโนมัติพร้อม Python extension

Language server ของ Python — เร็วกว่าเดิม 5-10 เท่า ทำ type checking, auto-import, go to definition ได้เหนือกว่า

3. Python Debugger (by Microsoft)

Publisher: Microsoft · ติดตั้งอัตโนมัติพร้อม Python extension

Debug Python ได้เต็มรูปแบบ — set breakpoint, step through, inspect variable, watch expression

4. Jupyter (by Microsoft)

รัน Jupyter Notebook (.ipynb) ได้โดยตรงใน VSCode ไม่ต้องเปิด browser — จำเป็นสำหรับ Data Science / ML

🤖 กลุ่ม AI Coding (สำหรับ ThaiLLM)

5. Continue (by Continue)

Publisher: Continue · **ฟรี · Open Source**

AI coding assistant ที่รองรับ OpenAI-compatible API — เชื่อม ThaiLLM ได้ เขียนโค้ดคุยกับ AI ได้ทั้งใน chat และ inline

⭐ **ตัวที่เราใช้ในคู่มือนี้ (อ่านรายละเอียดในบทต่อไป)**

6. Cline (ทางเลือก Continue)

AI agent ที่ทำได้มากกว่า auto-complete — สามารถสร้างไฟล์ใหม่, แก้หลายไฟล์, รันคำสั่ง terminal ได้เอง รองรับ ThaiLLM ผ่าน OpenAI-compatible config

⑂ กลุ่ม Git (Source Control)

7. GitLens — Git supercharged

Publisher: GitKraken · **Downloads:** 30M+

เสริม Git features ของ VSCode — ดู git blame แต่ละบรรทัด, history, compare branches สวยงาม

8. GitHub Pull Requests and Issues

จัดการ PR และ Issues ของ GitHub ได้ใน VSCode โดยตรง ไม่ต้องสลับไป browser

🎨 กลุ่ม UI & Productivity

9. Material Icon Theme

เปลี่ยน icon ของไฟล์/โฟลเดอร์ให้สวยและแยกประเภทได้ง่าย เช่น .py จะเป็นไอคอน Python, .json เป็นไอคอน JSON

10. Prettier — Code formatter

จัด indent และ format โค้ดให้สวยงามอัตโนมัติเมื่อ save รองรับ JS, TS, HTML, CSS, JSON, YAML, Markdown

11. Error Lens

แสดง error และ warning ในบรรทัดเดียวกับโค้ดเลย (inline) ไม่ต้องเลื่อนเมาส์ไปดู tooltip

12. indent-rainbow

ระบายสี indentation ต่างกันในแต่ละระดับ — ช่วยเห็น structure ของ Python (ที่ใช้ indent แทน brackets)

🌐 กลุ่ม Web Dev & Misc

13. Live Server

Live Preview launches a local development server with live reload for static and dynamic pages — รัน server เปิด HTML ดูผลเรียลไทม์

14. Rainbow CSV

เปิดไฟล์ .csv แสดงแต่ละ column เป็นสีต่างกัน ทำให้อ่านง่ายขึ้นมาก

15. DotENV

Syntax highlighting สำหรับไฟล์ `.env` (ใช้เก็บ THAILLM_API_KEY ในคู่มือนี้)

5E.3 ติดตั้ง Extensions ผ่าน Command Line

สะดวกสำหรับการ setup เครื่องใหม่หลายเครื่อง หรือเขียนเป็น script:

```
# ติดตั้งทีละตัว
code --install-extension ms-python.python
code --install-extension ms-python.vscode-pylance
code --install-extension continue.continue

# ติดตั้งชุดเดียว (เหมาะทำ setup script)
code --install-extension ms-python.python \
     --install-extension ms-python.vscode-pylance \
     --install-extension continue.continue \
     --install-extension eamodio.gitlens \
     --install-extension pkief.material-icon-theme \
     --install-extension esbenp.prettier-vscode \
     --install-extension usernamehw.errorlens \
     --install-extension mikestead.dotenv

# ดู extensions ที่ติดตั้งอยู่
code --list-extensions

# ถอน extension
code --uninstall-extension ms-python.python
```

5F Python Setup ใน VSCode

สำคัญมาก — เตรียม *VSCode* ให้พร้อมสำหรับ *ThaiLLM Python code*

5F.1 ติดตั้ง Python (ระบบ OS)

ก่อนใช้ Python ใน VSCode ต้องมี Python ในเครื่องก่อน VSCode เป็นแค่ editor ไม่ใช่ interpreter

Windows

1. ไปที่ python.org/downloads
2. ดาวน์โหลด Python 3.11+ (แนะนำ 3.12) — **อย่าใช้ 3.13 ถ้าไม่จำเป็น** เพราะ packages บางตัวยังไม่รองรับ
3. Run installer — **ติ๊ก "Add Python to PATH" ก่อน Install!** (สำคัญมาก ไม่งั้น VSCode หาไม่เจอ)
4. เลือก Install Now

macOS

```
# วิธี 1: Homebrew (แนะนำ)
brew install python@3.12

# วิธี 2: ดาวน์โหลด .pkg จาก python.org
```

Linux (Ubuntu/Debian)

```
sudo apt update
sudo apt install python3 python3-pip python3-venv
```

ทดสอบว่า Python ติดตั้งสำเร็จ

```
# Windows
python --version
pip --version

# macOS / Linux
python3 --version
pip3 --version

# ผลลัพธ์ที่ต้องได้
Python 3.12.0
pip 23.3.0
```

5F.2 Virtual Environment — ทำไมต้องใช้?

💡 **เหตุผลที่ต้องใช้ venv เสมอ**

ปัญหา: โปรเจกต์ A ต้องการ openai เวอร์ชัน 1.0 โปรเจกต์ B ต้องการ openai เวอร์ชัน 1.50 — ถ้าไม่มี venv จะติดตั้งได้แค่เวอร์ชันเดียวทั่วทั้งเครื่อง

วิธีแก้: Virtual Environment สร้างโลก Python แยกสำหรับแต่ละโปรเจกต์ — แต่ละโลกมี packages ของตัวเอง ไม่ชนกัน

สร้าง venv

```
# เข้าไปในโฟลเดอร์โปรเจกต์
cd my-thaillm-project

# สร้าง venv ชื่อ '.venv' (convention)
# Windows
python -m venv .venv

# macOS / Linux
python3 -m venv .venv
```

Activate venv

```
# Windows (PowerShell)
.venv\Scripts\Activate.ps1

# Windows (CMD)
.venv\Scripts\activate.bat

# macOS / Linux
source .venv/bin/activate

# ถ้าสำเร็จ จะมี (.venv) โผล่หน้า prompt
(.venv) $ _
```

ติดตั้ง packages ใน venv

```
(.venv) pip install openai python-dotenv
(.venv) pip freeze > requirements.txt

# ดู packages ทั้งหมด
(.venv) pip list
```

ออกจาก venv

```
deactivate
```

5F.3 VSCode ให้เลือก Python Interpreter

> **🎯 ขั้นตอนที่สำคัญที่สุด**
>
> หลังสร้าง venv แล้ว VSCode ต้องรู้ว่าจะใช้ Python ตัวไหน — กด Ctrl+Shift+P (Cmd+Shift+P บน Mac) เปิด Command Palette แล้วพิมพ์ Python: Select Interpreter

1. **เปิดโปรเจกต์ใน VSCode**

   File > Open Folder > เลือกโฟลเดอร์ที่มี .venv/

2. **Command Palette**

   `Ctrl+Shift+P` พิมพ์ `Python: Select Interpreter`

3. **เลือก venv ของโปรเจกต์**

   รายการจะแสดง Python interpreters ที่เจอทั้งหมด เลือกที่มี `('.venv': venv)` ต่อท้าย — VSCode จะจำไว้ใช้ทุกครั้งที่เปิดโปรเจกต์นี้

4. **ตรวจสอบที่ Status Bar**

   มุมล่างขวาของ VSCode จะแสดง `3.12.0 ('.venv': venv)` คลิกตรงนี้ก็เปลี่ยนได้

5F.4 Terminal ใน VSCode + venv

เมื่อเลือก interpreter เป็น venv แล้ว — เวลาเปิด Terminal ใหม่ใน VSCode (`Ctrl+``) จะ **activate venv อัตโนมัติ** ไม่ต้องพิมพ์ `source .venv/bin/activate` ทุกครั้ง

> **✅ เช็คว่า venv activate ถูกต้อง**

```
$ which python
/path/to/project/.venv/bin/python  # ← ชี้มาที่ venv ถูกแล้ว

/usr/bin/python3  # ← ถ้าเห็นนี่แปลว่า venv ไม่ทำงาน
```

5F.5 สร้างและรันไฟล์ Python แรก

1 **สร้างไฟล์ใหม่**

ใน Explorer คลิกขวา > New File ตั้งชื่อ `hello.py`

2 **เขียนโค้ด**

```
print("สวัสดี ThaiLLM!")
print("ทดสอบ Python ใน VSCode")
```

3 **Save**

`Ctrl+S`

4 **Run**

วิธีที่ 1: กดปุ่ม ▶ มุมบนขวาของ editor
วิธีที่ 2: `Ctrl+F5` (Run Without Debugging)
วิธีที่ 3: Terminal พิมพ์ `python hello.py`

5F.6 IntelliSense & Autocomplete

Pylance จะทำงานอัตโนมัติ — เวลาพิมพ์จะขึ้น suggestion

ฟีเจอร์	Shortcut	ทำอะไร
Trigger Suggestion	`Ctrl+Space`	เรียก autocomplete ขึ้นมา
Hover info	เมาส์ค้างบนคำ	ดู docstring, type
Parameter hints	`Ctrl+Shift+Space`	ดู arguments ของ function
Quick Fix	`Ctrl+.`	Auto-import, rename, refactor

5G Debugger + Git

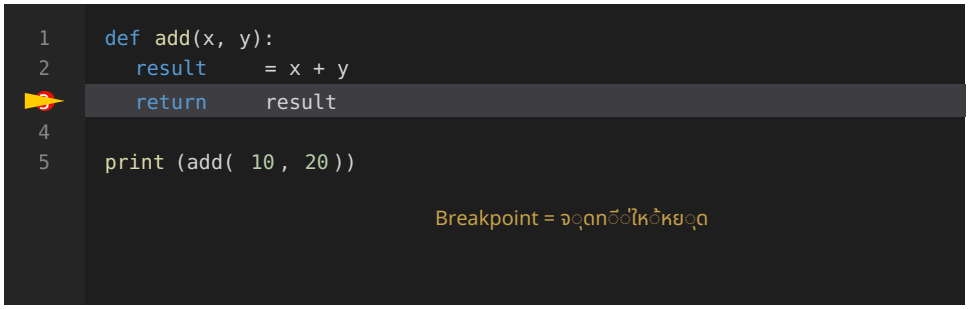
สองเครื่องมือที่ทำให้คุณเป็นมืออาชีพ

5G.1 Debugger — เก่งขึ้น 10 เท่าใน 10 นาที

Debugger คือเครื่องมือให้เราหยุดโปรแกรมตรงกลาง ดูค่าตัวแปร ทีละบรรทัด แทนที่จะใช้ `print()` ดูทุกจุด

Breakpoint

Breakpoint = จุดให้โปรแกรมหยุดรอเรา กด F9 หรือคลิกที่ gutter ซ้ายของหมายเลขบรรทัด เพื่อ set breakpoint



รูปที่ *5G.1 — Breakpoint* และ *Current line indicator* ในขณะ *debug*

เริ่ม Debug (วิธีง่ายที่สุด)

เปิดไฟล์ Python ที่อยากจะ debug

กด `F9` ที่บรรทัดที่อยากหยุด (จะเห็นจุดแดง)

กด `F5` เพื่อเริ่ม debug

ถ้าถามให้เลือก config เลือก **Python Debugger: Python File**

โปรแกรมจะรันจนถึง breakpoint แล้วหยุด

ปุ่มควบคุม Debug

ปุ่ม	Shortcut	ทำอะไร
▶ Continue	`F5`	วิ่งต่อจนถึง breakpoint ถัดไป
↷ Step Over	`F10`	ข้ามทีละบรรทัด (ถ้าเจอ function ไม่เข้าไปข้างใน)
↲ Step Into	`F11`	เข้าไปดูข้างใน function
↑ Step Out	`Shift+F11`	ออกจาก function ปัจจุบัน
↻ Restart	`Ctrl+Shift+F5`	เริ่มใหม่
■ Stop	`Shift+F5`	หยุดทั้งหมด

Debug View — 4 panel สำคัญ

- **VARIABLES** — ดูค่าตัวแปรทุกตัวในขณะนั้น ขยายเพื่อดู attributes
- **WATCH** — พิมพ์ expression ที่อยากดูเรื่อย ๆ (เช่น `len(myList)`)
- **CALL STACK** — ลำดับ function calls ที่เรียกมาจนถึงตรงนี้
- **BREAKPOINTS** — รายการ breakpoint ทั้งหมด enable/disable ได้

launch.json — ตั้งค่า Debug

Configuration ที่ขับเคลื่อน VSCode ขณะ debug session ถูกเก็บในไฟล์ launch.json ใน .vscode folder ของ workspace

```
// .vscode/launch.json
{
    "version": "0.2.0",
    "configurations": [
        {
            "name": "Python: ThaiLLM App",
            "type": "debugpy",
            "request": "launch",
            "program": "${file}",
            "console": "integratedTerminal",
            "env": {
                "THAILLM_API_KEY": "${env:THAILLM_API_KEY}"
            },
            "justMyCode": true
        },
        {
            "name": "Python: Streamlit",
            "type": "debugpy",
            "request": "launch",
            "module": "streamlit",
            "args": ["run", "app.py"]
        }
    ]
}
```

Logpoints — Debug ดีกว่า print()

> ✅ **Pro Tip: Logpoint**
>
> Logpoint เป็นเหมือน breakpoint แต่บันทึก message ไปที่ console โดยไม่หยุดโปรแกรม — ใช้แทน `print()` ไปทั่วโค้ด ที่สำคัญคือไม่ต้องแก้ source code! คลิกขวา gutter > Add Logpoint > ใส่ข้อความ เช่น `value of x is {x}`

5G.2 Git ใน VSCode — Source Control

VSCode มี Git support built-in — เลือก Source Control view ใน Activity Bar ทำ commit, stage changes, ดู history ทั้งหมดได้จากใน VSCode

Workflow พื้นฐาน

1 **Initialize Repository**

เปิด Source Control (`Ctrl+Shift+G`) กดปุ่ม **Initialize Repository** — สร้าง .git/ ในโฟลเดอร์

2 **Stage Changes**

แก้ไฟล์ → แสดงใน "Changes" hover mouse แล้วกด ➕ เพื่อ stage (หรือกด Stage All Changes)

3 **Commit**

พิมพ์ commit message ในช่องด้านบน กด `Ctrl+Enter` หรือปุ่ม Commit

4 **Push**

กดจุด 3 จุด (…) > Push — ต้อง add remote แล้วก่อน เช่น `git remote add origin https://github.com/you/repo.git`

ดูการเปลี่ยนแปลง (Diff View)

คลิกไฟล์ใน Source Control จะเห็น **Diff View**:

- 🟥 **สีแดง** — โค้ดที่ลบออก (ซ้าย)
- 🟩 **สีเขียว** — โค้ดที่เพิ่มเข้า (ขวา)
- 🟨 **สีเหลือง** — โค้ดที่แก้ไข

.gitignore — สำคัญมากสำหรับ ThaiLLM

```
# .gitignore สำหรับ ThaiLLM project

# API keys และ secrets
.env
.env.*
*.key

# Python
__pycache__/
*.pyc
.venv/
venv/

# VSCode (บางคนเก็บ บางคนไม่เก็บ)
# .vscode/

# OS
```

```
.DS_Store
Thumbs.db

# IDE
.idea/
*.swp

# Build
dist/
build/
*.egg-info/
```

5G.3 Settings Sync & Profiles

Settings Sync — ใช้ VSCode เครื่องใหม่เหมือนเครื่องเก่า

ซิงค์ settings, extensions, keyboard shortcuts ข้ามเครื่องผ่าน GitHub หรือ Microsoft account

1. คลิกปุ่ม Accounts (คนในมุมล่างซ้าย) > **Turn on Settings Sync**
2. เลือกสิ่งที่อยาก sync (Settings, Keybindings, Extensions, UI State, Snippets)
3. Sign in ด้วย GitHub หรือ Microsoft

พอไปเครื่องใหม่ ติดตั้ง VSCode → Sign in → ทุกอย่างกลับมาเหมือนเดิม

Profiles — หลาย environment ในเครื่องเดียว

สำหรับคนที่ทำงานหลายประเภท:

- **Profile "Python"** — เฉพาะ Python extensions, theme เข้ม, font ขนาด 12
- **Profile "Web Dev"** — HTML/CSS/JS extensions, Live Server, Prettier
- **Profile "ThaiLLM Tutorial"** — Continue, Python, minimal theme สำหรับอัดคลิป

สร้าง: `Ctrl+Shift+P` > `Profiles: Create Profile`

5H ใช้ VSCode + Continue + ThaiLLM

สรุปการ *integrate* ทุกอย่างเข้าด้วยกัน — ทำงานเป็นทีม *AI* ของคุณ

5H.1 ภาพรวมสถาปัตยกรรม

รูปที่ *5H.1* — สถาปัตยกรรมการทำงานร่วมกันของ *VSCode + Continue + ThaiLLM*

5H.2 Workflow ประจำวันของนักพัฒนา

หลังจากติดตั้งครบแล้ว workflow มักเป็นแบบนี้:

1. **เปิด VSCode → เปิดโปรเจกต์**

   `File > Open Recent` หรือ `code .` จาก Terminal

2. **venv activate อัตโนมัติ**

   VSCode จำ interpreter ไว้ Terminal ใหม่จะ activate venv ให้ทันที

3. **เขียนโค้ด (หรือให้ Continue ช่วย)**

   พิมพ์โค้ด หรือเปิด Continue chat (`Ctrl+L`) ถาม ThaiLLM เป็นภาษาไทย

4. **Test ใน Terminal**

   `Ctrl+`` เปิด terminal แล้วรัน `python script.py`

5 **Debug หากมีปัญหา**

ตั้ง breakpoint (`F9`) แล้วกด `F5` เพื่อ debug หรือถาม Continue ว่า "ทำไม error นี้เกิด?" โดย copy error message ไปถาม

6 **Commit ผ่าน Source Control**

`Ctrl+Shift+G` พิมพ์ message (ขอ Continue ช่วยแต่งได้) กด Commit

5H.3 ตัวอย่างการใช้ Continue กับ ThaiLLM

สถานการณ์ 1: ให้ AI อธิบายโค้ด

1. เลือกโค้ด (เช่น function ที่ซับซ้อน)
2. กด `Ctrl+L` เปิด Continue chat
3. พิมพ์ "อธิบายโค้ดนี้เป็นภาษาไทย ทีละบรรทัด"
4. AI (Pathumma) จะอธิบาย พร้อมทั้ง logic และจุดที่อาจมี bug

สถานการณ์ 2: ให้ AI แก้โค้ดให้

1. เลือกโค้ด
2. กด `Ctrl+I` (Continue Edit mode)
3. พิมพ์ "เพิ่ม error handling และ retry 3 ครั้งถ้าเจอ rate limit"
4. AI จะเสนอการเปลี่ยนแปลง — กด Accept หรือ Reject

สถานการณ์ 3: สร้างไฟล์ใหม่จาก spec

1. เปิด Continue chat
2. พิมพ์ "สร้างไฟล์ Streamlit app ที่ใช้ ThaiLLM แปลภาษา ไทย→อังกฤษ พร้อม error handling"
3. AI generate โค้ด — กด Copy แล้วสร้างไฟล์ใหม่

5H.4 Checklist สุดท้าย — พร้อมเริ่ม!

✅ ถ้าครบทั้งหมดนี้ = พร้อมใช้งานระดับมืออาชีพ

- ☑️ VSCode ติดตั้งครบ + ใน PATH
- ☑️ Python 3.12 ติดตั้ง + venv
- ☑️ Python extension + Pylance
- ☑️ Continue extension
- ☑️ GitLens + Git configured
- ☑️ Material Icon + Prettier + Error Lens
- ☑️ THAILLM_API_KEY ใน .env
- ☑️ .env อยู่ใน .gitignore
- ☑️ Continue config.yaml ตั้งค่า
- ☑️ ลอง chat กับ Pathumma สำเร็จ
- ☑️ Settings Sync เปิด (เผื่อเปลี่ยนเครื่อง)
- ☑️ รู้จัก Command Palette

📺 ดูคลิปประกอบเพิ่มเติม

สำหรับการ setup ทีละขั้นแบบมีคลิปดู ดูวิดีโอประกอบของคู่มือนี้ที่ **youtu.be/MCBwEexM5C4** บนช่อง iCafeFX

บทถัดไปจะเข้าสู่การใช้ *Continue* กับ *ThaiLLM* อย่างละเอียด *(บท 6 เดิม)*

5 ติดตั้ง VSCode + Continue

ให้ *VSCode* ของคุณกลายเป็น *IDE* ที่มี *ThaiLLM* ช่วยเขียนโค้ด

5.1 ทำไมต้องใช้ Continue?

Continue เป็น extension open-source สำหรับ VSCode ที่ทำให้คุณใช้ LLM ใด ๆ (OpenAI, Anthropic, Ollama, ThaiLLM, ฯลฯ) ช่วยเขียนโค้ดได้โดยตรงในโปรแกรม โดยมีความสามารถหลัก 4 ด้าน:

ความสามารถ	คำอธิบาย	Keyboard Shortcut
Chat	คุยกับ AI ใน panel ด้านข้าง ถาม-ตอบ debug	Ctrl+L (Cmd+L)
Edit	เลือก code → สั่งให้แก้ไข	Ctrl+I (Cmd+I)
Autocomplete	AI เติมโค้ดให้อัตโนมัติขณะพิมพ์	Tab (แทน ghost text)
Apply	นำโค้ดจาก chat ไปใส่ในไฟล์จริง	ปุ่มในแชท

✅ **ทางเลือกอื่น**

Cline — คล้าย Continue แต่เน้น agent mode (AI แก้โค้ดหลายไฟล์ได้เอง) ทั้งสองตัวรองรับ OpenAI-compatible API เหมือนกัน ดังนั้น config ThaiLLM ได้เหมือนกัน

Cursor IDE — IDE แยกที่ fork มาจาก VSCode มี AI ในตัว แต่เสียเงิน ใช้ ThaiLLM ผ่านการตั้งค่า custom model ได้

Aider — CLI tool (ไม่ใช่ extension) ใช้ผ่าน terminal รองรับ OpenAI-compatible API

5.2 ขั้นตอนการติดตั้ง

1 **ติดตั้ง VSCode**

หากยังไม่มี ดาวน์โหลดที่ code.visualstudio.com ติดตั้งตามปกติ รองรับทั้ง Windows, macOS, และ Linux

2 **เปิด Extensions panel**

กด `Ctrl+Shift+X` (Windows/Linux) หรือ `Cmd+Shift+X` (macOS) จะปรากฏ panel ด้านซ้าย

3 **ค้นหา "Continue"**

พิมพ์ **Continue** ในช่องค้นหา เลือกตัวที่ publisher เป็น `Continue` (มีรูปลูกศรสีเขียว-น้ำเงิน) กดปุ่ม **Install**

4 **รอการติดตั้ง**

Continue จะดาวน์โหลด bundle ขนาดประมาณ 80MB ใช้เวลาประมาณ 30 วินาที – 1 นาที ขึ้นกับความเร็วเน็ต

5 **เปิด Continue panel**

กด `Ctrl+L` หรือคลิก icon Continue ในแถบด้านซ้าย จะปรากฏ panel ให้ chat ทางขวา

6 **ยืนยันการติดตั้ง**

หากเห็นข้อความให้เลือก model provider แสดงว่าติดตั้งสำเร็จ แต่อย่าเพิ่งเลือก — ขั้นตอนต่อไปเราจะตั้งค่าผ่าน `config.yaml` โดยตรง

5.3 โครงสร้างไฟล์ Continue

Continue เก็บ config ไว้ในโฟลเดอร์ global ของ user:

OS	ตำแหน่ง
Windows	`%USERPROFILE%\.continue\` (เช่น C:\Users\yourname\.continue\)
macOS / Linux	`~/.continue/`

ในโฟลเดอร์นี้จะมีไฟล์ `config.yaml` (หรือ `config.json` ในเวอร์ชันเก่า) ซึ่งเป็นที่เรากำหนดว่าจะใช้โมเดลไหนบ้าง บทต่อไปเราจะแก้ไฟล์นี้

6 ตั้งค่า config.yaml

เปลี่ยน *Continue* ให้เรียกใช้ *ThaiLLM* ทุกตัวใน *5* นาที

6.1 เปิดไฟล์ config.yaml

มี 2 วิธี:

วิธีที่ 1: ใน Continue panel กดไอคอน gear (⚙️) มุมบนขวา เลือก `Open config.yaml`

วิธีที่ 2: เปิดไฟล์ตรง ๆ

```
Windows:  C:\Users\YOURNAME\.continue\config.yaml
macOS:    ~/.continue/config.yaml
Linux:    ~/.continue/config.yaml
```

หากไม่มีไฟล์ ให้สร้างใหม่ได้เลย

6.2 Config พื้นฐาน (ตัวอย่างแรก)

คัดลอกโค้ดต่อไปนี้ลงใน `config.yaml` แล้วแทน `YOUR_API_KEY_HERE` ด้วย key จริงจากบทที่ 4:

```yaml
# ~/.continue/config.yaml
name: ThaiLLM Assistant
version: 1.0.0
schema: v1

models:
  # === โมเดลหลัก: Pathumma (มี thinking mode) ===
  - name: Pathumma Think
    provider: openai
    model: Pathumma-ThaiLLM-qwen3-8b-think-3.0.0
    apiBase: https://api.thaillm.or.th/v1
    apiKey: YOUR_API_KEY_HERE
    roles:
      - chat
      - edit
      - apply
    capabilities:
      - tool_use
    defaultCompletionOptions:
      temperature: 0.7
      topP: 0.8
      maxTokens: 4096
```

> 💡 **ประเด็นสำคัญ**
>
> **provider: openai** — ต้องใช้คำนี้ เพราะ ThaiLLM compatible กับ OpenAI format ไม่ใช่เพราะใช้ OpenAI จริง ๆ
>
> **apiBase** — ลงท้ายด้วย `/v1` อย่าลืม! เพราะ Continue จะเติม `/chat/completions` ต่อท้ายให้อัตโนมัติ
>
> **roles** — ถ้าไม่ใส่ Continue จะถือว่าโมเดลนี้ใช้ได้ทุก role

6.3 Config เต็มรูปแบบ (หลายโมเดล)

เพื่อให้คุณ **สลับโมเดลได้ตามงาน** แนะนำให้กำหนดโมเดลทั้ง 4 ตัวไว้ แล้วเลือกใช้จาก dropdown ใน Continue:

```yaml
name: ThaiLLM Assistant
version: 1.0.0
schema: v1

models:
  # 1. Pathumma - สำหรับ reasoning / คิดซับซ้อน
  - name: 🌸 Pathumma Think
    provider: openai
    model: Pathumma-ThaiLLM-qwen3-8b-think-3.0.0
    apiBase: https://api.thaillm.or.th/v1
    apiKey: YOUR_API_KEY_HERE
    roles: [chat, edit]
    defaultCompletionOptions:
      temperature: 0.6
      topP: 0.95
      maxTokens: 8192

  # 2. OpenThaiGPT - ใช้งานทั่วไป ตอบเร็ว
  - name: 🇹🇭 OpenThaiGPT
    provider: openai
    model: OpenThaiGPT-ThaiLLM-8B-Instruct-v7.2
    apiBase: https://api.thaillm.or.th/v1
    apiKey: YOUR_API_KEY_HERE
    roles: [chat, apply]
    defaultCompletionOptions:
      temperature: 0.7
      topP: 0.8
      maxTokens: 4096

  # 3. Typhoon-S - Agent / Tool Calling
  - name: 🌀 Typhoon-S
    provider: openai
    model: Typhoon-S-ThaiLLM-8B-Instruct
    apiBase: https://api.thaillm.or.th/v1
    apiKey: YOUR_API_KEY_HERE
    roles: [chat, edit]
    capabilities:
      - tool_use
    defaultCompletionOptions:
      temperature: 0.7
```

```
      topP: 0.8
      maxTokens: 4096

  # 4. THaLLE - งานการเงินโดยเฉพาะ
  - name: 🌊 THaLLE Finance
    provider: openai
    model: THaLLE-0.2-ThaiLLM-8B-fa
    apiBase: https://api.thaillm.or.th/v1
    apiKey: YOUR_API_KEY_HERE
    roles: [chat]
    defaultCompletionOptions:
      temperature: 0.3
      topP: 0.9
      maxTokens: 4096

context:
  - provider: code
  - provider: diff
  - provider: terminal
  - provider: file
  - provider: folder
  - provider: codebase

rules:
  - ตอบเป็นภาษาไทยเมื่อถูกถามเป็นภาษาไทย
  - ใช้ภาษาอังกฤษเฉพาะกับศัพท์เทคนิค
  - เขียนคำอธิบายโค้ดเป็นภาษาไทย
  - ใช้ TypeScript หากเป็นโค้ด JavaScript

prompts:
  - name: explain-th
    description: อธิบายโค้ดเป็นภาษาไทย
    prompt: |
      กรุณาอธิบายโค้ดต่อไปนี้เป็นภาษาไทย ระบุ:
      1. ทำอะไร
      2. รับ input อะไร
      3. คืน output อะไร
      4. มีจุดที่อาจเกิด bug หรือควรปรับปรุงไหม
  - name: review-th
    description: Review โค้ดเป็นภาษาไทย
    prompt: |
      ช่วย review โค้ดนี้เป็นภาษาไทย ตรวจสอบ:
      - Security issues
      - Performance
      - Code readability
      - Best practices
```

6.4 ตั้งค่า Cline แทน Continue

ถ้าคุณใช้ Cline อยู่แล้ว ไม่ต้องเปลี่ยน ตั้งค่าได้ดังนี้:

1

เปิด Cline Settings

กด icon Cline ใน VSCode → **Settings** (gear icon)

2

เลือก API Provider

ใน dropdown เลือก **OpenAI Compatible**

3

ใส่ข้อมูล

Base URL: `https://api.thaillm.or.th/v1`

API Key: paste key จากบทที่ 4

Model ID: `Pathumma-ThaiLLM-qwen3-8b-think-3.0.0` (หรือโมเดลอื่นที่ต้องการ)

4

Save และทดลอง

กด Save จากนั้นลองสั่ง: "อธิบายโครงสร้างโปรเจกต์นี้เป็นภาษาไทย"

7 Python Demo

5 ตัวอย่างการใช้งาน ThaiLLM กับ Python ใช้งานได้จริง

7.1 ติดตั้ง Dependencies

```
# เปิด terminal แล้วรันคำสั่งนี้
pip install openai python-dotenv

# (เสริม) สำหรับ streaming และ app demo
pip install rich streamlit
```

สร้างไฟล์ `.env` ไว้ในโฟลเดอร์เดียวกับโค้ด:

```
THAILLM_API_KEY=tllm_sk-xxxxxxxxxxxxx
THAILLM_BASE_URL=https://api.thaillm.or.th/v1
```

7.2 ตัวอย่างที่ 1 — Hello World

```
# hello_thaillm.py
from openai import OpenAI
from dotenv import load_dotenv
import os

load_dotenv()

client = OpenAI(
    api_key=os.getenv("THAILLM_API_KEY"),
    base_url=os.getenv("THAILLM_BASE_URL")
)

response = client.chat.completions.create(
    model="Pathumma-ThaiLLM-qwen3-8b-think-3.0.0",
    messages=[
        {"role": "user", "content": "สวัสดีครับ แนะนำตัวเองหน่อย"}
    ],
    temperature=0.7,
    max_tokens=500
)

print(response.choices[0].message.content)
```

7.3 ตัวอย่างที่ 2 — แปลภาษา (หลายภาษา)

```
# translator.py
from openai import OpenAI
from dotenv import load_dotenv
import os

load_dotenv()
client = OpenAI(
    api_key=os.getenv("THAILLM_API_KEY"),
    base_url=os.getenv("THAILLM_BASE_URL")
)

def translate(text, target_lang="English"):
    """แปลข้อความเป็นภาษาที่กำหนด"""
    response = client.chat.completions.create(
        model="Pathumma-ThaiLLM-qwen3-8b-think-3.0.0",
        messages=[
            {"role": "system", "content": "คุณคือนักแปลภาษามืออาชีพ แปลตรงตัว ไม่ต้องอธิบาย"},
            {"role": "user", "content": f"แปลเป็น {target_lang}: {text}"}
        ],
        temperature=0.3,  # ต่ำเพื่อความสม่ำเสมอ
        max_tokens=500
    )
    return response.choices[0].message.content

text = "สวัสดีครับ ยินดีต้อนรับสู่เว็บไซต์ของเรา"

print("EN:", translate(text, "English"))
print("JP:", translate(text, "Japanese"))
print("CN:", translate(text, "Chinese"))
print("KR:", translate(text, "Korean"))
print("DE:", translate(text, "German"))
```

7.4 ตัวอย่างที่ 3 — Streaming (ตอบทีละคำ)

สำหรับประสบการณ์เหมือน ChatGPT ที่คำตอบไหลมาทีละคำ:

```
# streaming.py
from openai import OpenAI
from dotenv import load_dotenv
import os

load_dotenv()
client = OpenAI(
    api_key=os.getenv("THAILLM_API_KEY"),
    base_url=os.getenv("THAILLM_BASE_URL")
)

stream = client.chat.completions.create(
    model="Pathumma-ThaiLLM-qwen3-8b-think-3.0.0",
    messages=[
```

```
        {"role": "user", "content": "เล่าประวัติพระนเรศวรสั้น ๆ 200 คำ"}
    ],
    stream=True,  # เปิด streaming
    temperature=0.7
)

for chunk in stream:
    if chunk.choices[0].delta.content:
        print(chunk.choices[0].delta.content, end="", flush=True)
print()  # บรรทัดใหม่ตอนจบ
```

7.5 ตัวอย่างที่ 4 — สรุปบทความ

```
# summarize.py
from openai import OpenAI
from dotenv import load_dotenv
import os

load_dotenv()
client = OpenAI(
    api_key=os.getenv("THAILLM_API_KEY"),
    base_url=os.getenv("THAILLM_BASE_URL")
)

article = """
ThaiLLM คือโมเดล AI ภาษาไทยขนาดใหญ่ พัฒนาโดย NECTEC, VISTEC, AIEAT,
KBTG, SCB 10X, BDI และพันธมิตร ภายใต้การสนับสนุนของ สวทช.
โมเดลถูกฝึกด้วยข้อมูลไทยกว่า 100 พันล้านโทเคน บนซูเปอร์คอมพิวเตอร์ LANTA
ที่มี 704 NVIDIA A100 GPUs ทำให้ไทยเป็น 1 ใน 4 ประเทศเอเชียที่มี LLM
ของตัวเอง ร่วมกับจีน ญี่ปุ่น และเกาหลีใต้
"""

response = client.chat.completions.create(
    model="OpenThaiGPT-ThaiLLM-8B-Instruct-v7.2",
    messages=[
        {"role": "user",
         "content": f"สรุปบทความต่อไปนี้เป็น 3 bullet points:\n\n{article}"}
    ],
    temperature=0.3
)

print(response.choices[0].message.content)
```

7.6 ตัวอย่างที่ 5 — Streamlit Web App แปลภาษา

สร้างเว็บแอปพลิเคชันแปลภาษาภายใน 30 บรรทัด:

```
# streamlit run translator_app.py
import streamlit as st
from openai import OpenAI
```

```
from dotenv import load_dotenv
import os

load_dotenv()
client = OpenAI(
    api_key=os.getenv("THAILLM_API_KEY"),
    base_url=os.getenv("THAILLM_BASE_URL")
)

st.set_page_config(page_title="ThaiLLM Translator", page_icon="🇹🇭")
st.title("🇹🇭 ThaiLLM Translator")
st.caption("แปลภาษาด้วย AI สัญชาติไทย - ฟรี ข้อมูลไม่ออกนอกประเทศ")

col1, col2 = st.columns(2)
with col1:
    source = st.text_area("ข้อความต้นฉบับ", height=200, placeholder="พิมพ์ข้อความ...")
with col2:
    target = st.selectbox("แปลเป็น",
        ["English", "Japanese", "Chinese", "Korean", "German", "Thai"])

if st.button("แปล", type="primary", use_container_width=True):
    if source.strip():
        with st.spinner("กำลังแปล..."):
            response = client.chat.completions.create(
                model="Pathumma-ThaiLLM-qwen3-8b-think-3.0.0",
                messages=[
                    {"role": "system", "content": "นักแปลมืออาชีพ แปลตรงตัว"},
                    {"role": "user", "content": f"แปลเป็น {target}: {source}"}
                ],
                temperature=0.3
            )
            st.success(response.choices[0].message.content)
    else:
        st.warning("กรุณาใส่ข้อความก่อน")
```

รันด้วย: `streamlit run translator_app.py` — จะเปิดเว็บที่ `localhost:8501`

8 เทคนิคขั้นสูง ใหม่ 2026

ข้อมูลเชิงลึกที่ไม่ค่อยมีในคู่มือทั่วไป — ใช้ *ThaiLLM* ให้เต็มประสิทธิภาพ

8.1 Thinking Mode Control — การเปิดปิดโหมดคิด

Pathumma และ Typhoon-S สร้างบน **Qwen3** ซึ่งมีคุณสมบัติพิเศษคือ **Thinking Mode** — โมเดลจะ "คิด" ภายใน block `<think>...</think>` ก่อนตอบจริง ทำให้คำตอบแม่นยำขึ้นในงานที่ต้องใช้เหตุผล

ปัญหา: Thinking Mode ช้ากว่าและใช้โทเคนเยอะกว่า

สำหรับงานง่าย ๆ เช่น "สวัสดี" หรือ "แปลคำนี้" โมเดลไม่จำเป็นต้องคิด การเปิด thinking mode เต็มที่ทุกครั้งจะทำให้:

- **ช้าลง 2-5 เท่า** เพราะต้องสร้าง thinking tokens ก่อน
- **ใช้ context window มากขึ้น** — ถ้าคำตอบยาว อาจเกิน max_tokens
- **เปลืองทรัพยากร API** โดยไม่จำเป็น

วิธีที่ 1: ใช้ /think และ /no_think ใน prompt

```
# ให้คิดก่อนตอบ (เหมาะกับโจทย์ซับซ้อน)
messages = [{
    "role": "user",
    "content": "ร้าน A ลด 20% จากนั้นลดอีก 15% รวมลดกี่%? /think"
}]

# ตอบเร็วโดยไม่คิด (เหมาะกับ chatbot)
messages = [{
    "role": "user",
    "content": "สวัสดีครับ /no_think"
}]
```

วิธีที่ 2: System prompt บังคับโหมด

```
messages = [
    {"role": "system", "content": "ตอบสั้น กระชับ /no_think"},
    {"role": "user", "content": "ประเทศไทยอยู่ในทวีปไหน?"}
]
```

วิธีที่ 3: กรอง thinking block ออกจากผลลัพธ์

บางครั้งเราเปิด thinking mode เพื่อคุณภาพ แต่ไม่ต้องการให้ user เห็น:

```
import re

def strip_thinking(text):
    """ลบ <think>...</think> ออกจากคำตอบ"""
```

```
    return re.sub(r'<think>.*?</think>', '', text, flags=re.DOTALL).strip()

response = client.chat.completions.create(...)
clean_answer = strip_thinking(response.choices[0].message.content)
print(clean_answer)
```

8.2 Temperature & Sampling พารามิเตอร์

Qwen3/ThaiLLM มี sampling parameters หลายตัว ที่นักพัฒนาทีมพัฒนาแนะนำค่าต่อไปนี้:

Parameter	ค่าแนะนำ (Instruct)	ค่าแนะนำ (Thinking)	ใช้ทำอะไร
`temperature`	0.7	0.6	ความ creative / random (0=แน่นอน, 2=มั่ว)
`top_p`	0.8	0.95	nucleus sampling (จำกัดเฉพาะ top candidates)
`top_k`	20	20	จำกัดจำนวน token ที่เลือก
`min_p`	0	0	probability ขั้นต่ำที่ยอมรับ
`presence_penalty`	0 - 2	0 - 1.5	ลดการซ้ำคำ (ค่าสูง = ซ้ำน้อย)
`max_tokens`	4096	8192-16384	ความยาวคำตอบสูงสุด

✅ **คู่มือตั้งค่าตามประเภทงาน**

งานสร้างสรรค์ (เขียนนิยาย, แต่งเพลง, brainstorm) → `temperature=1.0-1.3, top_p=0.95`

งานแปล / สรุป → `temperature=0.3, top_p=0.8` (ให้สม่ำเสมอ)

งานวิเคราะห์ / ถามตอบ → `temperature=0.7, top_p=0.8` (balanced)

สกัด JSON / Structured output → `temperature=0, top_p=1.0` (deterministic)

หลีกเลี่ยง greedy decoding (temperature=0) กับโหมด thinking — มักจะเกิด loop / repeat

8.3 Long Context — ใช้ YaRN ขยายได้ถึง 128K

โดย default ThaiLLM รองรับ **32,768 tokens** (ประมาณ 25,000 คำไทย / 50 หน้า A4) ซึ่งเพียงพอสำหรับงานทั่วไป แต่ถ้าต้องวิเคราะห์เอกสารยาวมาก ๆ เช่น สัญญา 200 หน้า หรือ code base ขนาดใหญ่ จะต้องขยายด้วยเทคนิคชื่อ **YaRN (Yet another RoPE extensioN)**

> ⚠️ **หมายเหตุสำคัญ**
> YaRN ใช้ได้เฉพาะกรณีที่คุณ **host model เอง** (บน vLLM, SGLang, Ollama ฯลฯ) เพราะต้องแก้ไฟล์ `config.json` ของโมเดล ไม่ใช่ผ่าน API cloud โดยตรง

ถ้าใช้ API: ดู context ที่มีอยู่

```python
import tiktoken

# ประมาณจำนวน token ใน text
def count_tokens(text):
    encoding = tiktoken.get_encoding("cl100k_base")
    return len(encoding.encode(text))

long_doc = open("contract.txt").read()
n = count_tokens(long_doc)
print(f"ประมาณ {n} tokens")

if n > 30000:
    print("⚠️ ใกล้เกิน limit — ควรแบ่ง chunk หรือ ใช้ RAG")
```

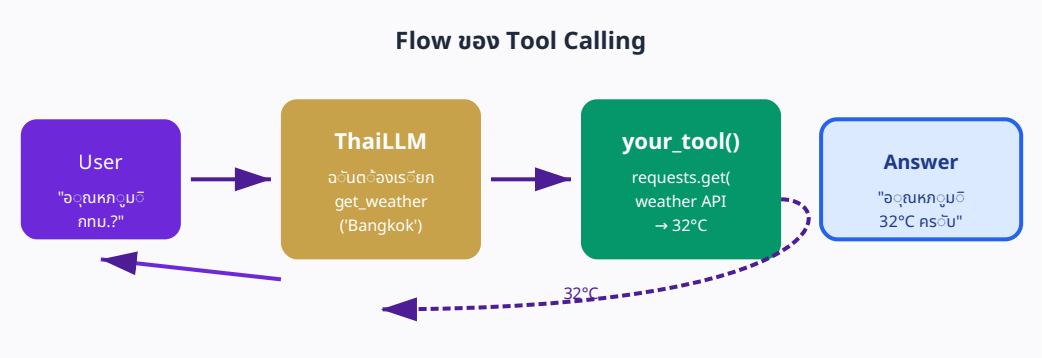
ถ้า self-host: แก้ config.json เพื่อเปิด YaRN

```
// config.json ของ model
{
  ...
  "rope_scaling": {
    "factor": 4.0,
    "original_max_position_embeddings": 32768,
    "type": "yarn"
  }
}
```

ด้วย factor=4 จะได้ context 128K tokens

8.4 Tool Calling / Function Calling

Tool Calling ทำให้ AI สามารถ **เรียก function ที่คุณเขียนไว้** เพื่อดึงข้อมูลจริง เช่น อุณหภูมิปัจจุบัน ราคาหุ้น ข้อมูลจาก database ฯลฯ



รูปที่ *8.1 — Tool Calling Flow: AI เรียก function → รับข้อมูลจริง → ตอบ*

ตัวอย่าง Tool Calling สมบูรณ์

```
from openai import OpenAI
import json

client = OpenAI(api_key="...", base_url="https://api.thaillm.or.th/v1")

# 1. กำหนด function ที่ AI เรียกได้
def get_weather(city: str) -> dict:
    """ดึงอุณหภูมิจาก API จริง (mock ไว้)"""
    return {"city": city, "temp": 32, "unit": "Celsius"}

# 2. ระบุ tool schema ให้ AI รู้
tools = [{
    "type": "function",
    "function": {
        "name": "get_weather",
        "description": "ดึงอุณหภูมิปัจจุบันของเมืองที่ระบุ",
        "parameters": {
            "type": "object",
            "properties": {
                "city": {"type": "string", "description": "ชื่อเมือง"}
            },
            "required": ["city"]
        }
    }
}]

# 3. ถาม AI
messages = [{"role": "user", "content": "อุณหภูมิที่กรุงเทพตอนนี้?"}]

response = client.chat.completions.create(
    model="Typhoon-S-ThaiLLM-8B-Instruct",
    messages=messages,
    tools=tools,
    tool_choice="auto"
)

# 4. AI เลือกเรียก tool - ดึง arguments
msg = response.choices[0].message
if msg.tool_calls:
```

```
    tc = msg.tool_calls[0]
    args = json.loads(tc.function.arguments)
    result = get_weather(**args)

    # 5. ส่งผลกลับให้ AI สร้างคำตอบ
    messages.append(msg)
    messages.append({
        "role": "tool",
        "tool_call_id": tc.id,
        "content": json.dumps(result)
    })

    final = client.chat.completions.create(
        model="Typhoon-S-ThaiLLM-8B-Instruct",
        messages=messages
    )
    print(final.choices[0].message.content)
    # → "อุณหภูมิที่กรุงเทพตอนนี้ 32°C ครับ"
```

8.5 RAG ภาษาไทย — ถามตอบบนเอกสารของคุณ

RAG (Retrieval-Augmented Generation) คือเทคนิคที่ให้ AI "หา" ข้อมูลจากเอกสารของคุณก่อนตอบ ทำให้ AI ตอบได้ **เฉพาะข้อมูลที่คุณให้** ไม่มั่ว (hallucinate)

Flow ของ RAG

1. **Index:** แบ่งเอกสารเป็น chunks แล้ว embed เป็น vector เก็บใน vector DB
2. **Retrieve:** เมื่อ user ถาม → embed คำถาม → หา chunks ที่คล้ายที่สุด
3. **Generate:** ส่ง chunks + คำถามให้ ThaiLLM → ได้คำตอบที่อ้างอิงจากเอกสาร

> 💡 **RAG Stack แนะนำสำหรับคนไทย**
>
> **Embedding:** `BAAI/bge-m3` หรือ `intfloat/multilingual-e5-large` (รองรับภาษาไทย)
> **Vector DB:** Chroma, FAISS, หรือ Qdrant (ใช้ local ได้ ฟรี)
> **LLM:** Pathumma (thinking) หรือ OpenThaiGPT ผ่าน API
> **Framework:** LangChain, LlamaIndex, หรือ OpenThaiRAG (โอเพ่นซอร์สจาก AIEAT)

ตัวอย่าง RAG สั้น ๆ (Python + ChromaDB)

```
from openai import OpenAI
import chromadb
from sentence_transformers import SentenceTransformer

# 1. Embedding model รองรับภาษาไทย
embedder = SentenceTransformer("BAAI/bge-m3")

# 2. สร้าง vector DB
db = chromadb.Client().create_collection("company_docs")
```

```
docs = [
    "บริษัทเปิดทำการจันทร์-ศุกร์ 9:00-18:00",
    "ค่าจัดส่งสินค้าในกรุงเทพ 50 บาท ต่างจังหวัด 120 บาท",
    "ซื้อครบ 1,000 บาท ส่งฟรีทั่วประเทศ"
]
embeddings = embedder.encode(docs).tolist()
db.add(embeddings=embeddings, documents=docs, ids=["d1","d2","d3"])

# 3. ฟังก์ชันถาม-ตอบ
client = OpenAI(api_key="...", base_url="https://api.thaillm.or.th/v1")

def rag_ask(question):
    # Retrieve
    q_emb = embedder.encode([question]).tolist()
    results = db.query(query_embeddings=q_emb, n_results=2)
    context = "\n".join(results["documents"][0])

    # Generate
    response = client.chat.completions.create(
        model="OpenThaiGPT-ThaiLLM-8B-Instruct-v7.2",
        messages=[
            {"role": "system",
             "content": f"ตอบโดยใช้ข้อมูลต่อไปนี้เท่านั้น:\n{context}"},
            {"role": "user", "content": question}
        ],
        temperature=0.3
    )
    return response.choices[0].message.content

print(rag_ask("ซื้อครบเท่าไหร่ถึงส่งฟรี?"))
# → "ซื้อครบ 1,000 บาท ส่งฟรีทั่วประเทศครับ"
```

8.6 Prompt Engineering สำหรับภาษาไทย

แม้ ThaiLLM จะเข้าใจภาษาไทยดีกว่าโมเดลต่างชาติ แต่การเขียน prompt ที่ดียังช่วยเพิ่มคุณภาพได้อีกมาก

เทคนิคที่ใช้ได้ผลจริง

เทคนิค	ตัวอย่าง
Role Prompting	"คุณคือครูสอนคณิตศาสตร์ระดับมัธยม อธิบาย..."
Few-shot Examples	ให้ตัวอย่าง 2-3 คู่ `input→output` ก่อนถามจริง
Chain-of-Thought	เติม "คิดทีละขั้น" ต่อท้ายคำถาม (หรือใช้ /think)
Output Format	"ตอบเป็น JSON มี keys ดังนี้: ..."
Delimiters	ใช้ `"""` หรือ `###` แยก context กับคำสั่ง
ย้ำข้อจำกัด	"ห้ามตอบนอกเหนือจากข้อมูลที่ให้", "ถ้าไม่รู้ให้บอกว่าไม่รู้"

✅ Template prompt ที่ใช้ได้ดีกับ ThaiLLM

```
### บทบาท
คุณคือ [ระบุ role]

### งาน
[อธิบายงานให้ชัดเจน]

### ข้อจำกัด
- [ข้อ 1]
- [ข้อ 2]

### รูปแบบคำตอบ
[ระบุรูปแบบ เช่น JSON / bullet points / ตาราง]

### ข้อมูล
"""
[paste ข้อมูลที่ AI ต้องใช้]
"""

### คำถาม
[คำถามจริงที่ต้องการคำตอบ]
```

9 Outro และ Next Steps

ถึงเส้นชัยแล้ว — คุณพร้อมใช้ *ThaiLLM* ในงานจริง

9.1 สรุปสิ่งที่คุณได้

มาถึงตรงนี้ คุณได้เรียนรู้:

- ✅ เข้าใจว่า ThaiLLM คืออะไร และทำไมต้องมี AI ของคนไทย
- ✅ รู้จัก 9 องค์กรผู้พัฒนา และ LANTA supercomputer
- ✅ เลือกโมเดลที่เหมาะกับงานได้ (Pathumma / OpenThaiGPT / Typhoon-S / THaLLE)
- ✅ สมัคร API Key และเก็บอย่างปลอดภัย
- ✅ ติดตั้ง Continue + VSCode และ config เชื่อม ThaiLLM
- ✅ เขียน Python เรียก API ได้หลายรูปแบบ (สนทนา, แปล, สรุป, streaming)
- ✅ เข้าใจเทคนิคขั้นสูง: Thinking Mode, Tool Calling, RAG, Prompt Engineering

9.2 Roadmap ต่อไป

1. **สร้าง Side Project ของคุณเอง**

   เลือก 1 ปัญหาที่อยากแก้ในชีวิตประจำวัน แล้วสร้างด้วย ThaiLLM + Python เช่น: สรุปอีเมลให้, วิเคราะห์รีวิวสินค้า, บอทตอบคำถามลูกค้า Line OA

2. **ลองทำ RAG กับข้อมูลจริง**

   เช่น: เอกสารคู่มือของบริษัท, กฎหมายไทย (ประมวลกฎหมายแพ่งและพาณิชย์), ตำราเรียน

3. **Deploy เป็น Production**

   Upload Streamlit แอปขึ้น `streamlit.io` (ฟรี) หรือ FastAPI แอปขึ้น `render.com` / `railway.app`

4. **Fine-tune โมเดลของคุณเอง**

   ถ้างานคุณเฉพาะทางมาก ให้ download ThaiLLM-8B base จาก Hugging Face แล้ว fine-tune ด้วย LoRA บน Google Colab (ฟรี) หรือ LANTA (ต้องยื่นโครงการ)

5. **เข้าร่วมชุมชน**

   ThaiLLM Discord, Facebook Group "Thai AI Community", งาน NSTDA Annual Conference, AIEAT events

9.3 ฝากช่องทาง iCafeFX × SiamCafe.net

💜 ขอบคุณที่อ่านจนจบครับ

หากคู่มือฉบับนี้เป็นประโยชน์ รบกวนแชร์ต่อให้เพื่อน ๆ ที่สนใจ AI ไทยด้วยครับ

📺 **YouTube:** youtube.com/@icafefx — Subscribe เพื่อรับ tutorial ใหม่ ๆ

🌐 **iCafeFX:** icafeforex.com — เทรด Forex กับ Broker ชั้นนำ

💬 **คอมมูนิตี้:** siamcafe.net — พูดคุยเรื่อง IT, Trading, AI

📊 **Forex Signals:** xmsignal.com — สัญญาณเทรดรายวัน

ภาคผนวก A — Code Library

ชุดโค้ดพร้อมใช้สำหรับงานที่พบบ่อย

A.1 Helper Class (เรียกใช้ ThaiLLM ได้ง่าย ๆ)

```python
# thaillm_client.py
from openai import OpenAI
from dotenv import load_dotenv
import os, re, time

class ThaiLLMClient:
    MODELS = {
        "pathumma": "Pathumma-ThaiLLM-qwen3-8b-think-3.0.0",
        "openthai": "OpenThaiGPT-ThaiLLM-8B-Instruct-v7.2",
        "typhoon":  "Typhoon-S-ThaiLLM-8B-Instruct",
        "thalle":   "THaLLE-0.2-ThaiLLM-8B-fa"
    }

    def __init__(self):
        load_dotenv()
        self.client = OpenAI(
            api_key=os.getenv("THAILLM_API_KEY"),
            base_url=os.getenv("THAILLM_BASE_URL", "https://api.thaillm.or.th/v1")
        )

    def chat(self, prompt, model="openthai", system=None,
             temperature=0.7, max_tokens=2048, strip_think=True, **kw):
        msgs = []
        if system: msgs.append({"role":"system", "content":system})
        msgs.append({"role":"user", "content":prompt})

        resp = self.client.chat.completions.create(
            model=self.MODELS.get(model, model),
            messages=msgs, temperature=temperature, max_tokens=max_tokens, **kw
        )
        text = resp.choices[0].message.content
        if strip_think:
            text = re.sub(r'<think>.*?</think>', '', text, flags=re.DOTALL).strip()
        return text

    def stream(self, prompt, model="openthai", **kw):
        for chunk in self.client.chat.completions.create(
            model=self.MODELS.get(model, model),
            messages=[{"role":"user", "content":prompt}],
            stream=True, **kw
        ):
            if chunk.choices[0].delta.content:
                yield chunk.choices[0].delta.content

# ใช้งาน
```

```
if __name__ == "__main__":
    llm = ThaiLLMClient()
    print(llm.chat("สวัสดีครับ", model="openthai"))
    for w in llm.stream("เล่าเรื่องอาหารไทยสั้น ๆ"):
        print(w, end="", flush=True)
```

A.2 Retry with Backoff

```
import time, random
from openai import OpenAI, APIError, RateLimitError

def call_with_retry(client, max_tries=5, **kwargs):
    for attempt in range(max_tries):
        try:
            return client.chat.completions.create(**kwargs)
        except RateLimitError:
            wait = (2 ** attempt) + random.random()
            print(f"Rate limit — รอ {wait:.1f} วินาที")
            time.sleep(wait)
        except APIError as e:
            if attempt == max_tries - 1: raise
            time.sleep(2 ** attempt)
    raise Exception("Max retries exceeded")
```

A.3 JSON Structured Output

```
import json

def extract_info(text):
    response = client.chat.completions.create(
        model="OpenThaiGPT-ThaiLLM-8B-Instruct-v7.2",
        messages=[
            {"role": "system",
             "content": """ตอบเป็น JSON เท่านั้น ไม่มี markdown, ไม่มีคำอธิบาย
รูปแบบ:
{"name": "ชื่อคน", "age": อายุ, "city": "เมือง"}"""},
            {"role": "user", "content": text}
        ],
        temperature=0,
        max_tokens=200
    )
    # parse JSON พร้อม error handling
    content = response.choices[0].message.content.strip()
    # ลบ markdown code block ถ้ามี
    if content.startswith("```"):
        content = content.split("```")[1].replace("json", "").strip()
    return json.loads(content)
```

```
info = extract_info("สวัสดีครับ ผมชื่อสมชาย อายุ 35 ปี อยู่กรุงเทพ")
# → {"name": "สมชาย", "age": 35, "city": "กรุงเทพ"}
```

ภาคผนวก B — Troubleshooting

วิธีแก้ปัญหาที่พบบ่อย

อาการ / Error	สาเหตุและวิธีแก้
401 Unauthorized	API key ผิด หรือหมดอายุ/ถูก revoke แก้: ตรวจสอบว่า copy key ครบ ไม่มีช่องว่าง และยังไม่ได้ revoke
429 Too Many Requests	เรียก API ถี่เกินไป (rate limit) แก้: ใช้ retry with exponential backoff (ดู A.2)
404 Not Found	ชื่อ model ผิด หรือ endpoint URL ไม่ถูก แก้: ตรวจสอบ base_url มี `/v1` ลงท้าย และชื่อ model ตรงกับ playground
SSL Certificate Error *(พบบ่อยในไทย)*	Cloudflare บางครั้งบล็อก HTTPS จาก IP ไทย แก้: ลองเปลี่ยน DNS (1.1.1.1), ใช้ HTTP, หรือใช้ VPN ชั่วคราว
Response ช้ามาก (Pathumma)	Thinking mode ทำงาน ใช้โทเคนเยอะ แก้: เติม `/no_think` ในคำถามถ้าไม่ต้องการความแม่นยำสูง
คำตอบมี <think> tags	โมเดล thinking แสดง reasoning block แก้: ใช้ regex strip ออก (ดู A.1)
คำตอบ repeat / วนลูป	temperature=0 กับ thinking mode มักเกิด แก้: ตั้ง temperature=0.6-0.7, เพิ่ม presence_penalty=1.0-1.5
Context too long	รวม messages + response เกิน 32K tokens แก้: ลด system prompt, สรุป history, ใช้ RAG แทนส่งเอกสารทั้งหมด
Continue ไม่เห็น model	YAML syntax ผิด (indent, ตัวเล็ก/ใหญ่) แก้: เปิดใน VSCode ดู error ที่ไฮไลต์สีแดง, ตรวจ indent 2 spaces
Streamlit แสดงภาษาไทยเป็น □□□	ระบบไม่มี font ไทย แก้: ติดตั้ง font Sarabun / Noto Sans Thai ใน OS

B.2 การทดสอบ API แบบ Manual (ก่อนเขียนโค้ด)

ใช้ `curl` ทดสอบ API จาก terminal:

```
curl https://api.thaillm.or.th/v1/chat/completions \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -d '{
    "model": "Pathumma-ThaiLLM-qwen3-8b-think-3.0.0",
    "messages": [{"role": "user", "content": "สวัสดี"}],
```

```
    "temperature": 0.7,
    "max_tokens": 100
  }'
```

ถ้า curl ทำงาน แต่ Python error → ปัญหาอยู่ที่โค้ด ไม่ใช่ที่ API

ภาคผนวก C — Glossary

ศัพท์เทคนิคที่ปรากฏในคู่มือ

คำศัพท์	ความหมาย
LLM	Large Language Model — โมเดล AI ที่เข้าใจและสร้างภาษาได้
Foundation Model	โมเดลพื้นฐานที่ถูกสร้างขึ้นเพื่อต่อยอด (fine-tune) ให้เฉพาะทางได้
Parameter (พารามิเตอร์)	ค่าถ่วงน้ำหนักในโมเดล ยิ่งเยอะยิ่งซับซ้อน เช่น 8B = 8 พันล้าน
Token	หน่วยที่ AI ใช้ประมวลผล — ภาษาไทย 1 คำ ≈ 2-4 tokens
Context Length	ความยาวสูงสุดของ input + output ที่ AI รับได้ในครั้งเดียว
Temperature	ควบคุมความ "สุ่ม" ของคำตอบ (0=แน่นอน, 2=มั่ว)
Top-p / Top-k	วิธีเลือก token ถัดไป — จำกัดจากผู้สมัครที่มีความน่าจะเป็นสูง
Fine-tuning	การฝึกโมเดลเพิ่มด้วยข้อมูลเฉพาะเพื่อปรับให้เหมาะกับงาน
CPT	Continued Pre-Training — ฝึกโมเดลต่อจาก base model ด้วยข้อมูลใหม่
SFT	Supervised Fine-Tuning — ฝึกโมเดลด้วยคู่ question-answer
DPO	Direct Preference Optimization — ปรับโมเดลตาม feedback ของมนุษย์
LoRA	Low-Rank Adaptation — เทคนิค fine-tune ที่ประหยัดหน่วยความจำ
Chain-of-Thought (CoT)	ให้ AI คิดทีละขั้นก่อนตอบ ทำให้คำตอบแม่นยำขึ้น
Thinking Mode	โหมดที่ AI สร้าง `<think>` block ก่อนตอบ (ใน Qwen3/Pathumma)
Tool Calling / Function Calling	AI เรียกใช้ function ที่เราเขียนเพื่อดึงข้อมูลจริง
RAG	Retrieval-Augmented Generation — AI ค้นหาข้อมูลก่อนตอบ
Embedding	การแปลงข้อความเป็น vector (ตัวเลข) สำหรับค้นหาความหมาย
Vector Database	ฐานข้อมูลที่เก็บ embedding และค้นหาได้เร็ว (Chroma, FAISS, Qdrant)
OpenAI-compatible	API ที่ใช้รูปแบบเดียวกับ OpenAI — slot in ได้ด้วยโค้ดเดียวกัน
YaRN	Yet another RoPE extensioN — เทคนิคขยาย context length ของโมเดล
PDPA	พ.ร.บ.คุ้มครองข้อมูลส่วนบุคคลของไทย (เทียบ GDPR ของยุโรป)
Data Sovereignty	อธิปไตยทางข้อมูล — สิทธิควบคุมข้อมูลของประเทศ
HPC	High-Performance Computing — คอมพิวเตอร์สมรรถนะสูง (supercomputer)
PFlops	PetaFlop/s = 1,000 ล้านล้านคำสั่งต่อวินาที

ภาคผนวก D — Resources & Links

แหล่งข้อมูลอ้างอิงทั้งหมด

D.1 เว็บไซต์ทางการ

แหล่งที่มา	URL
ThaiLLM Playground	thaillm.or.th
ThaiLLM บน Hugging Face	huggingface.co/ThaiLLM
สวทช. (NSTDA)	nstda.or.th
NECTEC	nectec.or.th
ThaiSC (LANTA)	thaisc.io
AIEAT (OpenThaiGPT)	openthaigpt.aieat.or.th
SCB 10X (Typhoon)	opentyphoon.ai
KBTG Labs (THaLLE)	huggingface.co/KBTG-Labs
BDI	bdi.or.th/en/thaillm

D.2 Documentation & Tools

เครื่องมือ	URL
Continue.dev Docs	docs.continue.dev
OpenAI Python SDK	github.com/openai/openai-python
Qwen3 Docs	qwen.readthedocs.io
Streamlit	streamlit.io
LangChain	python.langchain.com
ChromaDB	trychroma.com

D.3 Technical Papers

- **Typhoon 2 Technical Report** — arxiv.org/abs/2412.13702
- **THaLLE-ThaiLLM Report** — arxiv.org/abs/2601.04597
- **Qwen3 Technical Report** — arxiv.org/abs/2505.09388
- **OpenThaiGPT benchmark** — github.com/OpenThaiGPT/openthaigpt_eval

D.4 ข่าวและประชาสัมพันธ์

- NSTDA — เปิดตัว ThaiLLM: nstda.or.th/home/news_post/s-and-t-implementation-thai-llm/
- Nation Thailand — ThaiLLM launched: nationthailand.com/blogs/business/tech/40065259
- Thansettakij — เปิด ThaiLLM AI พันธุ์ไทย: thansettakij.com/technology/ai/657014

เกี่ยวกับผู้สอน

อ.บอม

iCafeFX × SiamCafe.net

30+ ปีใน IT & Forex

ประวัติโดยย่อ

อ.บอม คือผู้ก่อตั้ง **SiamCafe.net** ตั้งแต่ปี พ.ศ. 2540 (ค.ศ. 1997) ซึ่งเป็นหนึ่งใน community IT ที่เก่าแก่ที่สุดของประเทศไทย และผู้ก่อตั้ง **iCafeFX** แพลตฟอร์มด้าน Forex Trading

ประสบการณ์มากกว่า **30 ปี** ในด้าน IT และ Forex:

- Partner ระดับ **XM Global Legend** มากกว่า 13 ปี
- เจ้าของโดเมน icafeforex.com, xmsignal.com
- ผู้พัฒนา iCafeFX iOS App (App Store ID: 6759817119)
- ผู้สอนและผู้เผยแพร่ความรู้ด้าน Trading และ AI บน YouTube

สิ่งที่เขียน / ทำ

- คู่มือ ThaiLLM ฉบับนี้ (คุณกำลังอ่าน)
- Tutorial ด้าน Forex, MT5, Elliott Wave บน iCafeFX
- ชุดเครื่องมือ AI Content Pipeline (yt-tutorial-kit)
- Trading signals และการวิเคราะห์กราฟรายวัน

🙏 ขอบคุณและติดตาม

ขอบคุณทุกท่านที่อ่านมาจนจบครับ หากมีคำถาม ข้อแนะนำ หรือพบข้อผิดพลาดในเอกสาร สามารถติดต่อได้ทางช่องทางใดก็ได้ของ iCafeFX

YouTube: youtube.com/@icafefx · **Facebook:** fb.com/icafefx
Website: icafeforex.com · **Community:** siamcafe.net

🇹🇭 **AI ของคนไทย สอน AI ของคนไทย — วิทยาทาน 2026** 🇹🇭

🔗 ลิงก์ทั้งหมด & ช่องทางติดตาม

ทุกลิงก์ในส่วนนี้ **ตรวจสอบและยืนยันว่าใช้งานได้จริง** เมื่อ *22* เมษายน *2026* ✓ VERIFIED

🙏 ขอบคุณที่อ่านจนจบ — ช่วยแชร์ต่อคือวิทยาทาน 🇹🇭

คู่มือนี้ทำขึ้นเพื่อเผยแพร่ความรู้ฟรี หากรู้จักคนที่สนใจ AI ไทย ช่วยส่งต่อเพื่อขยายชุมชนนักพัฒนาไทย

⭐ ช่องทางหลัก iCafeFX Network ✓ ทดสอบแล้ว

🌐 เว็บไซต์หลัก

iCafeFX Main	icafeforex.com — สอนเทรด Forex ฟรี โดย อ.บอม
Blog	icafeforex.com/blog/
เกี่ยวกับ อ.บอม	icafeforex.com/about/
ติดต่อเรา	icafeforex.com/contact/
สมาชิก Portal	member.icafeforex.com

📲 iCafeFX Mobile App (iOS + Android)

Landing Page	icafeforex.com/icafefx-app/
App Store (iOS)	apps.apple.com/app/id6759817119
Google Play	play.google.com/store/apps/details?id=com.icafeforex.icafefx

iCafeFX Mobile App v3.2.x — Real-time Forex + Gold + AI Signals · ฟรีตลอดชีพ · *28* ภาษา

📺 Social Media

YouTube	youtube.com/@icafefx
Subscribe	youtube.com/@icafefx?sub_confirmation=1
คลิปคู่มือ ThaiLLM	youtube.com/watch?v=MCBwEexM5C4
Facebook Page	facebook.com/icafefx
Facebook Group	facebook.com/groups/cafefx
FB Messenger	m.me/icafefx
LINE Official	line.me/R/ti/p/@icafefx · @icafefx

🕸️ iCafeFX Network Sites (Partner Sites)

SiamCafe.net	siamcafe.net — คอมมูนิตี้ IT ไทย (ก่อตั้ง 1997)
SiamCafe Blog	siamcafe.net/blog/
XM Signal	xmsignal.com — สัญญาณเทรด XM
Siam2R	siam2r.com
SiamLanCard	siamlancard.com
iCafe Cloud	icafecloud.com
CafeNetwork	cafenetwork.net — รับโบนัส $30

🤖 EA & Trading Products

EA Forex (ทั้งหมด)	icafeforex.com/ea-forex-landing/
REDHAT WARP EA	icafeforex.com/redhat-warp-ea/
REDHAT Indicator	icafeforex.com/redhat-indicator-v20/
คู่มือ REDHAT	icafeforex.com/redhat-manual/
MT5 2026	icafeforex.com/mt5/
iCafe VPS Service	icafeforex.com/icafevpsservice/

📖 ลิงก์อ้างอิง ThaiLLM & เครื่องมือ

แหล่งข้อมูลทางการทั้งหมด ✓ ทางการ

🇹🇭 ThaiLLM & หน่วยงานพัฒนา

องค์กร / แหล่ง	URL
ThaiLLM Playground	thaillm.or.th
ThaiLLM API	api.thaillm.or.th/v1
ThaiLLM on Hugging Face	huggingface.co/ThaiLLM
ThaiLLM-8B Base Model	huggingface.co/ThaiLLM/ThaiLLM-8B
สวทช. (NSTDA)	nstda.or.th
NECTEC	nectec.or.th
ThaiSC / LANTA	thaisc.io
LANTA Specs	thaisc.io/thaisc-resorces/lanta
BDI	bdi.or.th/en/thaillm
OpenThaiGPT (AIEAT)	openthaigpt.aieat.or.th
Typhoon (SCB 10X)	opentyphoon.ai
KBTG Labs	huggingface.co/KBTG-Labs
VISTEC	vistec.ac.th

🤖 4 โมเดลโดยตรง (Hugging Face)

โมเดล	URL
Pathumma (NECTEC)	huggingface.co/nectec/Pathumma-ThaiLLM-qwen3-8b-it-2.0.0
Pathumma Vision	huggingface.co/nectec/Pathumma-llm-vision-2.0.0-preview
OpenThaiGPT ThaiLLM	huggingface.co/openthaigpt/openthaigpt-thaillm-8b-instruct-v0.7.2-research-preview
Typhoon-S ThaiLLM	huggingface.co/typhoon-ai/typhoon-s-thaillm-8b-instruct-research-preview
THaLLE ThaiLLM	huggingface.co/KBTG-Labs/THaLLE-0.2-ThaiLLM-8B-fa

📰 ข่าวและเอกสาร

หัวข้อ	URL
NSTDA - เปิดตัว ThaiLLM (Eng)	nstda.or.th/en/news/news-years-2026/thaillm.html
NSTDA - เปิดตัว ThaiLLM (ไทย)	nstda.or.th/home/news_post/s-and-t-implementation-thai-llm
Nation Thailand	nationthailand.com/blogs/business/tech/40065259
Thansettakij	thansettakij.com/technology/ai/657014
LANTA TOP500	top500.org/system/180125
THaLLE arXiv	arxiv.org/abs/2601.04597
Typhoon 2 arXiv	arxiv.org/abs/2412.13702
Qwen3 Technical Report	arxiv.org/abs/2505.09388

💻 VSCode & Extensions

เครื่องมือ	URL
VSCode Download	code.visualstudio.com/download
VSCode Docs	code.visualstudio.com/docs
VSCode GitHub	github.com/microsoft/vscode
VSCode Marketplace	marketplace.visualstudio.com
Python Extension	marketplace.visualstudio.com/items?itemName=ms-python.python
Continue Extension	marketplace.visualstudio.com/items?itemName=Continue.continue
GitLens	marketplace.visualstudio.com/items?itemName=eamodio.gitlens
Material Icon Theme	marketplace.visualstudio.com/items?itemName=PKief.material-icon-theme
Error Lens	marketplace.visualstudio.com/items?itemName=usernamehw.errorlens
Prettier	marketplace.visualstudio.com/items?itemName=esbenp.prettier-vscode
DotENV	marketplace.visualstudio.com/items?itemName=mikestead.dotenv

🐍 Python & AI Libraries

Library / Tool	URL
Python Official	python.org
Python Downloads	python.org/downloads
OpenAI Python SDK	github.com/openai/openai-python
Continue.dev Docs	docs.continue.dev
Qwen3 Docs	qwen.readthedocs.io
Streamlit	streamlit.io
LangChain Python	python.langchain.com
ChromaDB	trychroma.com
Hugging Face Transformers	huggingface.co/docs/transformers
BGE-M3 Embedding	huggingface.co/BAAI/bge-m3

⚡ iCafeFX × SiamCafe.net

โดย อ.บอม (กิตติทัศน์ เจริญพนาสิทธิ์)

Professional AI & Trading Tutorial Series · Est. 1997

icafeforex.com · youtube.com/@icafefx · siamcafe.net · xmsignal.com