สรุปวิธีการใช้งานทั้งหมดของระบบที่เราพัฒนาร่วมกัน อย่างเป็นระบบ เข้าใจง่าย และพร้อมใช้งานจริง

🎯 ระบบแปลนิยายจีน → ไทย (เวอร์ชันอัจฉริยะ)

> ✅ รองรับ:
>
> - การแปลใหม่
> - การอัปเดตบทเก่าเมื่อแก้ `glossary.yaml`
> - การควบคุมเวอร์ชันด้วย Git
> - การตรวจสอบและรายงานผลอัตโนมัติ

🔧 ส่วนประกอบหลักของระบบ

trans1.py	แปลบทใหม่
reprocess.py	อัปเดตบทแปลเดิมที่ "ล้าสมัย" จาก glossary
reprocess_all_outdated.py	ตรวจสอบและอัปเดตทุกบทที่ต้องการ
.git/hooks/pre-push	รันอัตโนมัติเมื่อ push (optional)
diff_report.html	แสดงความเปลี่ยนแปลงแบบเห็นภาพ
.meta/	เก็บ metadata เช่น เวอร์ชัน glossary ที่ใช้
glossary.yaml	พจนานุกรมศัพท์เฉพาะ (ชื่อ, สำนัก, วิชา, ตำแหน่ง)

📦 โครงสร้างโฟลเดอร์แนะนำ

```
Trans1/
├── chapters_src/              # ต้นฉบับภาษาจีน (.txt)
├── chapters_translated/       # บทแปลภาษาไทย
│   └── .meta/                 # metadata ของแต่ละบท (JSON)
├── logs/                      # log การแปล, cache, new_terms
├── reports/                   # diff_report.html
├── backups/                   # สำรอง glossary
├── trans1.py                  # แปลบทใหม่
├── reprocess.py               # อัปเดตบทเดียว
├── reprocess_all_outdated.py  # อัปเดตหลายบทอัตโนมัติ
├── glossary.yaml              # พจนานุกรมหลัก
└── .git/                      # ใช้ git จัดการเวอร์ชัน
```

🚀 วิธีใช้งานตามสถานการณ์

✅ 1. แปลบทใหม่ (เช่น บทที่ 0006)

```bash
python trans1.py chapters_src/sa_0006.txt
```

➡️ ระบบจะ:

- โหลด `glossary.yaml`
- ตรวจหาคำใหม่
- แปลโดยใช้ cache + SQLite
- บันทึกบทแปลใน `chapters_translated/`
- บันทึก metadata ใน `.meta/sa_0006.json` (รวม commit ของ glossary)

✅ 2. แก้ `glossary.yaml` (เช่น เปลี่ยน 'จางซาน' → 'จางเซียนเชิง')

```yaml
张三:
    th: จางเซียนเชิง
    source_type: manual
九阳神功:
    th: ศาสตร์กระบี่เก้ามังกร
```

แล้ว commit:

```bash
git add glossary.yaml
git commit -m "อัปเดตชื่อเอกและวิชา"
```

✅ 3. อัปเดตบทที่ได้รับผลกระทบ

ทางเลือก 3.1: อัปเดตทีละบท

```bash
python reprocess.py chapters_src/sa_0001.txt
```

ทางเลือก 3.2: อัปเดตทกบทที่ล้าสมัย (แนะนำ)

```bash
python reprocess_all_outdated.py
```

➡️ ระบบจะ:

- ตรวจสอบ metadata ของทุกบท
- ถ้าบทนั้นใช้ glossary เวอร์ชันเก่า → รัน `reprocess.py`
- แปลเฉพาะประโยคที่มีคำเปลี่ยน (Selective Retranslation)
- สร้างไฟล์แปลใหม่
- สร้าง `diff_report.html`

✅ 4. ดูรายงานการเปลี่ยนแปลง

เปิดไฟล์:

```
reports/diff_report_YYYYMMDD_HHMMSS.html
```

ใน browser เพื่อดู:

- ประโยคไหนเปลี่ยน
- เปลี่ยนจากอะไร → เป็นอะไร
- เพราะคำใดใน glossary เปลี่ยน

เหมาะสำหรับ:

- ตรวจสอบคุณภาพ (QA)
- แชร์กับทีมงาน
- เก็บประวัติการแก้ไข

✅ 5. ป้องกันข้อผิดพลาดด้วย Git Hook (Optional)

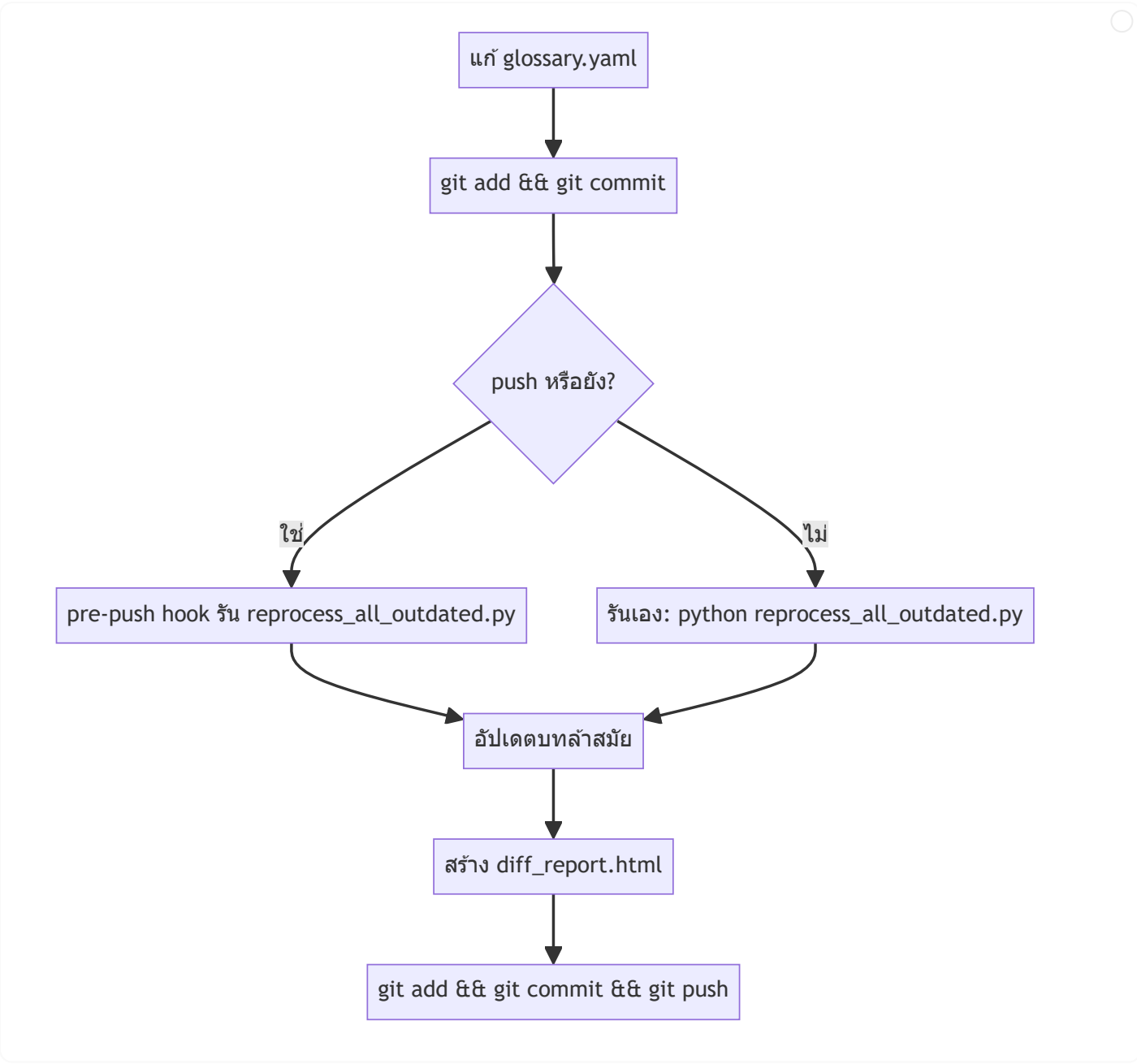
ตั้งค่า `pre-push` hook เพื่อให้ระบบ รัน `reprocess_all_outdated.py` อัตโนมัติก่อน push

```bash
chmod +x .git/hooks/pre-push
```

➡️ ผลลัพธ์:

- หากลืมอัปเดตบทแปล → push ไม่ได้
- ทุกคนในทีมได้ข้อมูลสอดคล้องกัน

🔄 Workflow สรุป (ภาพรวม)



💡 เคล็ดลับการใช้งาน

ใช้ `--dry-run`	ดูว่าจะอัปเดตบทไหนก่อนทำจริง

ใช้ tqdm	ดูความคืบหน้าขณะแปล
ใช้ SQLite	แปลเร็วขึ้นเมื่อมีคำศัพท์มาก
ใช้ Git	ย้อนกลับได้หากแก้ผิด
ใช้ Aho-Corasick	ตรวจจับคำเฉพาะได้เร็วมาก

✅ ประโยชน์ที่ได้รับ

ไม่ต้องแปลทั้งบทซ้ำเมื่อแก้ glossary	✅
ไม่พลาดบทที่ต้องอัปเดต	✅
รู้ว่า "คำไหนเปลี่ยนไปบ้าง"	✅
ทำงานหลายคนได้โดยไม่ชนกัน	✅
ระบบอัตโนมัติ ลดงานซ้ำซ้อน	✅

🛠️ สิ่งที่สามารถพัฒนาต่อ

📊 Dashboard	แสดงสถานะทุกบท: ทันสมัย / ล้าสมัย
🤖 CI/CD (GitHub Actions)	รัน reprocess บนเซิร์ฟเวอร์อัตโนมัติ
💬 แจ้งเตือน (Line/Discord)	เมื่อมีการอัปเดต
📥 Export เป็น EPUB/PDF	สำหรับแจกหรือขาย
🔍 Search engine สำหรับ glossary	ค้นหาศัพท์ได้เร็ว

✅ สรุปสุดท้าย

> คุณไม่ได้มีแค่ "สคริปต์แปลนิยาย"
> แต่คุณมี **ระบบจัดการแปลระดับโปร** ที่:
>
> - 🔄 อัปเดตอัตโนมัติ
> - 🏷️ ควบคุมเวอร์ชันได้
> - 📊 ตรวจสอบผลลัพธ์
> - 👥 ใช้งานร่วมกันได้หลายคน

หากต้องการ ผมสามารถช่วย:

- **แพ็คทั้งหมดเป็นโปรแกรม** GUI
- **ทำเวอร์ชันเว็บ** (Web App)
- **สร้าง** installer **สำหรับ** Windows/Mac