Poetry로 Python 프로젝트 환경관리 - 정리 가이드

> - **2025.10 기준**
> - `Mac` + `pyenv` + `VSCode` + **`Poetry`** 기준
> - 수백개 패키지가 충돌하는 `requirements.txt` → `Poetry` 환경 전환 → 충돌 관리 시도 사례

---

1. Poetry 설치 (최초 1회만)

```
# 맥 터미널에 아래 한 줄!
curl -sSL https://install.python-poetry.org | python3 -
```

- 기본 설치 경로: `~/.local/bin/poetry`
- 설치 확인:

```
poetry --version
```

- *(필요시) 경로 추가*

```
export PATH="$HOME/.local/bin:$PATH"
# ~/.zshrc 또는 ~/.bashrc에 추가
```

---

2. 프로젝트별 `pyenv` + `Poetry` 환경 세팅

```
# 1. pyenv로 원하는 Python 버전 설치/선택
pyenv install 3.13.5
pyenv local 3.13.5

# 2. 프로젝트 폴더 이동
cd ~/dev/my_project

# 3. Poetry 프로젝트 초기화(pyproject.toml 생성)
poetry init
```

---

3. 대량 requirements.txt → Poetry로 전환

➤ 안전백업 후 일괄 추가

```
cp requirements.txt requirements_backup.txt

# 주석/빈줄 제외 모든 패키지 poetry로 한 번에 추가(실패시 나눠서 반복!)
poetry add $(cat requirements.txt | grep -v "^#" | grep -vE '^\s*$')
```

- **en_core_web_sm** 등 PyPI에 없는 패키지는 자동 `skip` → 별도 처리(아래 참고!)

➤ group add가 너무 많거나 일부 충돌시

- 핵심, 꼭 필요한 패키지 위주로만 먼저 `add`
- 충돌/에러는 에러 메시지 기준으로 개별 해결

---

4. spaCy 모델/미지원 패키지 처리

- `spaCy` 언어모델은 `pip` or `poetry`로 설치 불가! → 아래처럼 **직접 설치**

```
# 오류 발생 시
poetry add requests

# 직접 설치
python -m spacy download en_core_web_sm
```

---

5. 가상환경 진입 및 실행

- `Poetry 2.x` 환경에서는 직접 `venv activate` (shell 명령 대신):

```
source $(poetry env info --path)/bin/activate
```

or

```
poetry env info --path    # 가상환경 경로 확인
cd [위 경로]/bin
source activate
```

- `VSCode`에서도 python 해설 인터프리터를 `.venv`/pyenv 환경으로 지정

---

6. 패키지 추가/삭제 및 환경 관리

- **필수 패키지는 `poetry add`로, 누락/ImportError 나오면 그때그때 추가!**

```
    poetry add streamlit   # 예시
    poetry remove numpy    # 삭제 예시
```

---

7. requirements.txt/환경 export (공유용, 서버 자동화 등)

- export 플러그인 설치(최초 1번)

```
    poetry self add poetry-plugin-export
```

- export 실행 (새 requirements.txt 덮어쓰기)

```
    poetry export --format requirements.txt --output requirements.txt --
without-hashes
```

- 이 결과에는 **실제로 Poetry에서 관리 중인(설치된) 패키지만** 들어감!

---

8. git 관리

- **pyproject.toml/poetry.lock은 깃에 반드시 커밋(버전관리)!**
- .venv/ 등 실제 가상환경 폴더는 .gitignore에 포함

```
/`.gitignore`
    .venv/
    __pycache__/
    *.pyc
    .DS_Store
```

---

9. 헷갈림/실행 확인 팁

```
    which python
    python --version
    pip list
    poetry show
```

- 패키지 목록/버전/경로가 pyenv + Poetry 환경 가리키면 OK!

---

10. 추가/실무 팁

- `requirements.txt`는 최종적으로만 export해서 공유/서버 자동화용
- 실사용은 반드시 `poetry add`/`remove`만 이용!
- 대량의 패키지 마이그레이션은 한 번에 다 안될 수도 있고, 점진적으로 해결
- en_core_web_sm과 같은 spaCy 모델/특수 모델은 별도 설치해야 함

🚩 *최종 요약*

- `pyenv` → **`Python` 버전 관리 + 선택**
- `Poetry` → **`가상환경` / `패키지 버전` / `충돌 자동 관리`**
- 패키지 추가/삭제 → *무조건* **`poetry add` / `remove`**
- `requirements.txt` → export로만 사용!
- `pyproject.toml` / `poetry.lock` → **`커밋 필수!`**
