Virtuális környezetek kezelése Poetry-vel

Szathmáry László

2023. jún. 12.

Ismétlés

Virt. körny. létrehozása hagyományos módszerrel:

```
# for plain old virt. env.'s
alias venv_init="python3 -m venv .venv"
alias venv_freeze="pip freeze --local"
alias on="source .venv/bin/activate"
alias off="deactivate"
```

Korábbi videók:

- Python virtuális környezetek (1. rész)
- Eldobható Python virtuális környezet használata (2. rész)

Ismétlés

- hozzunk létre egy virt. körny.-et
- aktiváljuk, majd
- telepítsük a köv. csomagokat:

```
flask
requests
psutil
```

- rögzítsük a telepített csomagok listáját (`requirements.txt`)

Ismétlés

Probléma #1

A virt. körny. az adott mappában lett létrehozva.

- elég nagy a mérete
- máshol is létre lehetne hozni, de az nehézkessé tenné az aktiválást

Probléma #2

A `requirements.txt` **minden** telepített csomagot tartalmaz! Mi csak 3 csomagot tettünk fel, de ezek függőségei is listázva lettek!

- töröljük le az előbbi 3 csomagot
- Üres lett a virtuális környezet?

Poetry

A Poetry (https://python-poetry.org/) projekt mindkét előző problémára megoldást nyújt.

"*Poetry is a tool for dependency management and packaging in Python. It allows you to declare the libraries your project depends on and it will manage (install/update) them for you. Poetry offers a lockfile to ensure repeatable installs, and can build your project for distribution.*" (docs)

Poetry

Tartalom

1. Telepítés
    - Ubuntu Linux alá
2. Használat
3. Telepítés
    - Manjaro Linux alá
    - Windows alá

Poetry

Telepítés Ubuntu Linux alá

Telepítéshez **NE** az Ubuntu csomagkezelőjét (`apt`) használjuk! Miért? Az Ubuntu repóiban túl régi szoftververziók vannak fent. **Javasolt** telepítési mód: `pipx`

- `pip` telepítése, ha hiányozna: `$ sudo apt install python3-pip`
- `pipx` telepítése:

```
$ pip3 install pipx --user -U
```

- `pipx` indítása:

```
$ pipx
```

Poetry

Telepítés Ubuntu Linux alá

- Ha nem találja a `pipx` parancsot, akkor a `~/.local/bin` mappát manuálisan adjuk hozzá a PATH-hoz
- Nyissunk egy új terminált és most már indulnia kell:

```
$ pipx
```

Poetry

Telepítés Ubuntu Linux alá

- A poetry csomagot a pipx-szel érdemes feltenni:

```
$ pipx install poetry
```

(Ennél a parancsnál kaptam egy hibaüzenetet, hogy előbb tegyem fel a `python3.10-venv` csomagot (`sudo apt install python3.10-venv`). Ezután már probléma nélkül lefutott a fenti parancs.)

- pipx-szel telepített csomagok listája:

```
$ pipx list
```

Poetry

Telepítés Ubuntu Linux alá

- Indítsuk el a poetry-t:

```
$ poetry

$ poetry -V
```

Poetry

Használat

Most csak azt nézzük meg, hogy hogyan lehet a poetry-vel egy virt. körny.-et kezelni (létrehozás, csomagok telepítése, csomagok törlése, stb.).

```
$ poetry init

# vagy:

$ poetry init -n
```

Ez létrehoz egy `pyproject.toml` nevű konfigurációs állományt. A második esetben nem kérdez semmit.

Poetry

Használat

Ezután hozzuk létre a virt. körny.-et:

```
$ poetry shell
```

Ez

1. létrehozza a virt. körny.-et, ill.
2. indít egy subshell-t (ebből lépjünk ki `Ctrl+D` -vel)

Poetry

Használat

Vegyük észre, hogy a virt. körny. **NEM** az aktuális könyvtárban jött létre, hanem el lett választva a projektmappától.

Helye: `~/.cache/pypoetry/virtualenvs`

Erre tegyünk egy szimbolikus linket, különben állandóan keresgélni fogjuk:

```
$ cd
$ ln -s .cache/pypoetry/virtualenvs .virtualenvs
```

Poetry

Használat

A projektmappában hogyan tudjuk lekérdezni, hogy hol van a virt. körny. mappája?

```
$ poetry env info -p
```

Definiáljunk két hasznos alias-t:

```
# for Poetry
poetry_on () {
    source "`poetry env info -p`/bin/activate"
}
alias pon="poetry_on"
alias poff="deactivate"
```

Poetry

Használat

Házi feladat: készítsünk egy `on` nevű alias-t, ami mindkét esetben működik! Vagyis a hagyományos virt. körny.-ekkel, ill. a poetry által kezelt virt. körny.-ekkel is legyen kompatibilis.

Tipp:

- az alias egy függvényt hívjon meg
- a függvényen belül nézzük meg, hogy van-e `.venv` mappa. Ha igen, akkor... Különben, ha van `pyproject.toml` file, akkor... Ha egyik sincs, akkor hiba.

(lásd demó)

Poetry

Használat

Telepítsük a köv. csomagokat:

```
flask
requests
psutil
```

Figyelem! Itt **nem** a `pip` paranccsal fogunk csomagokat telepíteni, hanem mindent a `poetry` paranccsal végzünk!

Megjegyzés #1: a `poetry` a háttérben a `pip`-et használja

Megjegyzés #2: elég csak belépni a projektmappába, a `poetry` parancs használatakor nem muszáj aktiválni a virt. körny.-et

Poetry

Használat

```
$ poetry add flask requests psutil
```

A `pyproject.toml` file automatikusan frissül.

Létrejön egy `poetry.lock` nevű file is, ami a telepített csomagokról tartalmaz információt. Ezt **ne** módosítsuk kézzel, a poetry automatikusan generálja / frissíti.

A verziókövető rendszerbe (pl. GitHub) *mindkét* file-t fel kell majd tölteni.

Poetry

Használat

```
$ cat pyproject.toml

[tool.poetry.dependencies]
python = "^3.10"
flask = "^2.3.2"
requests = "^2.31.0"
psutil = "^5.9.5"
```

Csak az általunk telepített csomagok vannak ott! Ezen csomagok függőségei számon vannak tartva, de itt **nem** szerepelnek!

Poetry

Használat

Töröljünk egy csomagot:

```
$ poetry remove flask
```

1. törli a csomagot a *függőségeivel együtt*
2. frissíti a `pyproject.toml` állományt

Poetry

Használat

Hasznos parancsok

```
$ poetry show                       # telepített csomagok listája

$ poetry show --tree                # csomagok közti függőségek

$ poetry show -o
$ poetry show --outdated            # Mely csomagokból van újabb verzió?

$ poetry update <pkg>               # adott csomag frissítése

$ poetry update                     # összes csomag frissítése
                                    # (a semver szabályait figyelembe véve)
```

Poetry

Használat

A telepítendő csomagok között lehetnek olyanok is, amik csak a fejlesztéshez kellenek (pl. `mypy`). Ha valaki csak futtani akarja az alkalmazást, annak ezek a fejlesztői csomagok nem kellenek.

A fejlesztői csomagokat egy külön csoportba érdemes telepíteni:

```
$ poetry add mypy --group dev
```

Poetry

Használat

```
$ cat pyproject.toml

[tool.poetry.dependencies]
python = "^3.10"
requests = "^2.31.0"
psutil = "^5.9.5"


[tool.poetry.group.dev.dependencies]
mypy = "^1.3.0"
```

Poetry

Használat

Tegyük fel, hogy a GitHub-ról letöltöttünk egy poetry által menedzselt projektet, amit az otthoni gépünkön használni szeretnénk. Hozzunk létre egy virt. körny.-et a projektnek, ill. telepítsük a szükséges csomagokat.

```
$ poetry install                # a dev. csomagokat is felteszi

$ poetry install --no-dev       # a dev. csomagokat NEM teszi fel
```

Poetry

Használat

Help rendszer:

```
$ poetry help                    # általános help

$ poetry help add                # a `poetry add` saját help-je
```

Poetry

Semantic Versioning

```
$ cat pyproject.toml

[tool.poetry.dependencies]
python = "^3.10"
requests = "^2.31.0"
psutil = "^5.9.5"
```

Az egyes csomagok verziói is rögzítve vannak. A verziók megadása a *semantic versioning* rendszert követi (lásd https://semver.org).

Például az `^1.2.3` jelentése: balról jobbra haladva a legelső nem nulla érték nem változhat, azaz ezekre frissülhet: `>=1.2.3` és `<2.0.0`

Poetry

Telepítés Manjaro Linux alá

Lehet használni a Manjaro saját csomagkezelőjét (`pacman` vagy `yay`).

Vagy, ha mindenből a legfrissebb verzió kell, akkor járjunk el úgy, mint Ubuntu esetén.

Poetry -- Telepítés Windows alá

- a `pip` parancs adott
- pip-pel tegyük fel a `pipx` -et
  - a `pipx ensurepath` parancs frissíti a PATH-t, vagy
  - manuálisan adjuk hozzá a PATH-hoz a `C:\Users\<username>\.local\bin` mappát
- pipx-szel tegyük fel a `poetry` -t
- Ha indul a `poetry` , akkor innen minden ugyanaz, mint Linux alatt.
- A virt. körny.-ek itt fognak létrejönni:
  `C:\Users\<username>\AppData\Local\pypoetry\Cache\virtualenvs`

Related Work

Pipenv

Link: https://github.com/pypa/pipenv

"*Pipenv is a Python virtualenv management tool that supports a multitude of systems and nicely bridges the gaps between pip, python (using system python, pyenv or asdf) and virtualenv. Linux, macOS, and Windows are all first-class citizens in pipenv.*" (github)

Gyakorlás

Vegyünk egy GitHub projektet (pl. https://github.com/jabbalaci/ClosestX11Color), töltsük le, majd futtassuk a gépünkön.

Próbáljuk ki mindkét módszert:

1. Kialakítunk egy **hagyományos** virt. körny.-et, s abban futtatjuk az alkalmazást.
2. Kialakítunk egy **Poetry** által menedzselt virt. körny.-et, s abban futtatjuk az alkalmazást.

Extra:

- próbáljuk ki Windows alatt is

Tipp

Linux alatt be lehet vezetni egy alias-t a `poetry` parancsra:

```
alias po="poetry"
```

Ezt a sort tegyük be a `~/.bashrc` állomány végére.

Függelék

```
# javasolt ~/.bashrc beállítások:

alias d='ls -al'
alias p="python3"
alias p3="python3"
alias mc='. /usr/lib/mc/mc-wrapper.sh'
alias po="poetry"

export PATH=$PATH:$HOME/.local/bin

# for plain old virt. env.'s
alias venv_init="python3 -m venv .venv"
alias venv_freeze="pip freeze --local"
alias on="source .venv/bin/activate"
alias off="deactivate"

# for Poetry
poetry_on () {
    source "`poetry env info -p`/bin/activate"
}
alias pon="poetry_on"
alias poff="deactivate"
```

Vége

Köszönöm a figyelmet!