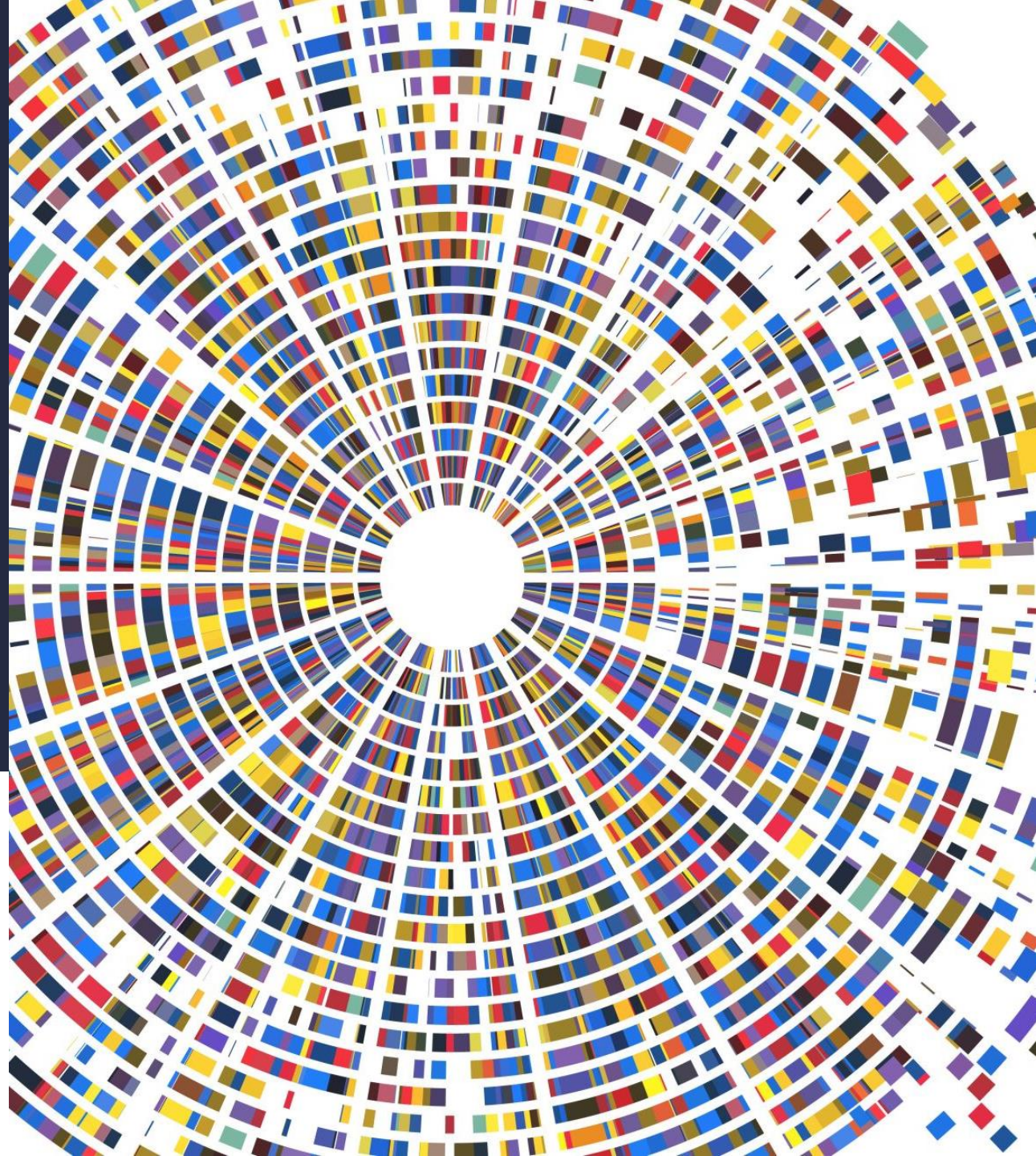
Einführung in Pipenv

automatisiert die Erstellung und Verwaltung virtueller Umgebungen und speichert Abhängigkeiten in einer Pipfile und einer Pipfile.lock, anstelle von requirements.txt.



Installation von pipenv

Um pipenv auf dem System zu installieren, installieren wir es mit pip

pip install pipenv

Um die erfolgreiche Installation zu prüfen

pipenv version

Projekt initialisieren

Um ein neues Projekt zu initialisieren, kann wie folgt vorgegangen werden, falls kein Projekt bisher existiert

ein neues, leeres Verzeichnis erstellen:

mkdir my_project && cd my_project

ein Pipfile installieren, dass das Verzeichnis als pipenv Projektverzeichnis kennzeichnet

pipenv install

Pipfile

Im Zuge der Initialisierung wurde ein Pipfile angelegt. In diesem werden wir später die Abhängigkeiten notieren.

```
[[source]]
url = "https://pypi.org/simple"
verify_ssl = true
name = "pypi"

[packages]
[dev-packages]
[requires]
python_version = "3.11"
```

Was genau ist das Pipfile?

Das Pipfile ist eine Datei, die von pipenv verwendet wird, um die Abhängigkeiten eines Python-Projekts zu definieren. Er ersetzt die klassische requirements.txt und bietet zusätzliche Vorteile wie:

Klare Trennung von Produktions- und Entwicklungs-Abhängigkeiten.

Einfache Lesbarkeit durch ein benutzerfreundliches TOML-Format.

Ein Pipfile wird automatisch erstellt, wenn du ein neues pipenv-Projekt initialisierst oder Pakete installierst.

Vorteile gegenüber requirements.txt

Das Pipfile ist besser organisiert und unterscheidet zwischen **Produktions- und Entwicklungs-Abhängigkeiten.**

Im Pipfile kann die **benötigte Python-Version** festgelegt werden.

Python Version

Wenn das Projekt mit dem vorherigen Befehl initialisiert wird, wird die aktuelle Python-Version genutzt, die in das Pipfile eintegragen wird.

Falls eine andere Python-Version gewünscht ist, muss das Projekt anders initialisiert werden.

allerdings muss diese Python Version auf dem Rechner vorhanden sein.

pipenv --python 3.13.1

pipenv install

Installation eines Pakets

Pakete können wir wie folgt installieren. Das Pipfile wird automatisch geändert.

für alle Umgebungen

pipenv install <paketname>

oder in der Entwicklungsumgebung (dev)

pipenv install pytest --dev

In der Pipfile erscheinen die Dev-Einträge unter [dev-packages]

Pipfile.lock

Die Datei Pipfile.lock stellt sicher, dass Abhängigkeiten exakt reproduzierbar installiert werden. Dabei handelt es sich um eine **AUSSCHLIESSLICH** automatisch generierte Datei in Projekten, die mit pipenv verwaltet werden. Sie dient dazu, exakt reproduzierbare Python-Umgebungen zu erstellen.

Während das **Pipfile** lose die Versionsnummer für ein Paket angibt, z.B. ">=2.20.0" für jede Version größer als 2.20.0 speichert das **Pipfile.lock** exakt die runtergeladene Version, die mit den anderen Paketen im Projekt aufgelöst wurde.

Vorteile des Lockfiles

Mit dem Pipfile.lock können alle Entwickler exakt die gleiche Umgebung installieren, unabhängig davon, wann oder wo das Projekt verwendet wird.

Dies ist besonders wichtig für stabile Builds, Tests und Deployment in Produktionssystemen.

Die Pipfile.lock speichert zusätzlich Hashwerte für jede Paketversion. Diese Hashes dienen als Sicherheitsmaßnahme, um sicherzustellen, dass nur verifizierte und unveränderte Pakete installiert werden.

Es ist darauf zu achten, dass das Pipfile.lock in der Versionskontrolle landet (z.B. git, svc, ..)

Lockfile updaten

Aktualisiere die Pipfile.lock regelmäßig, um sicherzustellen, dass dein Projekt mit den neuesten (und sichersten) Versionen kompatibel ist:

pipenv update <package>

oder für das gesamte Lockfile

pipenv lock

Pipfile.lock sync

Um exakt die Versionen zu installieren, die in der Pipfile.lock festgelegt sind, verwenden wir sync in einem Pipenv - Verzeichnis

pipenv sync

pipenv sync ignoriert die Angaben im Pipfile und verwendet nur die Informationen aus der Pipfile.lock. Das Pipfile.lock wird dabei nicht verändert.

Den pipenv Cache löschen

Es kann sinnvoll sein, bei Auflösungsschwierigkeiten den Resolve Cache zu löschen.

pipenv lock --clear

Abhängigkeiten löschen

Um eine Abhängigkeit zu löschen und aus dem Lock-File zu entfernen, gibt es den uninstall -Befehl. Er ändert das Pipfile und das Pipfile.lock entsprechend ab und entfernt die Abhängigkeit aus der Umgebung.

pipenv uninstall <paketname>

Dependency Graph

mit pipenv Graph lässt sich der Abhängigkeitsgraph visualisieren

pipenv graph

```
pandas==2.2.3
 ├──── numpy
 ├──── python-dateutil
 │     └──── six
 ├──── pytz
 └──── tzdata
scipy==1.15.0
 └──── numpy
```