07

Para saber mais: tipos de ambientes virtuais

A utilização de ambientes virtuais em projetos Python é uma prática padrão no desenvolvimento de software com a linguagem. O consenso da comunidade Python de que ambientes virtuais são uma ótima prática levou à criação de vários projetos com o objetivo de oferecer versões alternativas de ambientes virtuais e novas formas de gerenciá-los.

A seguir temos alguns exemplos de ambientes virtuais e ferramentas relacionadas mais utilizadas no mercado:

- **venv** (https://docs.python.org/pt-br/3/library/venv.html): É o ambiente virtual "padrão" do Python e sua grande vantagem é já vir instalado como um módulo na linguagem a partir da versão 3.3. Se trata de um *subset* (parte menor) da ferramenta **virtualenv**.

- **Virtualenv** (https://virtualenv.pypa.io/en/latest/): É uma ferramenta feita especificamente para a criação de ambientes virtuais e precede a criação da **venv**, sendo um *superset* (parte maior) dela. Algumas de sua principais vantagens sobre a **venv** são:
  - Maior velocidade, graças ao método `app-data` seed;
  - Pode criar ambientes virtuais para versões arbitrárias do Python instaladas na máquina;
  - Pode ser atualizado utilizando a ferramenta **pip**;
  - Possui uma *Programmatic API*, capaz de descrever um ambiente virtual sem criá-lo.

- **Conda** (https://docs.conda.io/en/latest/): É uma alternativa não apenas às ferramentas de ambiente virtuais já citadas, mas ao instalador de pacotes **pip** também. Possui um escopo mais centrado na área de ciência de dados e possui a capacidade de instalar pacotes fora do ecossistema do Python.

- **Virtualenvwrapper** (https://virtualenvwrapper.readthedocs.io/en/latest/): É uma extensão do projeto **Virtualenv** que torna a criação, deleção e gerenciamento geral dos ambientes virtuais mais fácil. Uma grande vantagem de sua utilização é a organização de todos os ambientes virtuais utilizados em um só lugar, além de facilitar os comandos de CLI.

- **Poetry** (https://python-poetry.org/): É uma ferramenta para gerenciamento de dependências e pacotes do Python. Através do Poetry é possível declarar quais pacotes um projeto necessita para funcionar, de forma parecida ao `requirements.txt`, porém, de forma determinística.