Менеджеры пакетов и виртуальных окружений для Python

Глеб Ерофеев, весна 2021

Пакеты и модули в Python

- Пакет в Python – это каталог, включающий в себя другие каталоги и модули, но при этом дополнительно содержащий файл `__init__.py` Пакеты используются для формирования пространства имен, что позволяет работать с модулями через указание уровня вложенности (через точку).
- Модуль - файл с расширением .py. Предназначены для того, чтобы в них хранить часто используемые функции, классы, константы и т.п. Можно условно разделить модули и программы: программы предназначены для непосредственного запуска, а модули для импортирования их в другие программы. Стоит заметить, что модули могут быть написаны не только на языке Python, но и на других языках (например C)

Зачем это все (виртуальные окружения)

- Изолированность - работе изолированного решения нельзя случайно помешать
- Воспроизводимость/Переносимость - на другом компьютере решение будет работать так же (без ощибок и выдавать те же результаты)
- Фиксирование версий библиотек
- Приложения vs библиотеки/пакеты

Виртуальное окружение - это изолированные настройки среды Python которые позволяют нам использовать определенные, нужные нам, библиотеки и их версии в нашем приложении. Эти настройки не зависят от настроек и ограничений системы на которой мы запускаем наше приложение.

Виртуальное окружение не фиксирует настройки системы и драйвера -

для этого есть другие инструменты - например Docker

Пример Сентябрь 2019: pandas.SparseDataFrame

Виды менеджеров пакетов и виртуальных окружений:

- Pip - самый распостраненный менеджер пакетов
- Virtualenv - основа, самое первое виртуальное окружение
- Conda - пакеты + окружение + скомпилированные образы
- Pip + Virtualenv = Pipenv (мы рекомендуем этот вариант)
- Poetry - Новая инкарнация pipenv (проще управлять зависимостями)

Жизненный цикл виртуального окружения на примере pipenv

- Установка (один раз)

```
pip install --user pipenv
```

- Активация виртуального окружения

```
cd my_project
pipenv shell # входим в наше окружение
exit # выход из окружения
```

- Установка пакетов внутри окружения

```
pipenv install pandas
```

- Фиксирование состояния (автоматически)

```
pipenv install pandas==1.1.1
>>> Installing pandas==1.1.1...
>>> Adding pandas to Pipfiles [packages]...
>>> Installation Succeeded
>>> Pipfile.lock not found, creating...
>>> Locking [dev-packages] dependencies...
>>> Locking [packages] dependencies...
>>> Locking...Building requirements...
>>> Resolving dependencies...

pipenv lock #Фиксирование
```

- Запуск проекта в виртуальном окружении

```
pipenv run python your_project.py
```

Фиксирование состояния pipenv

- Pipfile

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

[packages]
pandas = "==1.1.1"

[dev-packages]

[requires]
python_version = "3.7"
```

- Pipfile.lock

```
{
    "_meta": {
        "hash": {
            "sha256": "8d14434df45e0ef884d6c3f6e8048ba72335637a8631cc44792f52fd20b6f97a
        },
        "host-environment-markers": {
            "implementation_name": "cpython",
            "implementation_version": "3.6.1",
            "os_name": "posix",
            "platform_machine": "x86_64",
            "platform_python_implementation": "CPython",
            "platform_release": "16.7.0",
            "platform_system": "Darwin",
            "platform_version": "Darwin Kernel Version 16.7.0: Thu Jun 15 17:36:27 PDT
            "python_full_version": "3.6.1",
            "python_version": "3.6",
            "sys_platform": "darwin"
        },
        "pipfile-spec": 5,
        "requires": {},
        "sources": [
            {
```

Как устроено виртуальное окружение pipenv изнутри

```
pipenv graph # Просмотр зграфа зависимостей
pipenv --venv
```

Виртуальное окружение физически - это папка внутри проекта или в определенном месте, в которую устанавливаются все библиотеки окружения. Именно поэтому внешние изменения на машине хосте не ломают проект в виртуальном окружении и наоборот.

Резюме

- Фиксировать окружение - экономия сил и времени
- Фиксировать окружение - просто
- Мы будем пользоваться этим в наших проектах

Семинар Pipenv

- Ставим pipenv `pip install --user pipenv`
- Переходим в папку проекта `cd [my_project]`
- Фиксируем версию питона в проекте`pipenv --python 3.8.3`
- Добавляем/удаляем пакеты `pipenv install/uninstall <package>`
- Версии пакетов `pipenv install <package>~=1.2|>=1.2|<=1.2|<1.2`
- Запуск и оболочка `pipenv shell | exit`
- Фиксация зависимостей `pipenv lock`
- Создаем requirements.txt `pipenv lock -r`
- Добавляем пакеты из requirements.txt `pipenv install -r requirements.txt`
- Обновление зависимостей `pipenv update --outdated | pipenv update | pipenv update <package>`
- Анализ зависимостей `pipenv graph`

Домашнее задание

- Разобраться с poetry и выбрать себе любимую среду
- Создать окружение для своего проекта с паролями
- Разместить в обновленное решение в git
- Опубликовать в чате ссылку на репозиторий
- Посмотреть решения других участников - если понравилось ставим * в гите