Pipfile

Pipfile — новая альтернатива requirements.txt. Конечно, все привыкли к старому доброму pip, но надо признать, что ему не хватает некоторых возможностей, которые существуют, например, в npm, bundler или cargo. Файл Pipfile позволяет гораздо больше.

Так, сначала определяется источник, который будет использовать pip, затем необходимая версия Python, ваши и dev-пакеты.

Используя синтаксис TOML, можно точно указать версию источника или варианта операционной системы, которую вы предполагаете использовать. Обратите внимание, что все версии пакетов не указываются жестко.

Pip будет поддерживать эту спецификацию в будущем: pip install -p Pipfile. Но уже сегодня можно использовать новые возможности с помощью инструмента Pipenv. Он напоминает virtualenvwrapper или pyenv, но с функциями pip.

In [ ]:

[[source]]

url = 'https://pypi.python.org/simple'

verify_ssl = true

name = 'pypi'

[requires]

python_version = '2.7'

[packages]

requests = { extras = ['socks'] }

records = '>0.5.0'

django = { git = 'https://github.com/django/django.git', ref = '1.11.4', editable = true }

"e682b37" = {file = "https://github.com/divio/django-cms/archive/release/3.4.x.zip"}

"e1839a8" = {path = ".", editable = true}

pywinusb = { version = "*", os_name = "=='nt'", index="pypi"}

[dev-packages]

nose = '*'

unittest2 = {version = ">=1.0,<3.0", markers="python_version < '2.7.9' or (python_version >= '3.0' and python_version < '3.4')"}

Pyenv

Pyenv — это простой, мощный и кроссплатформенный инструмент для управления несколькими версиями Python в Linux-системах, он используется для:

переключения глобальной версии Python для каждого пользователя;

установки локальной версии Python для каждого проекта;

управления виртуальными средами, созданными anaconda или virtualenv;

переопределения версии Python с переменной окружения;

поиска команд из нескольких версий Python и для многого другого.

Как правило, версия Python по умолчанию используется для запуска всех ваших приложений, если вы явно не укажете версию, которую хотите использовать в приложении. Но pyenv реализует простую концепцию использования прокладок (легкие исполняемые файлы), чтобы передать вашу команду правильной версии Python, которую вы хотите использовать, когда у вас установлено несколько версий.

Они вставлены pyenv в каталоги перед вашим PATH. Поэтому, когда вы запускаете команду Python, она перехватывается соответствующей прокладкой и передается в pyenv, который затем задает версию Python, указанную вашим приложением, и передает ваши команды правильной версии Python.

Venv

Виртуальная среда (также называемая venv) — это среда Python, в которой интерпретатор, библиотеки и скрипты, установленные в ней, изолированы от других установленных виртуальных сред. По умолчанию любые библиотеки, установленные в «системный» Python являются частью операционной системы

Когда создаётся новая виртуальная среда, производится создание собственного локального интерпретатора с собственными библиотеками и сценариями. Поэтому, когда используется локальный интерпретатор, он загружает

библиотеки из локальной среды. Если он не находит, то он пытается выполнить поиск библиотеки в родительской/системной среде.

Virtualenv

Virtualenv — это инструмент для разделения зависимостей, необходимых для проектов. При работе над несколькими проектами часто возникает проблема, что разным проектам нужны разные версии одних и тех же пакетов, virtualenv помогает нам решать подобные проблемы. Это также решает проблему засорения системы ненужными пакетами, так как виртуальные окружения можно легко создавать и удалять.

Установка:

Virtualenv — это просто пакет, доступный в pypi, вы можете использовать pip для его установки.

pip install virtualenv

После установки вам может потребоваться добавить C:\Python39\Scripts в переменную среды PATH. Таким образом, такие команды, как pip и virtualenv можно будет выполнять из любой директории.

Создание виртуального окружения

Создайте новую директорию с именем python_project и измените текущую рабочую директорию на python_project:

mkdir python_project cd python_project

Чтобы создать виртуальное окружение внутри python_project, вам нужно выполнить следующую команду:

virtualenv my_env

Это создаст новую директорию my_env внутри python_project. Эта директория будет содержать копию интерпретатора python и копию исполняемого файла pip. Здесь мы использовали my_env в качестве имени, но вы можете использовать любое другое имя. Теперь ваше виртуальное окружение готово к использованию, вам просто нужно его активировать.

В этом руководстве есть один момент: мы установили virtualenv используя python 3.4. Предположим, у вас также есть python 2.7 и вы хотите создать

виртуальное окружение, используя python 2.7 вместо python 3.4, вы можете сделать это с помощью следующей команды:

```
virtualenv -p c:\Python27\python.exe my_env
```