包管理

为什么使用包管理

Python的模块或者源文件直接可以复制到目标项目目录中，就可以导入使用了。
但是为了更多项目调用使用，或者共享给别人，就需要打包，或发布到网络，以便供人使用。
目的也是为了复用。

Pypi（Python Package Index），公共的模块存储中心，https://pypi.python.org/pypi

主要工具

distutils

官方库distutils，使用安装脚本setup.py 来构建、安装包。
从1998年就是标准库的一部分，直到2000年停止开发。

setuptools

它是替代distutils的增强版工具集，包含easy_install工具，使用ez_setup.py文件。支持egg格式的构建和安装。
提供查询、下载、安装、构建、发布、管理等包管理功能。
setuptools是包管理的核心模块。

后来，setuptools开发缓慢了，出现基于setuptools的distribute来替代setuptools。2013年，这两个项目重新合并，distribute被废弃，setuptools依然是Python安装打包的标准方式。

pip

pip目前包管理的事实标准。
构建在setuptools之上，替代easy_install的。同样提供丰富的包管理功能。
Python3.4之前，需要单独安装，从Python3.4开始直接包含在安装文件中。

wheel

wheel格式定义在PEP427中。
wheel是zip打包的扩展名为.whl的文件，文件中不包含.pyc文件。

提供 bdist_wheel 作为 setuptools 的扩展命令，这个命令可以用来生成新打包格式 wheel。
pip 从1.4版本开始 提供了一个 wheel 子命令来安装 wheel 包。当然，需要先安装 wheel 模块。
它可以让Python库以二进制形式安装，而不需要在本地编译。

egg包正在被wheel包代替。Pypi网站越来越多的安装包新版本都采用了wheel包。使用Wheel包，直接找到对应OS平台、对应Python版本的包下载安装，也不需要本地编译。

使用setup.py打包

Python的.py文件就是模块，如果它只依赖Python标准库，就直接可以复制给别人使用，当然要注意CPython的版本。

如果代码较多，组织到了包中，而且只依赖Python标准库，就可以使用sdist打包为Source Distribute源代码包。

帮助文档/Distributing Python Modules/Reading the Python Packaging User Guide/Building and packaging the project

在线网址：https://packaging.python.org/tutorials/packaging-projects/#creating-setup-py

```
# 包结构
m
|-- __init__.py
|-- m1.py
|-- m2
    |-- __init__.py
    |-- m21
        |-- __init__.py
    |-- m22.py
```

项目根目录下，构建一个setup.py文件，setup.py如下

```
import setuptools

setuptools.setup(
    name="m", # Replace with your own username
    version="0.0.1",
    author="wayne",
    author_email="wayne@magedu.com",
    description="A small example package",
    url="http://www.magedu.com/python",
    #packages=setuptools.find_packages(), # 自动发现包
    packages=['m'],
    python_requires='>=3.6',
)

# name名字
# version 版本
# packages=[] 打包列表，
# packages=['m'],指定m，就会把m所有的非目录子模块打包
# ['m', 'm.m1.m2.m3'],逐级建立目录，但是只把m的所有非目录子模块打包，把m.m1.m2.m3打包
# ['m', 'm.m1', 'm.m1.m2', 'm.m1.m2.m3']
# description 描述信息
# author 作者
# author_email 作者邮件
# url 包的主页，可以不写
```

data_files 参考https://docs.python.org/3.8/distutils/setupscript.html#distutils-additional-files

查询命令的帮助

```
$ setup.py --help [cmd1 cmd2 ...]
$ python setup.py --help-commands
$ setup.py cmd --help
```

build命令，编译

创建一个build目录

```
$ python setup.py build
```

以下是packages=['m']配置的结果

```
running build
running build_py
creating build
creating build\lib
creating build\lib\m
copying m\m1.py -> build\lib\m
copying m\__init__.py -> build\lib\m
```

在项目目录下多了build目录，有一个lib子目录，lib下就是模块m的目录了。
m目录下的*.py文件被复制了，但是子目录没有被复制。

以下是packages=['m.m2.m21']配置的结果

```
running build
running build_py
creating build
creating build\lib
creating build\lib\m
creating build\lib\m\m2
creating build\lib\m\m2\m21
copying m\m2\m21\__init__.py -> build\lib\m\m2\m21
```

可以看出，逐级构建了同样的目录结构，并只拷贝了m21的`__init__.py`文件

以下是packages=['m', 'm.m2.m21']配置的结果

```
running build
running build_py
creating build\lib\m
creating build\lib\m\m2
creating build\lib\m\m2\m21
copying m\m2\m21\__init__.py -> build\lib\m\m2\m21
copying m\m1.py -> build\lib\m
copying m\__init__.py -> build\lib\m
```

build得到的文件，直接拷贝到其他项目就可以用

install命令，安装

build后就可以install，直接运行

```
$ python setup.py install
```

如果没有build，会先build编译，然后安装。

sdist命令

sdist命令

```
$ python setup.py sdist
```

创建源代码的分发包。
产生一个dist目录，里面生成一个带版本号的压缩包。
在其他地方解压缩这个文件，里面有setup.py，就可以使用 `$ python setup.py install` 安装了，
也可以 `$ pip install m-0.1.0.zip` 直接使用pip安装这个压缩包。

bdist命令

二进制分发包，或称作安装程序。它可以生成目标操作系统的安装程序。

```
制作windows下的安装包
$ python setup.py bdist_wininst
$ python setup.py bdist_msi
$ python setup.py bdist --format=msi

rpm包
$ python setup.py bdist_rpm
$ python setup.py bdist --format=rpm

压缩文件
$ python setup.py bdist --format=zip
$ python setup.py bdist --format=gztar
```

可以把自己写好的模块发布到公共的Pypi上，也可以搭建Pypi私服，供企业内部使用。

注意：Pypi里面的模块没有太好的审核机制，**不保证安全，请慎重使用。**

wheel包

安装wheel依赖。

```
$ pip install wheel
```

```
$ python setup.py bdist_egg
$ python setup.py bdist_wheel
```

更多打包，请参考https://packaging.python.org/overview/