Insper

17 - Pacotes Python

Desenvolvimento Aberto - 2018/2

Igor Montagner

Neste roteiro iremos criar nosso primeiro pacote Python.

Parte 1 - nosso pacote

Nosso módulo se chamará `my_hello` e disponibilizará um programa executável `hello.py` que importa nosso módulo e chama a função `my_hello.world()`.

```
pacote_exemplo/
    my_hello/
        __init__.py
        my_hello.py
    scripts/
        hello.py
    README.md
    LICENSE
```

Exercício: pesquise para que serve o arquivo `__init__.py`.

Exercício: crie os arquivos acima e preencha-os com o conteúdo correto.

Exercício: crie um projeto no github para esta atividade. Faça um primeiro commit nele com o conteúdo "zerado" do projeto.

Parte 2 - o arquivo `setup.py`

A descrição de um pacote Python é feita usando um arquivo setup.py Veja abaixo uma versão inicial deste arquivo:

```
from setuptools import setup

setup(name='my_hello_seunome',
      version='0.1',
      packages=['my_hello']
      )
```

Exercício: crie o arquivo acima no seu projeto, substituindo *seunome* por .... seu nome. Instale o seu próprio pacote usando

```
> pip install .
```

Exercício: em outra pasta, abra um console Python e tente importar seu módulo.

Exercício: pesquise quais argumentos são usados para especificar o autor do pacote, as versões de Python e sistemas operacionais suportados. Preencha estes valores com suas informações. O nome de seu pacote

Dependências

Para adicionar pacotes que são automaticamente instalados quando instalamos nosso pacote precisamos identificá-los no nosso arquivo *setup.py*. Para adicionar uma dependência de instalação basta adicionar o seguinte

argumento:

```
...
install_requires=[
    'pacote>=1.0',
    'pacote2'
],
...
```

Exercício: Adicione uma dependência ao nosso pacote.

requirements.txt

Muitos softwares usam também um arquivo *requirements.txt* para listar **todas** as dependências do software de modo a obter uma instalação idêntica à do desenvolvedor.

Exercício: crie um *requirements.txt* para seu projeto com as mesmas dependências listadas no seu *setup.py*.

Scripts executáveis

Além de instalar o nosso módulo para uso via `import` desejamos também disponibilizar o arquivo *hello.py* como um executável para todo o sistema. Isto pode ser feito adicionando a seguinte linha no nosso *setup.py* indicando que *scripts/hello.py* deverá ser instalado como um executável.

```
...
scripts=['scripts/hello.py'],
...
```

Não se esqueça de adicionar a seguinte linha no topo de seu arquivo para que ele possa ser executado diretamente do shell:

```
#!/usr/bin/env python3
```

Parte 3 - criando arquivos de distribuição

Dois tipos de arquivos de distribuição podem ser usados:

- sdist: é um arquivo contendo os fontes do projeto, incluindo arquivos adicionais especificados usando o argumento `data_files`. Usado se seu projeto for Python-puro.
- wheel: é um formato pré-compilado e específico para cada plataforma. Mais usado quando o projeto contém extensões em *C*.

A criação de um arquivo de distribuição de fontes é bem simples:

```
> python setup.py sdist
```

A instalação deste pacote pode ser feita via `pip`.

Parte 4 - envio para o PyPI

Vamos agora enviar nosso pacote para o *Python Package Index* para que ele possa ser instalado diretamente via `pip`. Para não poluir o repositório com pacotes temporários e de teste, podemos usar o *TestPyPI*. Toda sua infraestrutura é igual ao oficial, mas ele é limpo de maneira regular.

Visite https://test.pypi.org/account/register/ e registre-se no *TestPyPI*.

Após o registro, usaremos o pacote *twine* (instalável via *pip*) para fazer o upload:

```
> twine upload --repository-url https://test.pypi.org/legacy/ dist/*
```

Você poderá, então, instalar seu pacote a partir do test PyPI usando o seguinte comando:

```
> pip install --index-url https://test.pypi.org/simple/ my_hello_nome
```