

AULA PRÁTICA

Descomplicando a criação de pacotes em Python

Karina Kato

Machine Learning Engineer/Machine Learning Tech Lead - Take

Objetivos do Projeto

1. Entender conceitos relacionados aos pacotes
2. Atualizar o projeto e gerar as distribuições
3. Publicar o pacote

Requisitos Básicos

- ✓ Python instalado
- ✓ Ter um projeto a ser empacotado
- ✓ Git (recomendado)

Parte 1: Introdução e conceitos

Descomplicando a criação de pacotes em Python

Módulo vs Pacote

Módulo: objeto que serve como unidade organizacional do código que é carregado pelo comando de import.

Pacote: coleção de módulos com hierarquia.

Modularização

Vantagens da modularização:

- Legibilidade
- Manutenção
- Reaproveitamento de código

Pacote em Python

Vantagens de criar um pacote:

- Facilidade de compartilhamento
- Facilidade de instalação



Conceitos

- **Pypi:** repositório público oficial de pacotes
- **Wheel e Sdist:** dois tipos de distribuições
- **Setuptools:** pacote usado em setup.py para gerar as distribuições
- **Twine:** pacote usado para subir as distribuições no repositório Pypi



Passos

```
project_name/  
  README.md  
  setup.py  
  requirements.txt  
package_name/  
  __init__.py  
module1_name/  
  __init__.py  
  file1_name.py  
  file2_name.py  
module2_name/  
  __init__.py  
  file1_name.py  
  file2_name.py
```

1.Create Project

setup.py

wheel

sdist

2.Generate
Distributions

twine



3.Upload to Pypi

pip install package_name

Parte 2: Criar o projeto e gerar as distribuições

Descomplicando a criação de pacotes em Python



Exemplos de estruturas

```
project_name/  
  README.md  
  setup.py  
  requirements.txt  
  package_name/  
    __init__.py  
    file1_name.py  
    file2_name.py
```

Simples

```
project_name/  
  README.md  
  setup.py  
  requirements.txt  
  package_name/  
    __init__.py  
    module1_name/  
      __init__.py  
      file1_name.py  
      file2_name.py  
    module2_name/  
      __init__.py  
      file1_name.py  
      file2_name.py
```

Com vários módulos



Estrutura de pacote simples

```
project_name/  
  README.md  
  setup.py  
  requirements.txt  
  package_name/  
    __init__.py  
    file1_name.py  
    file2_name.py
```

Exemplos de chamadas a file1_name



```
import package_name.file1_name
```



```
from package_name import file1_name
```



Estrutura de pacote com vários módulos

```
project_name/  
  README.md  
  setup.py  
  requirements.txt  
  package_name/  
    __init__.py  
    module1_name/  
      __init__.py  
      file1_name.py  
      file2_name.py  
    module2_name/  
      __init__.py  
      file1_name.py  
      file2_name.py
```

Exemplos de chamadas a file1_name

```
import package_name.module1_name.file1_name
```

```
from package_name.module1_name import file1_name
```



DIGITAL
INNOVATION
ONE

Repositórios disponíveis

```
project_name/  
  README.md  
  setup.py  
  requirements.txt  
  package_name/  
    __init__.py  
    file1_name.py  
    file2_name.py
```



simple-package-template

```
project_name/  
  README.md  
  setup.py  
  requirements.txt  
  package_name/  
    __init__.py  
    module1_name/  
      __init__.py  
      file1_name.py  
      file2_name.py  
    module2_name/  
      __init__.py  
      file1_name.py  
      file2_name.py
```



package-template

<https://github.com/tiemi/>



Passos

```
project_name/  
  README.md  
  setup.py  
  requirements.txt  
package_name/  
  __init__.py  
module1_name/  
  __init__.py  
  file1_name.py  
  file2_name.py  
module2_name/  
  __init__.py  
  file1_name.py  
  file2_name.py
```

1.Create Project

setup.py

wheel

sdist

2.Generate
Distributions

twine



3.Upload to Pypi

pip install package_name



Passos para criar o projeto

```
project_name/  
  README.md  
  setup.py  
  requirements.txt  
  package_name/  
    __init__.py  
  module1_name/  
    __init__.py  
    file1_name.py  
    file2_name.py  
  module2_name/  
    __init__.py  
    file1_name.py  
    file2_name.py
```

1. Create Project

1. Fork do template
2. Adição do conteúdo dos módulos do projeto
3. Edição do arquivo setup.py
4. Edição do requirements.txt
5. Edição do README.md



Exemplo de pacote com vários módulos

```
project_name/  
  README.md  
  setup.py  
  requirements.txt  
  package_name/  
    __init__.py  
    file1_name.py  
    file2_name.py
```



simple-package-template

```
project_name/  
  README.md  
  setup.py  
  requirements.txt  
  package_name/  
    __init__.py  
    module1_name/  
      __init__.py  
      file1_name.py  
      file2_name.py  
    module2_name/  
      __init__.py  
      file1_name.py  
      file2_name.py
```



package-template



```
image-processing-package/  
  README.md  
  setup.py  
  requirements.txt  
  image_processing/  
    __init__.py  
    processing/  
      __init__.py  
      combination.py  
      transformation.py  
    utils/  
      __init__.py  
      io.py  
      plot.py
```

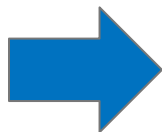


image-processing-package



Exemplo de pacote com vários módulos

```
project_name/  
  README.md  
  setup.py  
  requirements.txt  
  package_name/  
    __init__.py  
    module1_name/  
      __init__.py  
      file1_name.py  
      file2_name.py  
    module2_name/  
      __init__.py  
      file1_name.py  
      file2_name.py
```



```
image-processing-package/  
  README.md  
  setup.py  
  requirements.txt  
  image_processing/  
    __init__.py  
  processing/  
    __init__.py  
    combination.py  
    transformation.py  
  utils/  
    __init__.py  
    io.py  
    plot.py
```



DIGITAL
INNOVATION
ONE

Arquivos do projeto image-processing

```
image-processing-package/  
  README.md  
  setup.py  
  requirements.txt  
image_processing/  
  __init__.py  
  processing/  
    __init__.py  
    combination.py  
    transformation.py  
  utils/  
    __init__.py  
    io.py  
    plot.py
```

Image 1



Image 2



Result



Image 1



Image 2



Result





Arquivos do projeto image-processing

```
image-processing-package/  
  README.md  
  setup.py  
  requirements.txt  
image_processing/  
  __init__.py  
  processing/  
    __init__.py  
    combination.py  
    transformation.py  
  utils/  
    __init__.py  
    io.py  
    plot.py
```

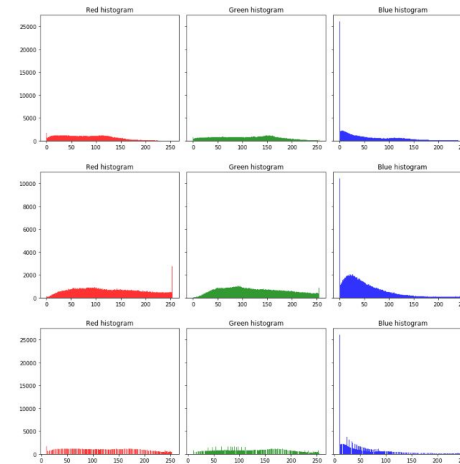
Image 1



Image 2



Result





Arquivos do projeto image-processing

```
image-processing-package/  
  README.md  
  setup.py  
  requirements.txt  
  image_processing/  
    __init__.py  
    processing/  
      __init__.py  
      combination.py  
      transformation.py  
  utils/  
    __init__.py  
    io.py  
    plot.py
```

Similarity of the images: 0.9500740181283867

Image 1



Image 2



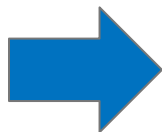
Result





Exemplo de pacote com vários módulos

```
project name/  
  README.md  
  setup.py  
  requirements.txt  
  package_name/  
    __init__.py  
    module1_name/  
      __init__.py  
      file1_name.py  
      file2_name.py  
    module2_name/  
      __init__.py  
      file1_name.py  
      file2_name.py
```



```
image-processing-package/  
  README.md  
  setup.py  
  requirements.txt  
  image_processing/  
    __init__.py  
  processing/  
    __init__.py  
    combination.py  
    transformation.py  
  utils/  
    __init__.py  
    io.py  
    plot.py
```



Arquivo setup.py

Usado para especificar como o pacote deve ser construído.

Documentação:

<https://setuptools.readthedocs.io/en/latest/setup-tools.html>

```
from setuptools import setup, find_packages

with open("README.md", "r") as f:
    page_description = f.read()

with open("requirements.txt") as f:
    requirements = f.read().splitlines()

setup(
    name="package_name",
    version="0.0.1",
    author="my_name",
    author_email="my_email",
    description="My short description",
    long_description=page_description,
    long_description_content_type="text/markdown",
    url="my_github_repository_project_link",
    packages=find_packages(),
    install_requires=requirements,
    python_requires='>=3.8',
)
```

Arquivo requirements.txt

Usado para passar as dependências que devem ser instaladas com o seu pacote.
Opcionalmente, podem ser especificadas as versões.

Arquivo README.md

Será exibido como documentação na página do Pypi do seu pacote. Foi usado markdown.

package_name

Description. The package package_name is used to: - -

Installation

Use the package manager [pip](#) to install package_name

```
pip install package_name
```

Usage

```
from package_name.module1_name import file1_name  
file1_name.my_function()
```

Author

My_name

License

[MIT](#)



Passos

```
project_name/  
  README.md  
  setup.py  
  requirements.txt  
package_name/  
  __init__.py  
module1_name/  
  __init__.py  
  file1_name.py  
  file2_name.py  
module2_name/  
  __init__.py  
  file1_name.py  
  file2_name.py
```

1.Create Project

setup.py

wheel

sdist

2.Generate
Distributions

twine



3.Upload to Pypi

pip install package_name

Distribuições

Para subir o pacote, criar uma distribuição binária ou distribuição de código fonte.

As versões mais recentes do pip instalam primeiramente a binária e usam a distribuição de código fonte, apenas se necessário.

De qualquer forma, iremos criar ambas distribuições.



Passos para gerar as distribuições



2. Generate
Distributions

1. Acessar a raiz do projeto
2. Comandos de instalação
3. Comando para criar a distribuição



DIGITAL
INNOVATION
ONE

Comandos de instalação

```
python -m pip install --upgrade pip
```

```
python -m pip install --user twine
```

```
python -m pip install --user setuptools
```



DIGITAL
INNOVATION
ONE

Comandos para criar distribuições

```
python setup.py sdist bdist_wheel
```



DIGITAL
INNOVATION
ONE

Parte 3: Publicando o pacote

Descomplicando a criação de pacotes em Python



Passos

```
project_name/  
  README.md  
  setup.py  
  requirements.txt  
package_name/  
  __init__.py  
module1_name/  
  __init__.py  
  file1_name.py  
  file2_name.py  
module2_name/  
  __init__.py  
  file1_name.py  
  file2_name.py
```

1.Create Project

setup.py

wheel

sdist

2.Generate
Distributions

twine



3.Upload to Pypi

pip install package_name



Passos para subir o pacote



3.Upload to Pypi

1. Criar conta no Test Pypi
2. Publicar no Test Pypi
3. Instalar pacote usando Test Pypi
4. Testar pacote
5. Criar conta no Pypi
6. Publicar no Pypi
7. Instalar pacote usando Pypi



Passos para subir o pacote



3.Upload to Pypi

1. Criar conta no Test Pypi
2. Publicar no Test Pypi
3. Instalar pacote usando Test Pypi
4. Testar pacote
5. Criar conta no Pypi
6. Publicar no Pypi
7. Instalar pacote usando Pypi



DIGITAL
INNOVATION
ONE

Criando contas no Pypi

<https://pypi.org/account/register/>

<https://test.pypi.org/account/register/>



DIGITAL
INNOVATION
ONE

Comando para publicar no Test Pypi

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

Comando para instalar o pacote de teste

```
pip install --index-url https://test.pypi.org/simple/  
image-processing
```



DIGITAL
INNOVATION
ONE

Comando para publicar no Pypi

```
python -m twine upload --repository-url  
https://upload.pypi.org/legacy/ dist/*
```

Comando para instalar o pacote

```
python -m pip install package_name
```



Resumo

```
project_name/  
  README.md  
  setup.py  
  requirements.txt  
package_name/  
  __init__.py  
module1_name/  
  __init__.py  
  file1_name.py  
  file2_name.py  
module2_name/  
  __init__.py  
  file1_name.py  
  file2_name.py
```

1.Create Project

setup.py

wheel

sdist

2.Generate
Distributions

twine



3.Upload to Pypi

pip install package_name



Resumo

```
project_name/  
  README.md  
  setup.py  
  requirements.txt  
  package_name/  
    __init__.py  
  module1_name/  
    __init__.py  
    file1_name.py  
    file2_name.py  
  module2_name/  
    __init__.py  
    file1_name.py  
    file2_name.py
```

1. Create Project

1. Fork do template
2. Adição do conteúdo dos módulos do projeto
3. Edição do arquivo setup.py
4. Edição do requirements.txt
5. Edição do README.md



Resumo



2. Generate
Distributions

1. Acessar a raiz do projeto
2. Comandos de instalação
3. Comando para criar a distribuição



Resumo




3.Upload to Pypi

1. Criar conta no Test Pypi
2. Publicar no Test Pypi
3. Instalar pacote usando Test Pypi
4. Testar pacote
5. Criar conta no Pypi
6. Publicar no Pypi
7. Instalar pacote usando Pypi

Exercício prático

Fazer um pacote usando a estrutura simples de um módulo para testar os conhecimentos adquiridos.



```
project_name/  
  README.md  
  setup.py  
  requirements.txt  
  package_name/  
    __init__.py  
    file1_name.py  
    file2_name.py
```

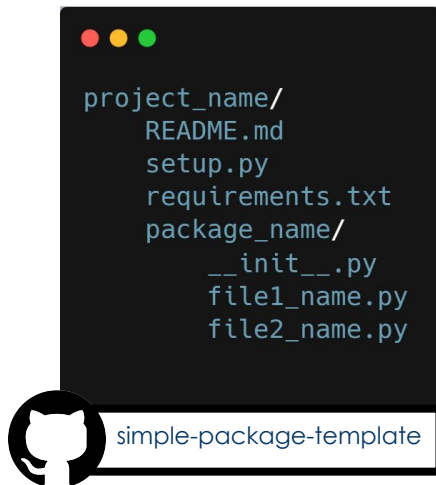


simple-package-template



Exercício prático

Fazer um pacote usando a estrutura simples de um módulo para testar os conhecimentos adquiridos.



Adicionais:

- Documentação do setuptools:
<https://setuptools.readthedocs.io/en/latest/setuptools.html>
- Testes automatizados:
<https://docs.pytest.org/en/latest/goodpractices.html>
- Uso do Tox:
<https://tox.readthedocs.io/en/latest/>



DIGITAL
INNOVATION
ONE

Repositórios disponíveis

```
project_name/  
  README.md  
  setup.py  
  requirements.txt  
  package_name/  
    __init__.py  
    file1_name.py  
    file2_name.py
```



simple-package-template

```
project_name/  
  README.md  
  setup.py  
  requirements.txt  
  package_name/  
    __init__.py  
    module1_name/  
      __init__.py  
      file1_name.py  
      file2_name.py  
    module2_name/  
      __init__.py  
      file1_name.py  
      file2_name.py
```



package-template



```
image-processing-package/  
  README.md  
  setup.py  
  requirements.txt  
  image_processing/  
    __init__.py  
    processing/  
      __init__.py  
      combination.py  
      transformation.py  
    utils/  
      __init__.py  
      io.py  
      plot.py
```



image-processing-package

<https://github.com/tiemi/>



DIGITAL
INNOVATION
ONE

Obrigada!



Karina Kato



[https://www.linkedin.com/in/
karina-kato-4b2a56182/](https://www.linkedin.com/in/karina-kato-4b2a56182/)

Dúvidas?

Descomplicando a criação de pacotes em Python