Programação Orientada a Objetos – Aula 06

Prof. João Luiz
joao.laoliveira@gmail.com

Importações de Arquivos e Módulos

Importando de um .py na Mesma Pasta

```
meu_projeto/
├── meu_programa.py
├── calculadora.py
```

calculadora.py

```python
def soma(a: float, b: float) -> float:
    return a+b
```

meu_programa.py

```python
from calculadora import soma

print(soma(10, 20))
```

Exemplo (colab)

Importando de uma Subpasta

```
meu_projeto/
├─ funcoes/
|   ├─ calculadora.py
|   ├─ estatistica.py
├─ meu_programa.py
```

meu_programa.py

```python
from funcoes.calculadora import soma
from funcoes import estatistica

print(soma(10, 20))
print(estatistica.media([5, 6, 7, 8]))
```

Exemplo (colab)

Uso do __init__.py

É utilizado para indicar que um diretório deve ser tratado como um **pacote Python**, permitindo que ele seja importado. Sua presença era obrigatória, mas **a partir do Python 3.3 não é mais necessária**. Ainda assim, o uso do __init__.py continua sendo uma boa prática para controlar e organizar a inicialização do pacote.

Funções principais do __init__.py:

- **Marcar um pacote (retrocompatibilidade):** em projetos antigos ou para manter clareza, ainda é comum incluir esse arquivo.
- **Inicialização:** pode conter código que precisa ser executado automaticamente quando o pacote é importado.
- **Organização de imports:** opcionalmente, pode importar classes, funções ou variáveis de outros módulos internos para simplificar o uso do pacote, mas não é obrigatório. Mesmo que algo não seja listado no __init__.py, ele continua acessível diretamente pelo caminho do módulo.

Uso do __init__.py

```
meu_projeto/
├─ funcoes/
|   ├─ __init__.py
|   ├─ calculadora.py
|   ├─ estatistica.py
├─ meu_programa.py
```

__init__.py

```python
# Importa o arquivo inteiro como submódulo
from . import estatistica
# Importa uma função de um arquivo
from .calculadora import soma
```

meu_programa.py

```python
from funcoes import soma
from funcoes import estatistica
```

Exemplo (colab)

Import Absoluto e Import Relativo

- **Import absoluto:** Utilizado para importar módulos de pacotes. É indicado o caminho absoluto do módulo (sem “.” no início)

```
from funcoes import soma
from funcoes import estatistica
```

- **Import relativo:** Utilizado somente dentro de pacotes para importar outros módulos do mesmo pacote. É indicado um caminho relativo do módulo (indicado com o “.” no início).

```
# Importa o arquivo inteiro como submódulo
from . import estatistica
# Importa uma função de um arquivo
from .calculadora import soma
```

O . indica que o estamos procurando um arquivo na mesma pasta ou alguma função/classe/variável no __init__.py

Exemplo (colab)

Criação e uso do setup.py

- O **setup.py** é um script em Python utilizado para configurar a distribuição de pacotes.

- Ele contém metadados sobre o pacote e instruções para empacotar e instalar o projeto. Esse arquivo é executado pelo utilitário de empacotamento **setuptools**.

Principais Componentes

- **name**: Nome do pacote
- **version**: Versão do pacote
- **description**: Descrição curta
- **install_requires:** Dependências necessárias
- **packages**: Lista dos pacotes Python a serem incluídos (pode ser listado manualmente ou automaticamente com a função **setuptools.find_packages()**).

Exemplo de setup.py

```
from setuptools import setup, find_packages

setup(
    name='meu_pacote',
    version='0.1',
    description='Um pacote exemplo',
    install_requires=['numpy'],
    packages=find_packages()
)
```

Instalando um pacote python com setup.py

Um pacote python contendo um arquivo setup.py pode ser instalado no environment python com o comando:

python setup.py install

Para executar tal instalação é preciso que os pacotes **setuptools** e **wheel** estejam disponíveis, para instalá-los execute:

python -m pip install --upgrade pip setuptools wheel

README.md

- O arquivo **README** de um projeto é um documento de texto que fornece informações essenciais sobre o projeto.

- Ele é geralmente o primeiro arquivo que um usuário ou colaborador vê ao visitar o repositório de código-fonte e serve como uma espécie de guia de introdução ao projeto

Componentes comum de um README

- **Título e Descrição Curta:** O nome do projeto e uma descrição de uma ou duas frases sobre o que o projeto faz.
- **Pré-requisitos:** Ferramentas ou conhecimentos necessários para usar ou contribuir para o projeto.
- **Instalação:** Instruções para instalar o projeto.
- **Uso:** Exemplos e código de amostra para mostrar como usar o projeto ou biblioteca.
- **Documentação:** Links para documentação mais extensa, se disponível.
- **Contribuição:** Como os colaboradores externos podem contribuir para o projeto.
- **Testes:** Como executar testes no código.
- **Licença:** Tipo de licença que o projeto usa (por exemplo, MIT, GPL, etc.)
- **Contato:** Informações de contato dos mantenedores ou do time do projeto.
- **Agradecimentos:** Opcionalmente, uma seção para agradecer aos colaboradores, usuários ou qualquer outra ajuda recebida.

Licenças

Uma licença em um projeto de software define as regras sob as quais o código pode ser usado, modificado e distribuído.

Ao escolher uma licença, o autor estabelece os termos legais para a utilização do seu projeto.

Licenças são especificadas no arquivo **LICENSE** ou **LICENSE.txt.**

Por que usar uma licença?

1. **Legalidade**: Específica os termos legais sob os quais outras pessoas podem usar seu projeto.
2. **Proteção**: Alguns tipos de licença protegem o autor contra responsabilidades relacionadas ao software.
3. **Clareza**: Deixa claro o que outros desenvolvedores podem ou não fazer com o código.
4. **Abertura**: Algumas licenças garantem que modificações do código também sejam abertas.

Tipos Comuns de Licenças

- **MIT**: Permissiva; permite reutilização, modificação e distribuição, até mesmo em projetos proprietários
  - https://opensource.org/license/mit/
- **GPL**: Exige que modificações sejam também abertas; não permite integração em projetos proprietários sem compatibilidade de licença
  - https://opensource.org/license/gpl-3-0/
- **Apache**: Similar ao MIT, mas também fornece uma concessão expressa de direitos de patente aos usuários.
  - https://opensource.org/license/apache-2-0/
- **The Unlicense**: Licença de software mais permissiva, todos os direitos autorais são renunciados e o código é colocado em domínio público.
  - https://opensource.org/license/unlicense/

Requerimentos

Os requerimentos do seu pacote devem ser listados (um por linha) no arquivo **requirements.txt** e no parâmetro **install_requires** (na forma de lista) do **setup.py**.

Estrutura básica de um pacote/projeto python

```
meu_projeto/
├─ meu_modulo/
│   ├─ __init__.py
│   ├─ meu_submodulo_1.py
│   ├─ meu_submodulo_2.py
├─ exemplos.py
├─ LICENSE
├─ README.md
├─ requirements.txt
├─ setup.py
```

O seu pacote é somente os arquivos contidos na pasta **meu_modulo**, os demais arquivos são somente auxiliares para instalação/documentação

Obrigado!

Prof. João Luiz

joao.laoliveira@gmail.com