

# Gerenciamento de pacotes e Boas práticas

**Guilherme Arthur de Carvalho**

Analista de sistemas

**<https://linktr.ee/decarvalhogui>**

# Objetivo Geral

Aprender a trabalhar com gerenciamento de pacotes em Python, e boas práticas de codificação seguindo as convenções da PEP 8.

# Pré-requisitos

✓ Python e VSCode

# Conteúdo

- ❑ Gerenciamento de pacotes
- ❑ Boas práticas em Python

# Gerenciamento de pacotes

Gerenciamento de pacotes e Boas práticas

# O que são pacotes em Python?

Pacotes são módulos que podem ser instalados e utilizados em seus programas Python. Eles permitem que você utilize código que foi escrito por outras pessoas, economizando tempo e esforço.

# O papel do pip

Pip é o gerenciador de pacotes do Python. Ele nos permite instalar, atualizar e remover pacotes facilmente. Ele se comunica com o PyPI (Python Package Index), que é onde a maioria dos pacotes Python são armazenados.

# Exemplo de código:



```
1 pip install numpy  
2 pip uninstall numpy  
3 pip list
```



# Uso de ambientes virtuais

Ambientes virtuais, como os criados por `venvs`, nos permitem manter as dependências de diferentes projetos. Isso é importante para evitar conflitos entre versões de pacotes.

# Exemplo de código:




```
1 python3 -m venv myenv  
2 source myenv/bin/activate
```

# Comandos do pip

Como um programador que está aprendendo Python e deseja gerenciar os pacotes do seu projeto, é importante conhecer alguns dos principais comandos do pip.

# Comandos do pip



```
1  # Instalar um pacote
2  pip install nome_do_pacote
3  # Desinstalar um pacote
4  pip uninstall nome_do_pacote
5  # Listar pacotes instalados
6  pip list
7  # Atualizar um pacote
8  pip install --upgrade nome_do_pacote
9  # Procurar por pacotes
10 pip search termo_de_busca
```

# Introdução ao pipenv

Pipenv é uma ferramenta de gerenciamento de pacotes que combina a gestão de dependências com a criação de ambiente virtual para seus projetos e adiciona/remove pacotes automaticamente do arquivo Pipfile conforme você instala e desinstala pacotes.

# Comandos do pipenv



```
1 pip install pipenv  
2 pipenv install numpy  
3 pipenv uninstall numpy  
4 pipenv lock
```

# Introdução ao poetry

Poetry é outra ferramenta de gerenciamento de dependências para Python que permite declarar as bibliotecas de que seu projeto depende e gerencia (instala/atualiza/remove) essas bibliotecas para você. Ela também suporta o empacotamento e a publicação de projetos no PyPI.

# Comandos do poetry



```
1 pip install poetry
2 poetry new myproject
3 cd myproject
4 poetry add numpy
5 poetry remove numpy
```



# Boas práticas em Python

Gerenciamento de pacotes e Boas práticas

# Introdução

Python tem uma série de convenções e melhores práticas codificadas em PEPs (Propostas de Melhoria do Python). A mais conhecida destas é provavelmente a PEP 8, que cobre o estilo de codificação.

# O que é PEP 8?

[PEP 8](#) é o guia de estilo para codificação em Python. Ele inclui convenções sobre nomes de variáveis, uso de espaços em branco, comprimento da linha e muitas outras coisas que ajudam a manter o código Python consistente e legível.

# Principais recomendações da PEP 8

Algumas das principais recomendações da PEP 8 incluem usar 4 espaços para a indentação, limitar as linhas a 79 caracteres, usar nomes de variáveis em snake\_case para funções e variáveis, e CamelCase para classes.


# Exemplo de código:

```
1 def somar(argumento_1, argumento_2):  
2     # Esta é uma função de exemplo seguindo a PEP 8  
3     pass  
4  
5  
6 class ContaBancaria:  
7     # Esta é uma classe de exemplo seguindo a PEP 8  
8     pass
```

# Uso de ferramentas de checagem de estilo

Para nos ajudar a seguir as recomendações da PEP 8, podemos usar ferramentas de checagem de estilo como flake8. Essas ferramentas verificam nosso código e nos informam onde estamos desviando do guia de estilo.

# Exemplo de código:



```
1 pip install flake8
2 flake8 meu_script.py
```

# Formatação automática de código

Black é uma ferramenta de formatação de código Python que segue a filosofia "formato único". Black reformata todo o seu arquivo em um estilo consistente, simplificando a tarefa de manter o código em conformidade com a PEP 8.



# Exemplo de código:



```
1 pip install black  
2 black meu_script.py
```

# Organização de imports com isort

Isort é uma ferramenta Python para classificar importações alfabeticamente e separá-las automaticamente em seções. Ele proporciona uma maneira rápida e fácil de ordenar e categorizar suas importações.

# Exemplo de código:



```
1 pip install isort  
2 isort meu_script.py
```

# Dúvidas?

> Fórum/Artigos - <https://web.dio.me/articles>

# Exemplo de código:

Insira sua imagem dentro deste espaço  
(retire o retângulo azul, ele deverá ser utilizado  
somente para referência)

