**🧼 Guia de Código Limpo em Python**

**📌 Objetivo**

Este documento tem como objetivo definir diretrizes para escrever **código limpo, legível, eficiente e sustentável** em projetos Python. Ele se baseia nos princípios do livro *Clean Code* (Robert C. Martin), nas *PEP 8* e outras práticas consagradas.

**1. 🧠 Nomes Claros e Descritivos**

**✅ Boas práticas:**

* Use nomes significativos para variáveis, funções e classes.
* Prefira nomes completos ao invés de abreviações obscuras.
* Use snake\_case para variáveis e funções, PascalCase para classes e Constantes.

**❌ Evite:**

def prc(d, p): return d \* p

**✅ Prefira:**

def calcular\_preco(desconto, preco): return desconto \* preco

**2. 🧱 Estrutura e Organização**

**✅ Regras principais:**

* Um arquivo Python (módulo) deve conter uma única responsabilidade sempre que possível.
* Separe código em funções e classes bem definidas.
* Limite o tamanho das funções (ideal: até 20–30 linhas).

**3. 🧪 Funções e Métodos**

**✅ Devem:**

* Ter nomes que expressem **ações** (ex: obter\_dados, enviar\_email).
* Ter poucos parâmetros (até 3 é o ideal).
* Fazer **uma única coisa**.

**4. 🧼 Comentários**

**✅ Quando usar:**

* Para explicar *"por que"* algo foi feito, e não *"o que"* (o código já deve ser claro).
* Para marcações especiais: # TODO, # FIXME, etc.

**❌ Evite:**

python

CopiarEditar

x = 10 # define x como 10

**5. 🔁 Estruturas de Controle**

* Use listas por compreensão sempre que possível:

python

CopiarEditar

# ✅

quadrados = [x\*\*2 for x in range(10)]

# ❌

quadrados = []

for x in range(10):

quadrados.append(x\*\*2)

**6. 🧪 Testes**

* Escreva testes unitários sempre que possível.
* Use pytest ou unittest.
* Nomeie os testes com clareza: test\_login\_com\_senha\_invalida.

**7. 🛑 Tratamento de Erros**

* Evite capturar exceções genéricas (except:).
* Use exceções específicas e logue o erro.

python

CopiarEditar

try:

resultado = divisao(a, b)

except ZeroDivisionError:

logger.error("Divisão por zero.")

**8. 📦 Imports**

* Mantenha organizados em três grupos:
  1. Bibliotecas padrão
  2. Bibliotecas de terceiros
  3. Módulos do projeto
* Use apenas o necessário (evite import \*).

**9. 🧪 Tipagem**

Use **type hints** para tornar o código mais legível e integrável com ferramentas:

python

CopiarEditar

def somar(a: int, b: int) -> int:

return a + b

**10. 🧹 Ferramentas Recomendadas**

* **Formatador:** black ou autopep8
* **Lint:** flake8, pylint
* **Type checker:** mypy
* **Gerenciador de dependências:** poetry ou pip-tools
* **Testes:** UnitTest

**🧾 Exemplo de Código Limpo**

python

CopiarEditar

from datetime import datetime

from typing import List

class Pedido:

def \_\_init\_\_(self, itens: List[str]):

self.itens = itens

self.data = datetime.now()

def exibir\_resumo(self) -> None:

print(f"Pedido realizado em {self.data}")

for item in self.itens:

print(f"- {item}")

def main():

pedido = Pedido(itens=["Bíblia", "Devocional", "Hinário"])

pedido.exibir\_resumo()

if \_\_name\_\_ == "\_\_main\_\_":

main()