PADRÕES DE NOMECLATURA

Quando estamos desenvolvendo em Python, seguir padrões de nomenclatura ajuda a manter o código organizado, legível e consistente. Python possui uma série de convenções amplamente aceitas, que são descritas no guia oficial chamado PEP 8 (Python Enhancement Proposal 8).

https://peps.python.org/pep-0008/

NOMES DE VARIÁVEIS E FUNÇÕES

- Snake_case: Utilize letras minúsculas com palavras separadas por sublinhado (_).
- Exemplo: total_cost, calculate_area, user_name.
- Isso torna os nomes legíveis e descritivos, facilitando o entendimento do que a variável ou função representa.

NOMES DE CLASSES

- CamelCase (ou PascalCase): Utilize a primeira letra de cada palavra em maiúsculo e não utilize sublinhados.
- Exemplo: CustomerAccount, EmployeeDetails, BankTransaction.
- Isso ajuda a distinguir rapidamente classes de funções ou variáveis.

CONSTANTES

- Letras Maiúsculas com Substituições por Subtítulos: Use todas as letras em maiúsculo e separe as palavras com sublinhados.
- Exemplo: MAX_SPEED, PI, DEFAULT_TIMEOUT.
- Constantes são valores que não devem mudar ao longo da execução do programa.

MÓDULOS E PACOTES

- Nomes em minúsculas: Use letras minúsculas e, quando necessário, separe palavras com sublinhados para os nomes de módulos e pacotes.

- Exemplo: my_module.py, data_processing.py.

- Isso se alinha com a prática comum de importar módulos sem colisão de nomes.

MÉTODOS PRIVADOS

- Underline Simples: Use um sublinhado (_) antes do nome para indicar que um método ou variável é pretendido como privado.

- Exemplo: _calculate_total, _data_cleanup.

- Essa é uma convenção que sinaliza aos outros desenvolvedores que esse método não deve ser usado fora da classe onde foi definido.

MÉTODOS E ATRIBUTOS FORTEMENTE PRIVADOS

- Double Underline (Dunder): Utilize dois sublinhados (__) antes do nome para evitar colisões de nome com subclasses.
- Exemplo: __calculate_interest, __init__.
- Isso faz com que o nome do método ou atributo seja “mangled” (modificado) internamente, dificultando o acesso a partir de subclasses.

MÉTODOS E ATRIBUTOS FORTEMENTE PRIVADOS

- Usar dois sublinhados (__) no início do nome do método ou atributo faz com que o Python aplique "name mangling" (obfuscação de nomes), tornando mais difícil acessar o método ou atributo fora da classe.

- O Python transforma o nome para incluir o nome da classe, dificultando o acesso acidental ou intencional ao método ou atributo de fora da classe.

MÉTODOS E ATRIBUTOS FORTEMENTE PRIVADOS

```python
class MyClass:
    def __init__(self, value):
        self.__value = value  # Atributo fortemente privado

    def __private_method(self):
        return self.__value ** 2  # Método fortemente privado

obj = MyClass(10)
# print(obj.__private_method())  # Isso gera um erro

# No entanto, ainda é possível acessar o método com name mangling
print(obj._MyClass__private_method())  # Isso funciona, mas é desencorajado
```

Nota: O "name mangling" torna o método ou atributo menos acessível, mas não completamente inacessível. Pode ser acessado com _ClassName__MethodName.

NOMES ESPECIAIS

- Dunder (Double Underline): Certos nomes têm significado especial no Python e usam dois sublinhados antes e depois do nome.
- Exemplo: __init__, __str__, __call__.
- Esses métodos e atributos são conhecidos como "métodos mágicos" e têm significados específicos para o funcionamento de classes e objetos.

COMPRIMENTO DOS NOMES

- Descritivo, mas não excessivamente longo: Escolha nomes que sejam suficientemente descritivos para entender o propósito, mas não tão longos que se tornem difíceis de ler.

- Exemplo: calculate_circle_area é preferível a calc ou calculate_the_area_of_a_circle.

NOMES DE PARÂMETROS

- Use nomes claros: Os parâmetros de funções e métodos devem ser descritivos o suficiente para entender o que é esperado sem precisar de documentação adicional.

- Exemplo: calculate_tax(income, tax_rate) em vez de calculate_tax(x, y).

THANK YOU

Wagner Coutinho

https://www.linkedin.com/in/wagner-coutinho-mf/

wagner.filho@rarolabs.com.br