Antes de qualquer informação é importe ler o documento oficial.

Documento oficial PEP8: https://peps.python.org/pep-0008/#method-names-and-instance-variables

Em Python, não existem, de fato, atributos **estritamente privados** como em outras linguagens de programação, como Java ou C++. O que existe é uma convenção para indicar que um atributo deve ser tratado como privado.

A convenção se dá pelo uso de um único underscore _ antes do nome do atributo.

Diferenças:

- **Atributos "regulares":** Podem ser acessados e modificados diretamente de qualquer lugar do código.
- **Atributos com _ (underscore):** Sinalizam para outros programadores que esse atributo não deve ser acessado ou modificado diretamente fora da classe. Isso serve como um aviso de que a manipulação direta pode gerar comportamentos inesperados no código.

Exemplo:

```
class Exemplo:
  def __init__(self):
    self.publico = "Sou um atributo público"
    self._privado = "Sou um atributo 'privado'"

instancia = Exemplo()
print(instancia.publico)  # Acesso permitido
print(instancia._privado)  # Acesso também permitido (apesar da convenção)
```

Por que usar atributos "privados" se o acesso ainda é possível?

1. **Organização e legibilidade:** Facilita a leitura do código e a identificação de quais atributos devem ser manipulados internamente pela classe.
2. **Prevenir alterações acidentais:** Diminui o risco de modificações indevidas em partes críticas da classe.

Por que Python não possui atributos estritamente privados?

No meu entendimento é por uma questão de transição para conquistar novos programadores, criando uma espécie de liberdade.

recomendo ler:

https://peps.python.org/pep-0008/#method-names-and-instance-variables

https://www.geeksforgeeks.org/name-mangling-in-python/

https://peps.python.org/pep-0008/#designing-for-inheritance

https://www.programiz.com/python-programming/object-oriented-programming#encapsulation