Por: Alice Dantas

Métodos Mágicos

Métodos mágicos são funções especiais que começam e terminam com dois **underlines** (__) e servem para **personalizar o comportamento dos objetos** em situações específicas.

Eles fazem com que **objetos se comportem como tipos nativos do Python**.

Alguns exemplos comuns:

1. `__init__(self, ...)` — Inicializador do objeto

É chamado automaticamente quando você cria uma instância da classe, e é onde você inicializa os atributos do objeto.

***** Atributos definidos **no __init__** são **de instância** (pertencem a cada objeto). Já os atributos definidos **fora do __init__** são **de classe** (compartilhados por todos os objetos).

```
class Pessoa:
    def __init__(self, nome, idade):
        self.nome = nome
        self.idade = idade

p1 = Pessoa("Ana", 25)
print(p1.nome)  # Ana
print(p1.idade)  # 25
```

2. `__str__(self)` — Representação em String

Define o que será exibido ao usar `print(objeto)`. Sem ele, o Python mostra o endereço de memória. Esse método deixa a visualização dos objetos mais amigável.

```
class Pessoa:
    def __init__(self, nome, idade):
        self.nome = nome
        self.idade = idade

    def __str__(self):
        return f"{self.nome}, {self.idade} anos"

p1 = Pessoa("Ana", 25)
print(p1)  # Ana, 25 anos
```

3. `__enter__()` e `__exit__()` – Gerenciador de Contexto

Permitem que sua classe seja usada como um **gerenciador de contexto** (ou seja, com o `with`). O `with` garante que alguns códigos sejam **executados antes e depois de um bloco de código**, como quando você abre e fecha arquivos. O `__enter__` executa antes de entrar no bloco do with e pode retornar algum valor. O `__exit__` executa ao sair do bloco do with, seja com sucesso ou erro.

```
class ConexaoBanco:
    def __enter__(self):
        print("Abrindo conexão com o banco...")
        return self

    def __exit__(self, tipo, valor, traceback):
        print("Fechando a conexão com o banco...")


with ConexaoBanco() as banco:
    print("Conectado ao banco.")
```

 # criando uma instância da classe ConexaoBanco, ou seja, está **chamando o __init__** e o que o método **__enter__()** retornar será **atribuído à variável banco**

```
#Saída: Abrindo conexão com o banco…
        Conectado ao banco.
        Fechando a conexão com o banco...
```

4. `__call__(self, ...)` – Tornando o Objeto "Chamável"

Permite que o **objeto se comporte como uma função**. Ao colocar `()` após o objeto, o método `__call__` é executado.

```
class Somador:
    def __init__(self, valor):
        self.valor = valor

    def __call__(self, outro_valor):
        return self.valor + outro_valor

soma = Somador(10)
print(soma(5))  # 15
```

Por: Alice Dantas

⚠️ Não confunda `__init__` e `__enter__`

	`__init__`	`__enter__`
Quando é chamado	Quando o **objeto é criado** (instanciado)	Quando o objeto é usado com with
Para que serve	Para **inicializar os atributos** do objeto	Para **executar ações ao entrar no bloco with**
É obrigatório?	Não, mas mesmo que você **não o defina na sua classe**, o Python **usa automaticamente um `__init__` padrão**	Só usamos se quisermos permitir with
Retorna algo?	Não retorna nada (o objeto já está criado)	Retorna o valor que será atribuído no as