obrunojeronimo e python_direto_ao_ponto

Você conhece as estruturas básicas do Python?

Bora entender um pouco sobre os elementos que compõem a linguagem.

#python

obrunojeronimo e python_direto_ao_ponto

O que seriam essas estruturas básicas?

Se você está começando a aprender Python é essencial dominar primeiro as estruturas fundamentais da linguagem.

São como as peças de LEGO que usamos para construir scripts simples, sistemas complexos, APIs ou modelos de inteligência artificial.

Antes de entender como os objetos se comportam e interagem, é preciso compreender como Python organiza seus dados, como estruturar seu código e como funções e métodos executam tarefas.

#python

A primeira estrutura é a variável

Uma variável é como se fosse uma “caixa” que armazena informações na memória do computador. Você atribui um valor a um nome, e depois pode usar esse nome sempre que precisar do valor.

```
mensagem = "Olá, mundo!"
idade = 25
```

Aqui, mensagem e idade são variáveis que armazenam uma string e um número, respectivamente. Você pode pensar nelas como rótulos para acessar dados de forma fácil e clara.

#python

Agora, vamos para função

Uma função é um bloco de código que você pode usar sempre que precisar para realizar uma tarefa específica. Ela ajuda a organizar o código, evitar repetições e deixar mais fácil de entender.

Como funciona:

1. Defina a função usando **def**
2. Ela pode ter a inserção de parâmetros (entradas)
3. Pode ou não retornar um valor com return.

```
def somar(a, b):
    return a + b
```

Exemplo:
Você pode usar duas variáveis como argumentos e a função retorna com a soma.

```
a = 10
b = 5

print(somar(a, b))
```

✓ 0.0s

```
15
```

#python

Falando em argumentos....

Nessa parte, temos duas estruturas para detalhar: **parâmetro e argumento.** Quando se pensa em parâmetro, é o que a função espera receber. Já o argumento é o valor que você envia ao chamar a função.

Vamos ver no exemplo abaixo:

```
def saudacao(nome):
    print(f"Olá, {nome}!")

saudacao("Bruno")
```

parâmetro

argumento

Parâmetros tornam a função flexível e reutilizável com diferentes entradas.

#python

Você já ouviu falar sobre método?

Um método é uma função que vive dentro de uma classe. Ela pode usar e modificar os dados do próprio objeto, e é por isso que sempre recebe o self, que é a mesma coisa de dizer: “essa função pertence a mim, o objeto que está chamando”.

Olha o exemplo abaixo:

```
class Pessoa:
    def falar(self):
        print("Olá, tudo bem?")
```

método

Ao criar um objeto dessa classe, você pode chamar o método com obj.falar()

#python

Agora, vem a classe

Uma classe é como um molde para criar objetos. Ela diz quais dados (atributos) e comportamentos (métodos) cada objeto vai ter.

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

    def apresentar(self):
        print(f"Meu nome é {self.nome}")
```

classe

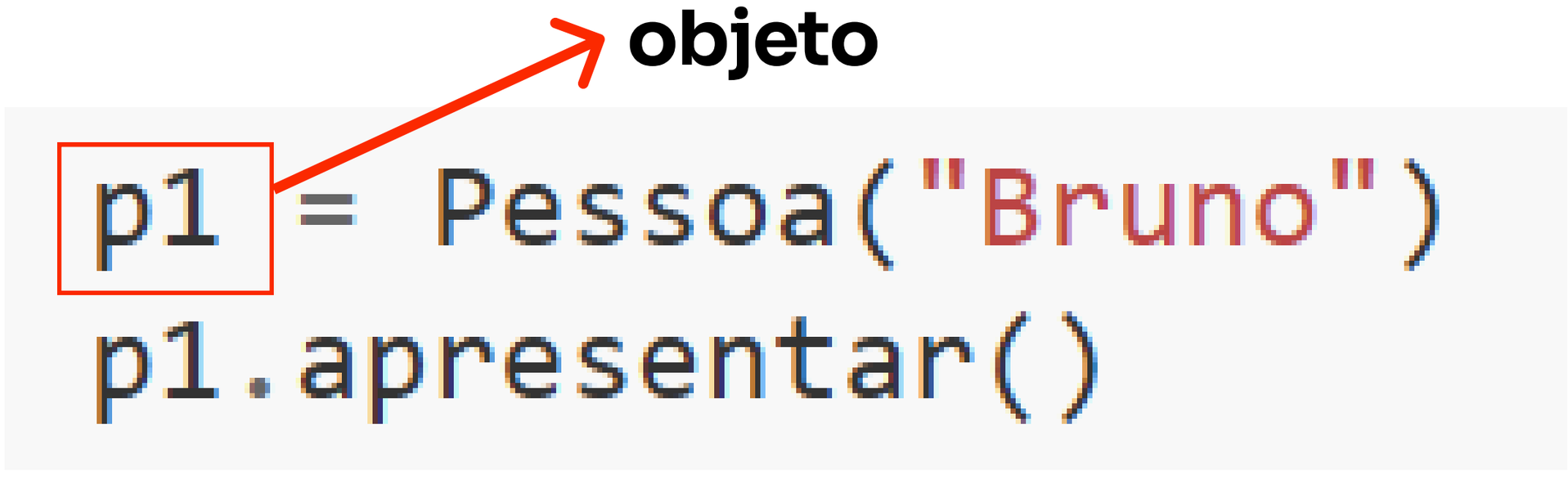
Com essa classe, você pode criar várias pessoas com nomes diferentes e métodos próprios.

#python

Finalmente, chegamos no objeto

Um objeto é algo criado a partir de uma classe. Ele carrega os dados e as ações definidos no molde da classe.

Bora pra mais um exemplo:



Nesse exemplo, p1 é um objeto da classe Pessoa. Ele tem seu próprio nome e pode executar métodos como apresentar().

#python

E pra finalizar o post, vamos falar de atributos.

Um atributo é uma informação guardada dentro do objeto. Ele é definido na classe e acessado com self.nome_do_atributo.

Exemplo:

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

p1 = Pessoa("Bruno")
print(p1.nome)
```

✓ 0.0s

atributo

```
Bruno
```

O valor "Bruno" é passado como argumento para a função. O Python guarda esse valor no atributo self.nome do objeto.

#python

obrunojeronimo e python_direto_ao_ponto

Se esse post tiver sido útil para você

Curte, Comenta & Compartilha.

Referências:

Python Direto ao Ponto, Estevão Paiva Fonseca, 2024.

#python

Para mais conteúdos, me segue no Instagram e LinkedIn