09 { ..

Orientação a objetos

} ..

Formador: Ricardo Mourão

Orientação a objetos

{

O que são objetos?

Formador: Ricardo Mourão

Orientação a objetos

{

Quais destes são objetos?

Formador: Ricardo Mourão

Orientação a objetos

O que são objetos?

Coisa material ou abstrata que pode ser percebida pelos sentidos e descrita de acordo com as suas características, comportamentos e estado atual.

Formador: Ricardo Mourão

Orientação a objetos

{ CANETA

Formador: Ricardo Mourão

Orientação a objetos

{ CANETA

Classe

Objeto

Formador: Ricardo Mourão

Orientação a objetos

{ CANETA

Atributos

- Modelo
- Cor
- Bico
- Carga
- Tampa

Métodos

- Escrevo
- Assino
- Pinto
- Rabisco
- Tapo/Destapo

Estado

- Com a tampa e carga cheia

Formador: Ricardo Mourão

Orientação a objetos

Classe Caneta

Modelo: BIC Cristal
Cor: Vermelha
Bico: 0.8
Carga: 70
Tampa: false

```
escrever(msg):
    if tampa == True and carga > 10:
        print(msg)

assinar(nome):
    if tampa == True and carga > 10:
        print(nome)

tampa():
    if tampa == True:
        tampa == False
    elif tampa == False:
        tampa == True
```

Formador: Ricardo Mourão

Orientação a objetos

Instanciar

Formador: Ricardo Mourão

Orientação a objetos

O que é Orientação a Objetos?

- ✓ A Orientação a Objetos (POO) é um paradigma de programação que usa 'objetos' para modelar dados e comportamentos.
- ✓ Neste paradigma, a programação é organizada em torno de objetos, não de ações e dados, não de lógica e sim em torno de como os objetos se ligam entre si.
- ✓ Um objeto é uma instância de uma classe, que é como um modelo ou um plano.
- ✓ Cada objeto pode conter dados (chamados de atributos ou propriedades) e código (chamados de métodos ou funções).

Formador: Ricardo Mourão

Orientação a objetos

Porquê Orientação a Objetos?

- ✓ **Modularidade:**

  A POO permite que os desenvolvedores construam módulos que não dependem uns dos outros.

  Isso facilita a manutenção e atualização do código, pois as alterações em um módulo geralmente não afetam outros.

- ✓ **Reutilização de Código:**

  Com a POO, é possível reutilizar classes existentes em novos programas.

  Isso economiza tempo e esforço, pois os desenvolvedores podem usar e modificar código já testado e comprovado em vez de escrever tudo do zero.

Formador: Ricardo Mourão

Orientação a objetos

Porquê Orientação a Objetos?

- ✓ **Encapsulamento:**

  Encapsulamento significa que os detalhes internos do funcionamento de uma classe são escondidos do mundo externo.

  Isso torna a interface com o software mais simples e reduz o impacto das mudanças, pois as alterações internas não afetam outras partes do programa.

- ✓ **Abstração:**

  Utilizando classes, os desenvolvedores podem criar objetos mais abstratos no topo da hierarquia e objetos mais concretos e especializados em níveis inferiores.

Formador: Ricardo Mourão

Orientação a objetos

Porquê Orientação a Objetos?

- ✓ **Herança:**

  Permite que uma nova classe herde propriedades e métodos de uma classe existente.

  Isso facilita a criação de novas abstrações, mantendo a consistência e reduzindo a repetição de código.

- ✓ **Polimorfismo:**

  O polimorfismo na POO permite que um método tenha muitas formas diferentes.

  Isso significa que uma função pode ser usada em diferentes contextos, dependendo do tipo de objeto que a invoca.

Formador: Ricardo Mourão

Orientação a objetos

```
class Contador:
    def __init__(self):
        self.valor = 0

    def incrementar(self):
        self.valor += 1

    def decrementar(self):
        self.valor -= 1

    def mostrar_valor(self):
        print(f"Valor atual do contador: {self.valor}")

meu_contador = Contador()
meu_contador.incrementar()
meu_contador.decrementar()
meu_contador.mostrar_valor()
```

Formador: Ricardo Mourão

Orientação a objetos

Criar Classes

- ✓ “__init__” serve como construtor de uma classe.
- ✓ Para a classe ser construída é necessário utilizar este método, tal e qual como uma função.
- ✓ **self**: O primeiro argumento de qualquer método é a referência a ele próprio.
- ✓ Em “__init__”, o self referencia o objeto criado recentemente, e em outros métodos, referencia a instância de qual o método foi invocado.

Formador: Ricardo Mourão

Orientação a objetos

{ Criar Classes

```
class Conta:
def __init__(self):
        self.valor = 0
```

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

Formador: Ricardo Mourão

Orientação a objetos

{ Criar Classes

```
class Conta:
def __init__(self):
    self.valor = 0
```

```
conta = Conta()
print(conta.valor)
```

Instância da Classe,
ou seja, um objeto

Formador: Ricardo Mourão

Orientação a objetos

{ Atributos

Atributos de instância

- ✓ Variáveis que pertencem a uma instância particular da classe
- ✓ Cada instância tem o seu próprio valor para o atributo

Atributos de classe

- ✓ Variáveis que pertencem à classe como um todo.
- ✓ Todas as instâncias da classe partilham o mesmo atributo (valor).

Formador: Ricardo Mourão

Orientação a objetos

Atributos

```
class Jogo:
def __init__(self, titulo, consola, preco):
        self.titulo = titulo
        self.consola = consola
        self.preco = preco
```

Formador: Ricardo Mourão

Orientação a objetos

Atributos

```
jogo = Jogo("Palworld", "PC", 29.90)

print(jogo.titulo)
print(jogo.consola)
print(jogo.preco)
```

Formador: Ricardo Mourão

Orientação a objetos

Atributos

```
class Jogo:
def __init__(self, titulo, consola, preco):
        self.titulo = titulo
        self.consola = consola
        self.preco = preco

jogo = Jogo("Palworld", "PC", 29.90)

print(jogo.titulo)
print(jogo.consola)
print(jogo.preco)
```

Formador: Ricardo Mourão

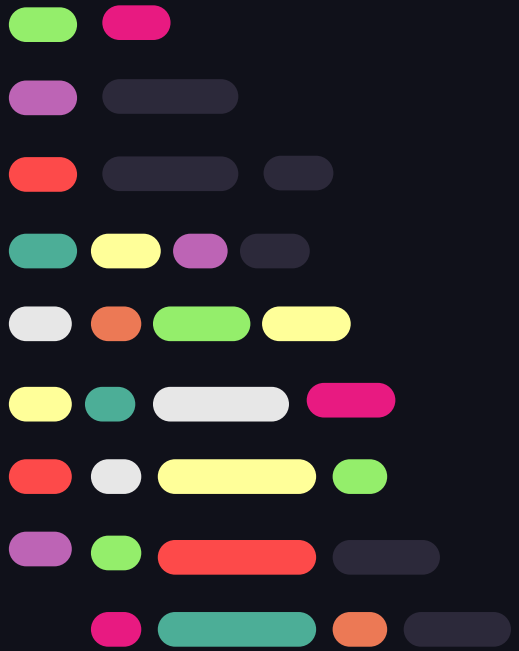
Orientação a objetos

Gestão de Memória

- ✓ Quando o objeto não for mais necessário para o restante correr do programa, não é necessário eliminá-lo explicitamente.
- ✓ O Python possui *garbage collection* de forma automática.
- ✓ Ele deteta automaticamente quando todas as referências para um espaço de memória não são mais referenciados e ela é automaticamente liberta.
- ✓ Poucos leaks de memória, e não há métodos “destrutores” em Python!

Formador: Ricardo Mourão

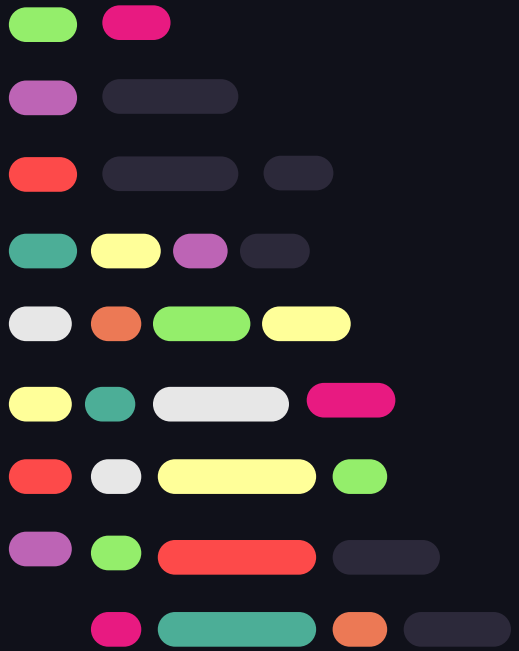
PRÁTICA! Exercicio 40



Crie uma classe chamada Livro que tenha dois atributos: titulo e autor. Instancie três objeto dessa classe e imprima os valores dos atributos.

Formador: Ricardo Mourão

PRÁTICA! Exercicio 41



Adicione um método à classe desenvolvida no exercício anterior Livro que imprime uma descrição do livro no formato:

“O livro com o titulo X foi escrito pelo autor Y".

Formador: Ricardo Mourão