PROGRAMAÇÃO EM PYTHON. ORIENTADO A OBJETOS.

Criando e exibindo um prato (Delivery de Comida)

Enunciado:

Crie a classe Prato com os atributos nome, preco e descricao. Depois, crie um objeto dessa classe e use o método exibirDetalhes() para mostrar suas informações.

Criamos a classe **Prato** com 3 atributos.

O método **__init__** é o construtor, usado para inicializar os objetos.

O método **exibirDetalhes** imprime as informações formatadas do prato.

No final, criamos um objeto p do tipo Prato e chamamos p.exibirDetalhes().

Código:

```
class Prato:
    def __init__(self, nome, preco, descricao):
        self.nome = nome
        self.preco = preco
        self.descricao = descricao

    def exibirDetalhes(self):
        print(f"{self.nome} - R$ {self.preco:.2f} | {self.descricao}")


# Teste
p = Prato("Pizza Calabresa", 35.0, "Massa fina com bastante queijo")
p.exibirDetalhes()
```

Cliente faz um pedido simples

Enunciado:

Crie as classes Cliente e Pedido. O cliente pode adicionar pratos ao pedido, e o sistema deve calcular o valor total.

A classe **Cliente** guarda apenas o nome do cliente.

A classe **Pedido** tem:

- O cliente associado ao pedido.
- Uma lista para armazenar os pratos.

- O método **calcularTotal** soma os preços dos pratos.

No teste, João faz um pedido com hambúrguer e refrigerante.

Código:

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

class Pedido:
    def __init__(self, cliente):
        self.cliente = cliente
        self.pratos = []

    def adicionarPrato(self, prato):
        self.pratos.append(prato)

    def calcularTotal(self):
        return sum([p.preco for p in self.pratos])
# Teste
c = Cliente("João")
p1 = Prato("Hamburguer", 20.0, "Carne, queijo e salada")
p2 = Prato("Refrigerante", 5.0, "Lata 350ml")

pedido = Pedido(c)
pedido.adicionarPrato(p1)
pedido.adicionarPrato(p2)

print(f"Cliente: {c.nome}")
print(f"Total do pedido: R$ {pedido.calcularTotal():.2f}")
```

Entregador entrega um pedido

Enunciado:

Crie a classe Entregador que recebe um pedido e mostra a mensagem de entrega.

O **Entregador** tem atributos nome e veiculo.

O método **entregarPedido** recebe um objeto Pedido e imprime uma mensagem da entrega.

Código:

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

    def entregarPedido(self, pedido):
        print(f"Entregador {self.nome} entregou o pedido do cliente {pedido.cliente.nome} usando {self.veiculo}")


# Teste
e = Entregador("Carlos", "Moto")
e.entregarPedido(pedido)
```

Criando e exibindo um carro (Agência de Carros)

Enunciado:

Crie a classe Carro com os atributos modelo e placa. O carro deve mostrar se está disponível ou reservado.

O carro é criado com **modelo** e **placa**.

O atributo disponivel começa como **True**.

O método especial **__str__** define como o objeto será mostrado ao usar print().

Código:

```python
class Carro:
    def __init__(self, modelo, placa):
        self.modelo = modelo
        self.placa = placa
        self.disponivel = True

    def __str__(self):
        status = "Disponível" if self.disponivel else "Reservado"
        return f"{self.modelo} ({self.placa}) - {status}"


# Teste
carro = Carro("Onix", "ABC-1234")
print(carro)
```

Cliente solicita um carro

Enunciado:

Crie a classe ClienteCarro. Quando ele solicitar um carro, este deve ficar reservado.

Código:

O cliente tenta alugar o carro.

Se o carro estiver disponível, muda o atributo disponivel para False.

Caso contrário, mostra uma mensagem de indisponibilidade.

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

    def solicitarCarro(self, carro):
        if carro.disponivel:
            carro.disponivel = False
            print(f"{self.nome} alugou o carro {carro.modelo}")
        else:
            print(f"Desculpe {self.nome}, o carro já está reservado.")


# Teste
cliente = ClienteCarro("Maria")
cliente.solicitarCarro(carro)
print(carro)
```

Esses exercícios são simples, mas já trabalham **atributos, métodos, associação entre classes, listas e controle de estado**.