Lista de Exercícios – POO, Concorrência e Boas Práticas

1. Criando uma Classe e um Objeto

Enunciado:
Crie uma classe chamada `Carro` com os atributos `marca`, `modelo` e `ano`.
Depois, instancie um objeto e exiba suas informações.

```
class Carro:
    def __init__(self, marca, modelo, ano):
        self.marca = marca
        self.modelo = modelo
        self.ano = ano

meu_carro = Carro("Toyota", "Corolla", 2022)
print(meu_carro.marca, meu_carro.modelo, meu_carro.ano)
```

Resposta:

Toyota Corolla 2022

2. Adicionando Métodos a uma Classe

Enunciado:
Amplie a classe `Carro` com um método chamado `ligar_motor()`, que imprime uma mensagem.

```
class Carro:
    def __init__(self, marca, modelo):
        self.marca = marca
        self.modelo = modelo

    def ligar_motor(self):
        print(f"O {self.marca} {self.modelo} está ligado!")

meu_carro = Carro("Ford", "Focus")
meu_carro.ligar_motor()
```

Resposta:

O Ford Focus está ligado!

3. Herança: Reaproveitando Código

Enunciado:
Crie uma classe `Veiculo` com o método `mover()`.
Depois, crie uma subclasse `Moto` que herda de `Veiculo` e adicione um método `empinar()`.

```
class Veiculo:
    def mover(self):
        print("O veículo está se movendo.")

class Moto(Veiculo):
    def empinar(self):
        print("A moto está empinando!")

minha_moto = Moto()
minha_moto.mover()
minha_moto.empinar()
```

Resposta:

```
O veículo está se movendo.
A moto está empinando!
```

4. Usando `threading` para Concorrência

Enunciado:
Crie duas funções que imprimem mensagens com um pequeno atraso e execute-as ao mesmo tempo usando `threading`.

```
import threading
import time

def tarefa1():
    print("Tarefa 1 iniciada")
    time.sleep(1)
    print("Tarefa 1 concluída")

def tarefa2():
    print("Tarefa 2 iniciada")
    time.sleep(1)
    print("Tarefa 2 concluída")

t1 = threading.Thread(target=tarefa1)
t2 = threading.Thread(target=tarefa2)
```

```
t1.start()
t2.start()

t1.join()
t2.join()
```

Resposta:
Saída (ordem pode variar):

```
Tarefa 1 iniciada
Tarefa 2 iniciada
Tarefa 1 concluída
Tarefa 2 concluída
```

5. Simulando o Escalonamento FIFO e SJF

Enunciado:
Crie uma simulação simples de escalonamento onde três processos têm diferentes durações.
Mostre a ordem de execução no FIFO (ordem de chegada) e no SJF (menor tempo primeiro).

```
processos = [("P1", 5), ("P2", 2), ("P3", 3)]

fifo = processos
sjf = sorted(processos, key=lambda x: x[1])

print("FIFO:", [p[0] for p in fifo])
print("SJF:", [p[0] for p in sjf])
```

Resposta:

```
FIFO: ['P1', 'P2', 'P3']
SJF: ['P2', 'P3', 'P1']
```

6. Aplicando Boas Práticas e PEP 8

Enunciado:
Corrija o seguinte código para deixá-lo dentro das boas práticas da linguagem (PEP 8):

```
class pessoa:
 def __init__(Self,nome,idade):
  Self.nome=nome
  Self.idade=idade
 def saudacao(Self):
```

```
  print("Ola",Self.nome)
```

Resposta corrigida:

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

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