**Fundamentos da Programação e Algoritmos**

**Unidade 4**

**Boas práticas e testes unitários**

**Carolina Lazzari Bez**

### **Título da Atividade:**

### Sistema de Planejamento de Refeições Sustentáveis - Parte 4

### **Objetivo da Atividade:**

### Aplicar conceitos de Programação Orientada a Objetos (POO) e boas práticas de programação para criar um sistema modular, reutilizável e bem documentado, integrando princípios éticos e colaborativos ao desenvolvimento de software.

### **Ferramentas Necessárias:**

Para esta atividade, utilizaremos o Google Colab como nosso ambiente de desenvolvimento.

**Instruções de Acesso às Ferramentas:**

* Acesse o site: Google Colab.
* Faça login com sua conta Google.
* Crie um novo notebook clicando em "Novo Notebook" na página inicial.
* Uma vez com o notebook aberto, você poderá começar a programar!

### **Passo a Passo:**

### **Passo 1:**

Vamos criar uma classe que modela o planejamento de refeições.

Digite e execute o código abaixo:

# Classe Refeicao

class Refeicao:

def \_\_init\_\_(self, nome, calorias):

self.nome = nome

self.calorias = calorias

def exibir\_informacoes(self):

print(f"Refeição: {self.nome} | Calorias: {self.calorias}")

# Criando objetos

refeicao1 = Refeicao("Macarrão ao molho", 500)

refeicao1.exibir\_informacoes()

**Ação:** Adicione um atributo opcional para o tempo de preparo da refeição e exiba essa informação também.

### **Passo 2:**

Agora, vamos criar subclasses para tipos específicos de refeições.

Digite e execute o código abaixo:

# Subclasse Sobremesa herdando de Refeicao

class Sobremesa(Refeicao):

def \_\_init\_\_(self, nome, calorias, contem\_acucar):

super().\_\_init\_\_(nome, calorias)

self.contem\_acucar = contem\_acucar

def exibir\_informacoes(self):

super().exibir\_informacoes()

print(f"Contém açúcar: {'Sim' if self.contem\_acucar else 'Não'}")

# Criando objetos

sobremesa1 = Sobremesa("Pudim", 250, True)

sobremesa1.exibir\_informacoes()

**Ação:** Crie outra subclasse chamada PratoPrincipal com um atributo adicional para tipo de proteína (por exemplo, "Frango", "Peixe", "Vegetariano").

### **Passo 3: Manipulação de Arquivos**

Agora, vamos salvar e carregar informações das refeições em um arquivo.

Digite e execute o código abaixo:

# Salvando informações em um arquivo

with open("refeicoes.txt", "w") as arquivo:

arquivo.write(f"{refeicao1.nome}, {refeicao1.calorias}\n")

arquivo.write(f"{sobremesa1.nome}, {sobremesa1.calorias}\n")

1. **Ação:** Leia o arquivo e exiba na tela linha por linha.
2. **Ação:** Modifique o código para registrar também o tempo de preparo e outros atributos das subclasses criadas.

### **Passo 4: Boas Práticas de Programação**

Vamos melhorar o código adicionando **docstrings** para documentar nossas classes e métodos.

class Refeicao:

"""

Classe que representa uma refeição.

Atributos:

nome (str): Nome da refeição.

calorias (int): Quantidade de calorias da refeição.

"""

def \_\_init\_\_(self, nome, calorias):

self.nome = nome

self.calorias = calorias

def exibir\_informacoes(self):

"""

Exibe informações da refeição.

"""

print(f"Refeição: {self.nome} | Calorias: {self.calorias}")

**Ação:** Adicione **docstrings** às subclasses e métodos criados nos passos anteriores.

Com esses passos, você aplicará conceitos de POO, manipulação de arquivos e boas práticas de programação, desenvolvendo um sistema eficiente e alinhado aos princípios que exploramos na disciplina.

**Parabéns pela conclusão da atividade!**

### **Entrega:**

Para submeter seu trabalho:

* **Revisão do Código:** Revise o código e garanta que ele está funcionando conforme esperado.
* **Exportação do Arquivo:** No **Google Colab**, você pode fazer isso clicando em **Arquivo > Fazer download como .ipynb**.
* **Envio da Atividade:** Acesse a plataforma do curso e faça o upload do seu notebook na atividade.