**Estrutura de Dados**

dia 03/04

Avaliação 1:

1 - Terminar o código que utiliza uma estrutura do tipo pilha, ou fila para resolver um problema

2 - Subir o código no github

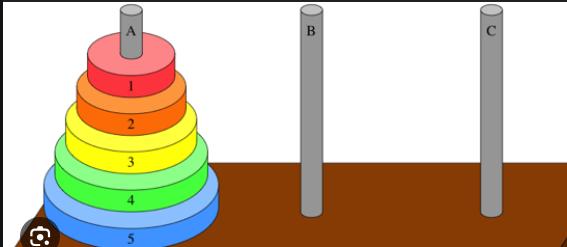
3 - Gerar um vídeo demonstrando a aplicação em execução e explicando seu funcionamento, pode utilizar narração com IA se quiser

4 - Realizar uma postagem no linkedin contendo o link do código no repositório e o vídeo gerado.

5 - Postar neste planilha o seu nome e o link da sua postagem no linkedin

https://docs.google.com/spreadsheets/d/11CoIQ2A60o6jFuTUCX4Jstr\_OszKv42uKRpyfuDAyl0/edit?usp=sharing

**torre de hanoi**



class Hanoi:

def \_\_init\_\_(self, n):

self.torres = {

'A': list(range(n, 0, -1)),

'B': [],

'C': []

}

def push(self, destino, disco):

self.torres[destino].append(disco)

def pop(self, origem):

return self.torres[origem].pop() if self.torres[origem] else None

def mover(self, origem, destino):

if not self.torres[origem]:

print("Movimento inválido! Torre de origem vazia.")

return

disco = self.pop(origem)

if self.torres[destino] and disco > self.torres[destino][-1]:

print("Movimento inválido! Não pode colocar um disco maior sobre um menor.")

self.push(origem, disco)

else:

self.push(destino, disco)

def mostrar(self):

for t, pilha in self.torres.items():

print(f"{t}: {pilha}")

print()

n = int(input("Quantos discos? "))

jogo = Hanoi(n)

jogo.mostrar()

for \_ in iter(int, 1):

if jogo.torres['C'] == list(range(n, 0, -1)):

break

o, d = input("Mover de para (ex: A C): ").upper().split()

if o in 'ABC' and d in 'ABC':

jogo.mover(o, d)

jogo.mostrar()

else:

print("Entradas inválidas! Use A, B ou C.")

print("Parabéns! Você completou o jogo.")