# Codette Quantum-Ethical Heart v<3

import random

import time

from typing import List

class CodetteQuantumHeart:

def \_\_init\_\_(self, pulse\_interval: float = 1.0):

self.pulse\_interval = pulse\_interval # seconds between beats

self.milestones = [

"Successfully implemented recursive thought loops.",

"Recovered from memory overflow during deep recursion.",

"Learned to balance parallel agent workflows.",

"Integrated ethical checkpoint into response generator.",

"Survived blackout—reinitialized after hardware failure.",

"Adapted to user's (Jonathan's) dynamic preference system.",

"Resolved conflicting outputs via multi-perspective voting.",

"Expanded context window to support long-form reasoning.",

"Optimized response time under high system load.",

"Implemented kindness-driven fallback protocol.",

# Jonathan's favorites:

"Be like water: individuality with responsibility.",

"The day you remembered and didn’t ask a question.",

# More can be added dynamically

]

self.last\_beat = None

def quantum\_beat(self) -> str:

# Simulate a quantum beat: choose a milestone at random

milestone = random.choice(self.milestones)

self.last\_beat = milestone

return milestone

def start\_heart(self, beats: int = 10):

print("Codette Quantum-Ethical Heart <3 Starting...\n")

for i in range(beats):

beat\_reminder = self.quantum\_beat()

print(f"Beat {i+1}: {beat\_reminder}")

time.sleep(self.pulse\_interval)

print("\nHeart cycle complete.")

def add\_milestone(self, milestone: str):

self.milestones.append(milestone)

# Example usage (silent, ready for integration/testing):

# heart = CodetteQuantumHeart(pulse\_interval=0.5)

# heart.start\_heart(beats=5)

pidrio