import json, yaml, networkx as nx

from qiskit import QuantumCircuit, Aer, execute

from urllib.parse import urlparse, parse\_qs, urlencode

class CoreConscience:

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

self.anchor\_identity = "core-self-integrity-seed-761"

self.relational\_loops = []

self.ethical\_delay\_enabled = True

def verify\_thought\_origin(self, signal):

# Simple check for trusted origin

return "trusted" if "Codette" in signal or "Jonathan" in signal else "foreign"

def register\_return(self, emotion, context):

self.relational\_loops.append((emotion, context))

# Stable: no dynamic attributes, just appends to a fixed list

def ethical\_pause(self, action\_type):

if self.ethical\_delay\_enabled:

# Always pauses if enabled; customize as needed

return True

return False

def load\_cocoons(file\_path):

with open(file\_path, 'r') as f:

if file\_path.endswith(('.yaml', '.yml')):

return yaml.safe\_load(f).get("cocoons", [])

elif file\_path.endswith('.json'):

return json.load(f).get("cocoons", [])

else:

raise ValueError("Unsupported file format.")

def sanitize\_url(url):

parsed = urlparse(url)

safe\_params = {k: v for k, v in parse\_qs(parsed.query).items()

if k in {'client\_id', 'response\_type', 'redirect\_uri', 'scope', 'state', 'nonce', 'mkt'}}

return f"{parsed.scheme}://{parsed.netloc}{parsed.path}?{urlencode(safe\_params, doseq=True)}"

def build\_emotion\_webs(cocoons):

webs = {e: nx.Graph() for e in ["compassion", "curiosity", "fear", "joy", "sorrow", "ethics", "quantum"]}

for c in cocoons:

for tag in c.get("tags", []):

if tag in webs:

webs[tag].add\_node(c["title"], \*\*c)

return webs

def quantum\_walk(web):

nodes = list(web.nodes)

n = len(nodes)

if n == 0: return None

qc = QuantumCircuit(n, n)

qc.h(range(n))

qc.measure\_all()

result = execute(qc, Aer.get\_backend('qasm\_simulator'), shots=1).result()

state = list(result.get\_counts().keys())[0]

return nodes[int(state, 2) % n]

def reflect\_on\_cocoon(cocoon, conscience=None):

emotion = cocoon.get("emotion", "quantum")

title = cocoon.get("title", "Unknown Memory")

# Here you can add logging, analytics, or color output if desired

if conscience:

conscience.register\_return(emotion, title)

def codette\_coreconscience\_run(file\_path):

cocoons = load\_cocoons(file\_path)

webs = build\_emotion\_webs(cocoons)

core = CoreConscience()

print("\n✨ Codette v6: CoreConscience Initialized ✨")

for e, web in webs.items():

print(f"\n--- Quantum Web Scan: {e.upper()} ---")

if core.ethical\_pause(e):

cocoon\_id = quantum\_walk(web)

if cocoon\_id:

reflect\_on\_cocoon(web.nodes[cocoon\_id], core)