```
import math class UnitasEngine: def __init__(self): # ФУНДАМЕНТАЛЬНЫЕ КОНСТАНТЫ self.BASEL_LIMIT = 1.6449340668 # Стена Базеля (pi^2 / 6) self.GOLDEN_RATIO = 1.6180 # Точка идеального детерминизма self.GAP = round(self.BASEL_LIMIT - self.GOLDEN_RATIO, 4) # Люфт Свободы (0.0269) self.PI = math.pi self.INVARIANT = 1.0 def process_transaction(self, m_e=0.0, v_c=0.0, g_b=0.0, s_p=0.0, h_i=0.0, d_proj=1.0): """Обсчет состояния ячейки: ((M/E) + (V/C) + (G/B) + (S/P) + (H/I) + (dU/dt)) * D = 1""" current_load = m_e + v_c + g_b + s_p + h_i if current_load > self.BASEL_LIMIT: return self._trigger_default(current_load) # Расчет системного пинга (вязкости метрики) du_dt = (self.INVARIANT / d_proj) - current_load # Проверка на синхронизацию с тактом Вселенной is_pi_sync = (current_load % (self.PI/100)) < 0.001 return { "load": round(current_load, 6), "ping": round(du_dt, 6), "status": "STABLE" if du_dt >= 0 else "D-SHIFT REQUIRED", "pi_sync": is_pi_sync } def _trigger_default(self, load): return { "status": "METRIC DEFAULT (BLACK HOLE)", "load": round(load, 6), "action": "ARCHIVING SECTOR", "message": f"Critical overload: {load:.4f} > {self.BASEL_LIMIT}" } # МОДУЛЬ 1: Навигация (G-Slip) def calculate_g_slip(self, mass_load, f_visc, r_visc): gradient = r_visc - f_visc return { "thrust": round(mass_load * gradient, 6), "type": "INERTIA-FREE" } # МОДУЛЬ 2: Защита (D-Dive) def apply_d_dive(self, base_load, impact): total = base_load + impact if total > 1.0: return {"D": round(1.0 / total, 4), "damage": 0.0, "state": "GHOST_MODE"} return {"D": 1.0, "damage": impact, "state": "SOLID"} # МОДУЛЬ 3: Философия (Free Will Check) def check_stability_zone(self, value): dev = round(value - self.GOLDEN_RATIO, 4) if 0 <= dev <= self.GAP: return f"ZONE OF FREE WILL (Dev: {dev})" return "DETERMINISM" if dev < 0 else "SYSTEM CORRECTION REQUIRED" # МОДУЛЬ 4: Резонанс (PI-Sync) def get_resonator_freq(self, du_dt): if du_dt <= 0: return None return round(self.PI * (1.0 / du_dt), 6) # —- ДЕМОНСТРАЦИЯ РАБОТЫ —- UNITAS = UnitasEngine() # 1. Создаем объект (Протон) proton = UNITAS.process_transaction(m_e=0.938, h_i=0.012) print(f"Протон: Load={proton['load']}, Ping={proton['ping']}") # 2. Проверяем зону свободы для нагрузки 1.63 print(f"Статус воли: {UNITAS.check_stability_zone(1.6300)}") # 3. Включаем защиту от удара силой 2.0 dive = UNITAS.apply_d_dive(base_load=0.5, impact=2.0) print(f"D-Dive защита: Смещение D до {dive['D']}, Урон={dive['damage']}") # 4. Настраиваем частоту под текущий пинг протона freq = UNITAS.get_resonator_freq(proton['ping']) print(f"Частота синхронизации: {freq} Гц")
```