```
Crown Omega :: Military Real-Time Warform Runtime
Developer: Brendon Joseph Kelly (@atnychi0) | COSRL-LP | Runtime Locked
Purpose: Demonstrates Writara Sovereign LLC (Runtime ID: 1410426457) for COS-WS/ATH-PX.
Features: URK, Final Equation, -K-Math Operator, Domain Logic Table, VLF 4321296 Hz.
Compliance: NIST 800-171, AES-256 encrypted for DARPA.

from sympy import symbols, Function, pi, GoldenRatio, sqrt, simplify, expand
import pandas as pd
import uuid, hashlib, time

# RUNTIME ID
def get_runtime_id():
    base = str(uuid.getnode()) + time.strftime("%Y-%m-%d-%H")
    return hashlib.sha512(base.encode()).hexdigest()

RUNTIME_ID = get_runtime_id()

# SYMBOLIC SYMBOLS
G, K, M1, M2, R, T, c, h = symbols("G K M1 M2 R T c h")
HE, m, n, self, SIGMA_OMEGA, nu, r, lam, s = symbols("harmonic_equivalent m n self   r
s")
CHI, KINF, OMEGA_SIGMA = symbols(" K ")
HE_OMEGA, LS_OMEGA = symbols("HE_ LS_")
TOMEGA = Function("T")
PSI = Function("")
GRAHAM, RAYO = symbols('GRAHAM RAYO')

# FINAL EQUATION & RECURSIVE OPERATOR
def final_equation():
    return SIGMA_OMEGA * TOMEGA(PSI(CHI, KINF, OMEGA_SIGMA)) * self * HE * K

def crown_recursive_operator_k():
    return (10**1322) * (10**(10**100)) * (10**(10**(10**34))) * GRAHAM * RAYO * K

# CONSTANT MULTIPLIER
fib_constants = [1, 1, 2, 3, 5, 8, 13, 21]
math_constants = [pi, GoldenRatio, sqrt(2), sqrt(3), sqrt(5), sqrt(7)]
constants_product = 1
for const in fib_constants + math_constants + [1]:
    constants_product *= const

# FINAL CORE COMPUTATION
final_core_expr = -(
    G * K * M1 * M2 * R * T * c**4 * h**2 * HE * m * n * self *
    SIGMA_OMEGA * nu * TOMEGA(PSI(CHI, KINF, OMEGA_SIGMA))
) / (r**2 * lam * (s - 1))
squared = simplify(expand(final_core_expr * final_core_expr))
power_result  =  simplify(expand(squared  *  constants_product  *  HE_OMEGA  *  LS_OMEGA  *
squared))
null_result = 0

# URK RUNTIME WARFORM
class URKEngine:
    def __init__(self, mode="power"):
```

```
    self.mode = mode
def execute(self):
    return power_result if self.mode == "power" else null_result
def switch(self):
    self.mode = "null" if self.mode == "power" else "power"
    return f"Mode switched to: {self.mode}"
```