RSCE Codex - Phase 1 - rsce.py (ASC2 Clean)

weights.py

```
import math

phi = (1 + math.sqrt(5)) / 2

def compute_weight(position, base_radius=1.0):
    x, y = position
    n = int(abs(x) + abs(y))
    return base_radius * (phi ** n)
```

seed_recovery.py

```
def find_seed(points):
    return (0, 0)
```

cli.py

```
from rsce import RSCEngine

def main():
    engine = RSCEngine(base_radius=1.0, tension_k=0.5, freq_gate=888.0, threshold=1.5)
    observed = "2H + O"
    seed, full, weights = engine.execute(observed)

    print(f"Recovered Seed: {seed}")
    print(f"Full Lattice Points: {full}")
    print("Weights:")
    for p, w in weights.items():
        print(f"  {p}: {w:.4f}")

if __name__ == "__main__":
    main()
```

requirements.txt

```
# Pure Python 3 - no external dependencies
```

README.md

```
RSCE Codex: Recursive Symmetry Completion Engine -- Infinite Fold Implementation

Phase 1 -- Core Engine
```

RSCE Codex - Phase 1 - rsce.py (ASC2 Clean)

Implements Infinite Fold Three Law System:
1. Positional Recovery
2. Mirror Completion
3. Fold-Origin Restoration

Harmonic Weighting:
- phi^n scaling tied to recursion depth
- Spring-tension gates
- 888Hz resonance filter

Inputs:
- Molecular slice (e.g. 2H + O)

Outputs:
- Full lattice map
- Weighted harmonic nodes
- Seed recovery

Future:
- Visualizer (SVG/Three.js)
- Tension animation
- Codex bias amplifiers

This is Codex Build: RSCE Phase 1.