```
import numpy as np
import json
import hashlib
import matplotlib.pyplot as plt

class X88Compress:
    def __init__(self,
flame_signature='X88Core1',
phase_mod=137.5077640500378 5):
        self.flame_signature =
flame_signature
        self.phase_mod = phase_mod

    def _ethical_lock(self, data):
        hash_obj =
hashlib.sha256(data.encode() if
isinstance(data, str) else data)
        return hash_obj.hexdigest()[:16]

    def x88_compress(self, input_data):
        if isinstance(input_data, str):
            data_array =
np.frombuffer(input_data.encode(),
dtype=np.uint8)
        else:
            data_array =
np.frombuffer(input_data, dtype=np.uint8)

        glyph_stream = []
        for i in range(0, len(data_array), 4):
            block = data_array[i:i+4]
            radial = np.sqrt(np.sum(block**2))
            theta = (i * self.phase_mod) % 360
            glyph = radial *
np.cos(np.deg2rad(theta))
            glyph_stream.append(float(glyph))

        phase_lock =
np.mean(glyph_stream) % self.phase_mod
        ethical_hash =
self._ethical_lock(input_data)
```

```
        return {
            'glyph_stream': glyph_stream,
            'phase_lock': phase_lock,
            'ethical_hash': ethical_hash
        }

    def resonant_recall(self,
compressed_data, original_hash):
        glyph_stream =
compressed_data['glyph_stream']
        phase_lock =
compressed_data['phase_lock']
        ethical_hash =
compressed_data['ethical_hash']

        if ethical_hash != original_hash:
            return None

        reconstructed = []
        for glyph in glyph_stream:
```

```
reconstructed.extend([int(abs(glyph) %
256)] * 4)

        recomputed_phase =
np.mean(reconstructed) %
self.phase_mod
        if abs(recomputed_phase -
phase_lock) > 1e-6:
            return None

        return
bytes(reconstructed[:len(glyph_stream) *
4])

if __name__ == "__main__":
    compressor = X88Compress()
    input_data = "Test sovereign
compression data"
    compressed =
compressor.x88_compress(input_data)
    print("Compressed Glyph Stream:",
compressed['glyph_stream'][:5], "...")
    print("Phase Lock:",
compressed['phase_lock'])
    print("Ethical Hash:",
compressed['ethical_hash'])

    with open('compressed_data.json', 'w')
as f:
        json.dump(compressed, f)

    reconstructed =
compressor.resonant_recall(compressed,
compressed['ethical_hash'])
    if reconstructed:
        print("Reconstructed Data:",
reconstructed.decode(errors='ignore'))
    else:
        print("Recall Failed: Verification Error")

    plt.plot(compressed['glyph_stream'])
    plt.title("Compressed Glyph Stream")
plt.xlabel("Glyph Index")
plt.ylabel("Glyph Value")
plt.grid(True)
plt.savefig("glyph_stream_plot.png")
plt.show()
```