```
import hashlib
import random

# Simple Schnorr-like demo (small numbers for illustration)
# Group setup (toy values)
q = 11           # prime order (small)
g = 2            # generator
x = 3            # secret key
h = pow(g, x, q) # public key

print("--- Honest Interactive Run ---")

# Honest prover
r = 5
# commitment t = g^r mod q
t = pow(g, r, q)
print(f"Prover commitment t = {t}")

# verifier gives challenge
e = 1
print(f"Verifier challenge e = {e}")

# prover computes response
s = (r + e * x) % (q - 1)
print(f"Response s = {s}")

# verification
left = pow(g, s, q)
right = (t * pow(h, e, q)) % q
print("Verification:", "Passed" if left == right else "Failed")

print("\n--- Cheating Attempt ---")
# cheating prover fakes t not equal to g^r
fake_t = 9
print(f"Prover fakes t = {fake_t}")

# verifier challenge
e = 0
print(f"Verifier challenge e = {e}")

# cheating prover tries random s
s = random.randint(0, q - 2)

left = pow(g, s, q)
right = (fake_t * pow(h, e, q)) % q
print("Verification:", "Passed" if left == right else "Failed")

print("\n--- Fiat–Shamir (Non-Interactive) ---")
# FS: challenge = H(t)
```

```
fs_e = int(hashlib.sha256(str(t).encode()).hexdigest(), 16) % 2
print(f"Hash-based challenge = {fs_e}")

s = (r + fs_e * x) % (q - 1)
left = pow(g, s, q)
right = (t * pow(h, fs_e, q)) % q
print("Verification:", "Passed" if left == right else "Failed")

print("\n--- Cheating Probability Experiment ---")
runs = 100
cheat_success = 0

for _ in range(runs):
    fake_t = random.randint(1, q - 1)
    e = random.randint(0, 1)
    s = random.randint(0, q - 2)
    left = pow(g, s, q)
    right = (fake_t * pow(h, e, q)) % q
    if left == right:
        cheat_success += 1

rate = cheat_success / runs
print(f"Cheating success rate = {rate:.2f} (after {runs} runs)")
```

```
--- Honest Interactive Run ---
Prover commitment t = 10
Verifier challenge e = 1
Response s = 8
Verification: Passed

--- Cheating Attempt ---
Prover fakes t = 9
Verifier challenge e = 0
Verification: Passed

--- Fiat–Shamir (Non-Interactive) ---
Hash-based challenge = 1
Verification: Passed

--- Cheating Probability Experiment ---
Cheating success rate = 0.06 (after 100 runs)
```