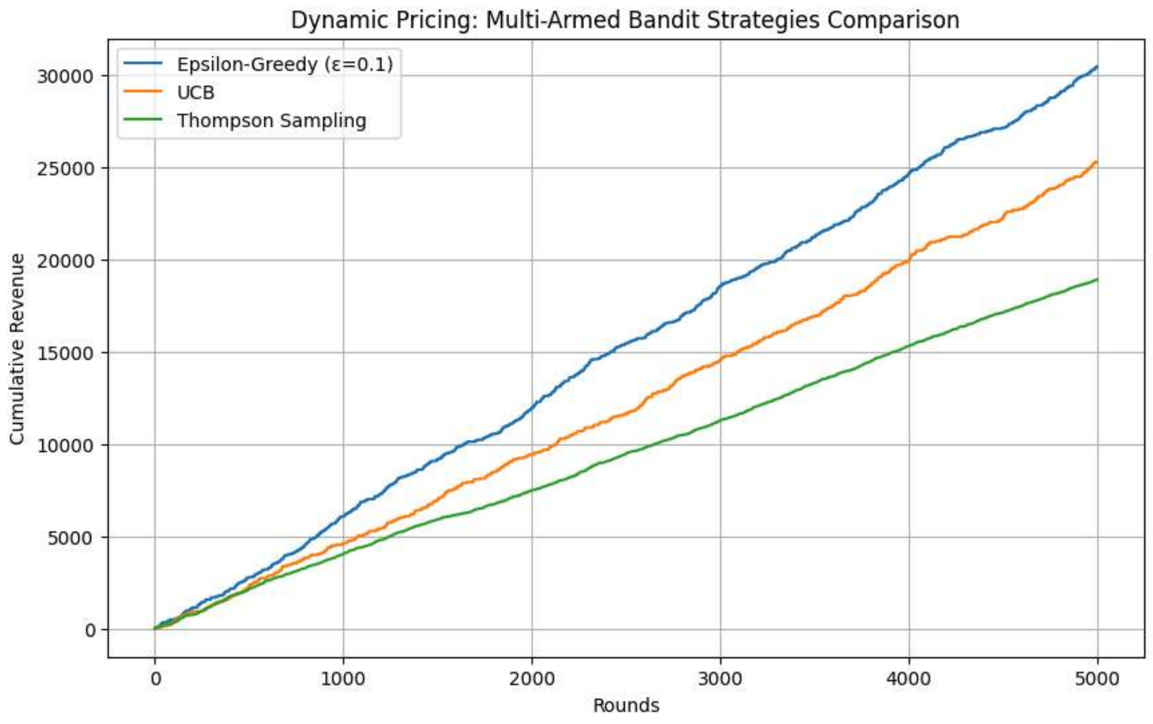
```python
import numpy as np
import matplotlib.pyplot as plt

# --- Simulation Setup ---

np.random.seed(42)
n_prices = 5                           # number of price options
prices = np.array([10, 20, 30, 40, 50])  # price points
true_conversion_rates = np.array([0.35, 0.25, 0.20, 0.15, 0.10])  # unknown to agent
n_rounds = 5000

# --- Helper function to simulate a sale or no sale ---
def simulate_sale(price_index):
    """Returns reward (revenue) if sale occurs, else 0."""
    sale = np.random.rand() < true_conversion_rates[price_index]
    return prices[price_index] if sale else 0

# --- ε-Greedy Strategy ---
def epsilon_greedy(epsilon=0.1):
    rewards = np.zeros(n_prices)
    counts = np.zeros(n_prices)
    total_rewards = []

    for t in range(n_rounds):
        if np.random.rand() < epsilon:
            price_index = np.random.randint(n_prices)  # explore
        else:
            price_index = np.argmax(rewards / (counts + 1e-5))  # exploit
        reward = simulate_sale(price_index)
        counts[price_index] += 1
        rewards[price_index] += reward
        total_rewards.append(reward)
    return np.cumsum(total_rewards)

# --- UCB Strategy ---
def ucb():
    rewards = np.zeros(n_prices)
    counts = np.zeros(n_prices)
    total_rewards = []

    for t in range(1, n_rounds + 1):
        if 0 in counts:  # play each arm once
            price_index = np.argmin(counts)
        else:
            avg_reward = rewards / counts
            confidence = np.sqrt(2 * np.log(t) / counts)
            ucb_values = avg_reward + confidence
            price_index = np.argmax(ucb_values)
        reward = simulate_sale(price_index)
        counts[price_index] += 1
        rewards[price_index] += reward
        total_rewards.append(reward)
    return np.cumsum(total_rewards)

# --- Thompson Sampling ---
def thompson_sampling():
    successes = np.ones(n_prices)
    failures = np.ones(n_prices)
    total_rewards = []

    for _ in range(n_rounds):
        sampled_rates = np.random.beta(successes, failures)
        price_index = np.argmax(sampled_rates)
        reward = simulate_sale(price_index)
        if reward > 0:
            successes[price_index] += 1
        else:
            failures[price_index] += 1
        total_rewards.append(reward)
    return np.cumsum(total_rewards)

# --- Run Simulations ---
eps_rewards = epsilon_greedy(epsilon=0.1)
ucb_rewards = ucb()
ts_rewards = thompson_sampling()
```

```
# --- Compare Results ---
plt.figure(figsize=(10,6))
plt.plot(eps_rewards, label='Epsilon-Greedy (ε=0.1)')
plt.plot(ucb_rewards, label='UCB')
plt.plot(ts_rewards, label='Thompson Sampling')
plt.xlabel("Rounds")
plt.ylabel("Cumulative Revenue")
plt.title("Dynamic Pricing: Multi-Armed Bandit Strategies Comparison")
plt.legend()
plt.grid(True)
plt.show()

# --- Print Final Revenues ---
print(f"Total Revenue (Epsilon-Greedy): {eps_rewards[-1]:.2f}")
print(f"Total Revenue (UCB): {ucb_rewards[-1]:.2f}")
print(f"Total Revenue (Thompson Sampling): {ts_rewards[-1]:.2f}")
```



```
Total Revenue (Epsilon-Greedy): 30460.00
Total Revenue (UCB): 25290.00
Total Revenue (Thompson Sampling): 18920.00
```