Design Summary

Data Structures

- **Ring Buffers (Lists, max 30 elements):**
    - Used for storing recent competitor prices, own prices, demands, and profit history.
    - Why? Keeps memory usage predictable and low, avoids unbounded growth.
    - Size: 30 periods (enough for short-term trends, fits well within RAM limits).

Online Update Equations

- **Mean & Variance (Welford's Algorithm):**
    - Updates mean and variance in O(1) time, no need to recalculate from scratch.
    - Example (for demand):

```
n += 1
delta = new_value - mean
mean += delta / n
m2 += delta * (new_value - mean)
```

    - Used for both demand and competitor price statistics.
- **Profit Momentum (Exponential Moving Average):**
    - Tracks changes in profit to adapt base price.
    - Formula:

```
momentum = alpha * profit_change + (1 - alpha) * previous_momentum
```

    - Alpha is set to 0.1 for smooth adaptation.

Cold Start & Edge Case Handling

- **Cold Start:**
    - If no previous state, initialize all buffers and stats to default values.
- **Corrupted State:**
    - If state is missing or invalid, fall back to cold start.
- **Price Bounds:**
    - All prices are clipped to [1.0, 20.0] for safety.
    - If price calculation fails (NaN/Inf), revert to base price.

Policy & Trade-offs

- **Base Price:**
  - Starts at 10.0, nudged by profit momentum and demand trend.
- **Competitor Response:**
  - Follows competitor price partially (30% gap), plus a trend-following signal.
- **Demand Response:**
  - Raises price if demand is strong and stable, lowers if weak.
  - If demand is volatile, price is more conservative.
- **Why this approach?**
  - Simple, fast, and robust.
  - No heavy ML, just smart use of streaming stats and bounded history.
  - Easy to debug and explain.

Rationale

- **Efficiency:**
  - All updates are O(1), so the function is fast and meets timing constraints.
- **Memory:**
  - Bounded buffers ensure memory stays low, even for long runs.
- **Robustness:**
  - Handles missing/corrupted data gracefully.
- **Transparency:**
  - Logic is clear, easy to audit, and tweak.

For details, see comments in duopoly.py.