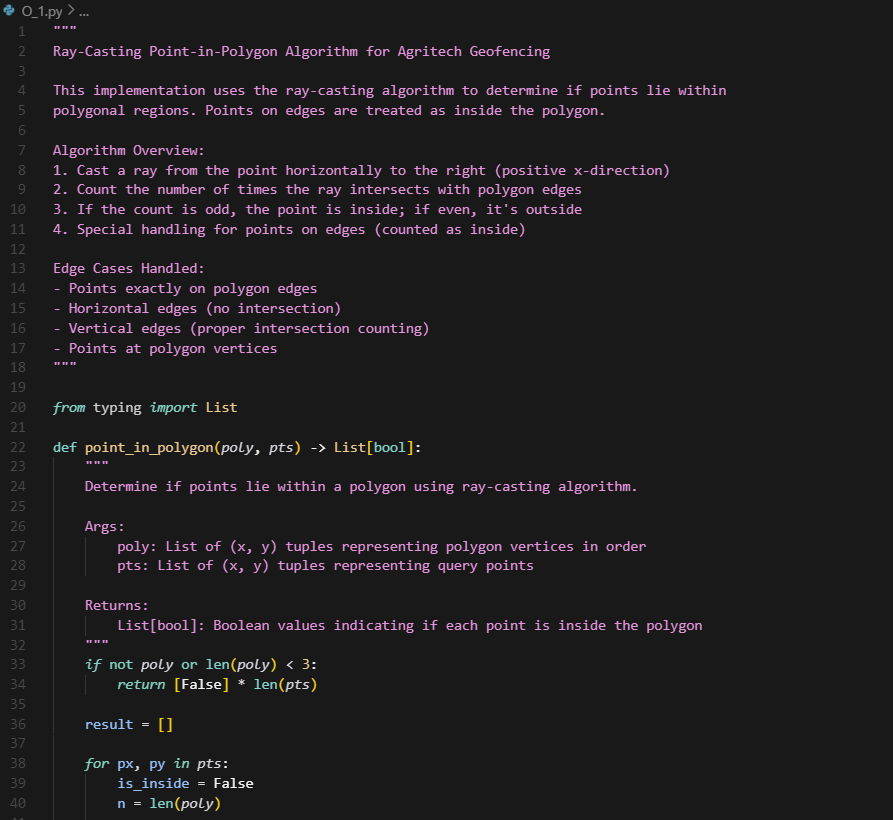
Subgroup O  
O.1 — [S13O1] Point-in-polygon (ray casting)  
Scenario (agritech):  
Context:  
Geofencing in agritech requires checking if points lie within polygonal regions.  
Your Task:  
Implement ray-casting point-in-polygon; treat points on edges as inside.

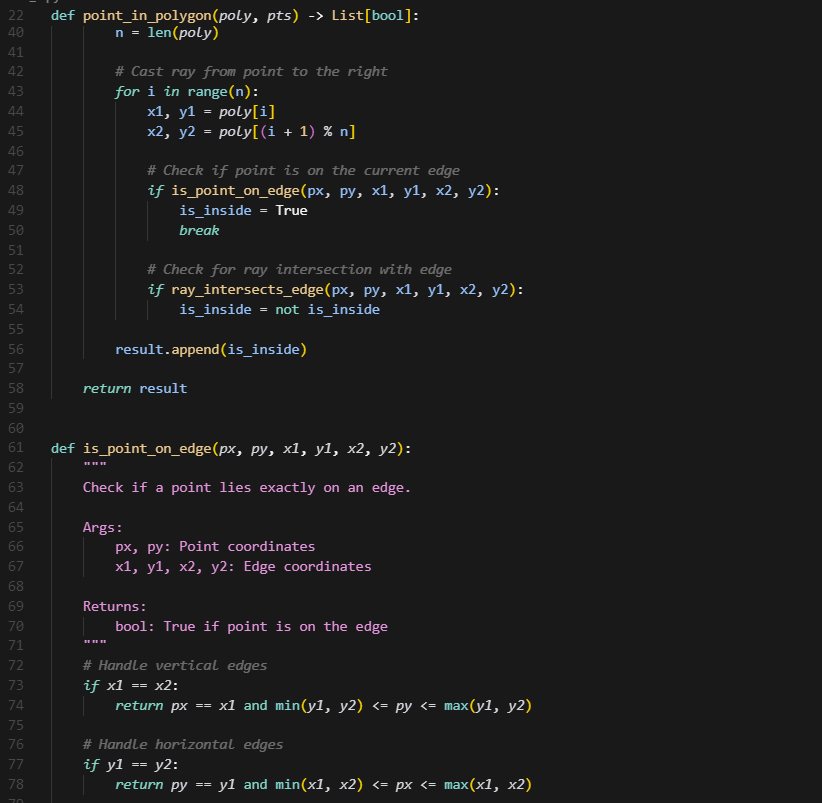
Data & Edge Cases:  
Square example provided; return boolean list for queries.  
AI Assistance Expectation:  
Ask AI to outline crossing-count logic and edge handling.  
Constraints & Notes:  
Return list[bool] aligning to input order.  
Sample Input  
poly=[(0,0),(4,0),(4,4),(0,4)], pts=[(2,2),(5,5)]  
Sample Output  
[True, False]  
Acceptance Criteria: Edges counted as inside

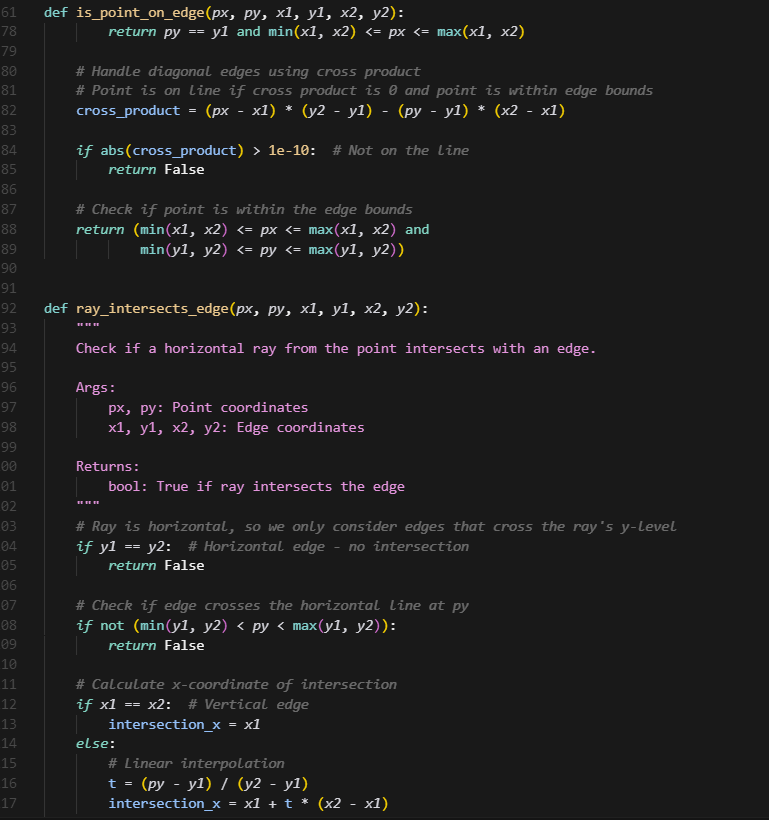
Prompt :

Implement ray-casting point-in-polygon for agritech geofencing. Return List[bool] for query points, treating points on edges as inside. Input: poly=[(x1,y1),...], pts=[(px1,py1),...]. Output: [bool1, bool2, ...].

Code :



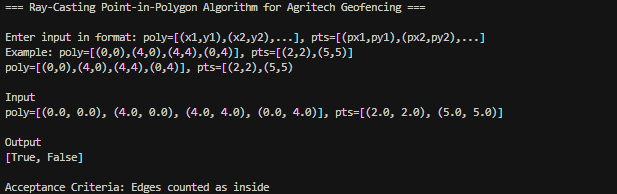








Output :



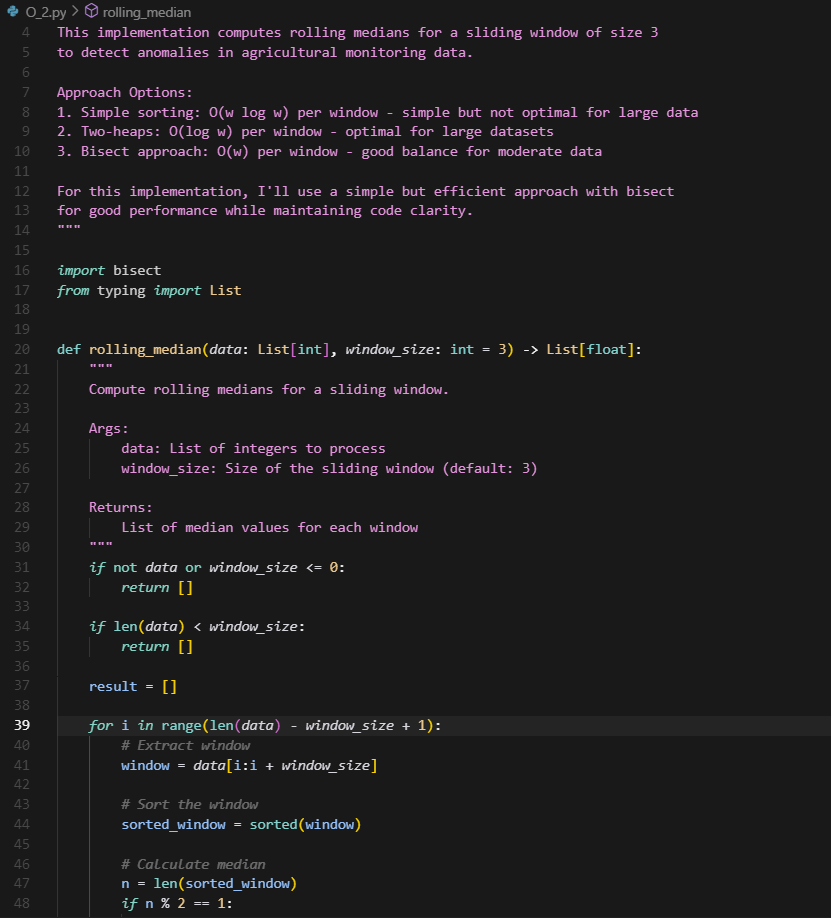
O.2 — [S13O2] Compute rolling median (w=3)  
Scenario (agritech):  
Context:  
A agritech monitoring job computes rolling medians (w=3) for anomaly detection.  
Your Task:  
Return the median for each sliding window; prefer an efficient approach.  
Data & Edge Cases:  
Example provided; for large n, two-heaps or bisect approach acceptable.  
AI Assistance Expectation:

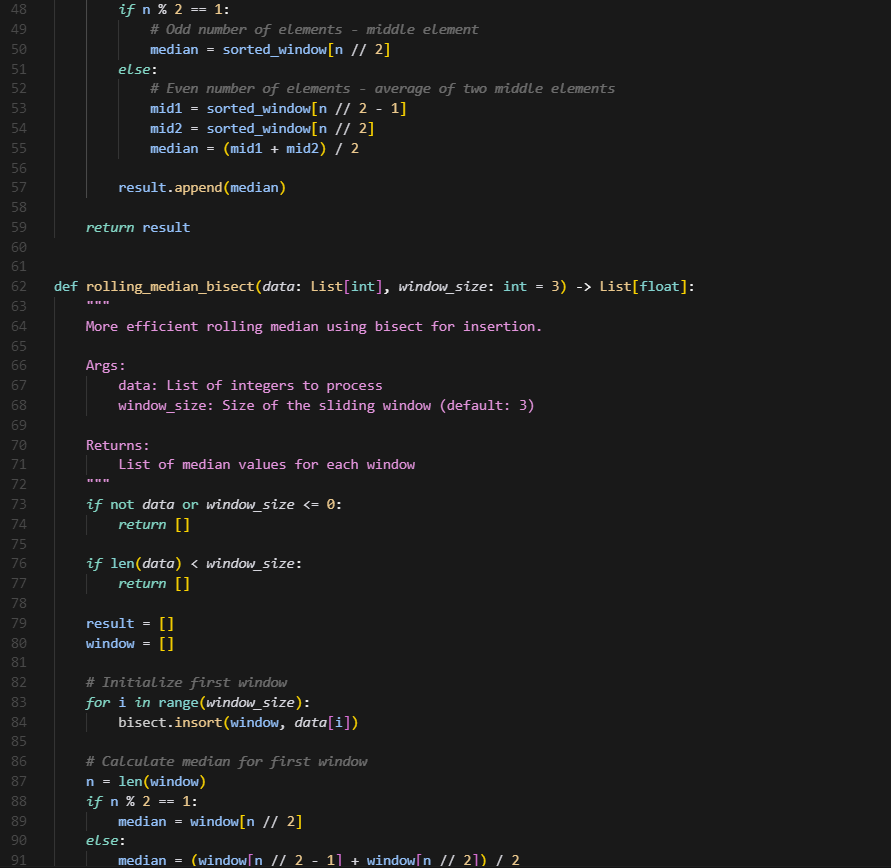
Use AI to propose options and complexity; implement simple but efficient enough method.  
Constraints & Notes:  
Include tests for short lists and exact window size.  
Sample Input  
[1, 3, 2, 5, 4]  
Sample Output  
[2, 3, 4]  
Acceptance Criteria: Efficient and correct

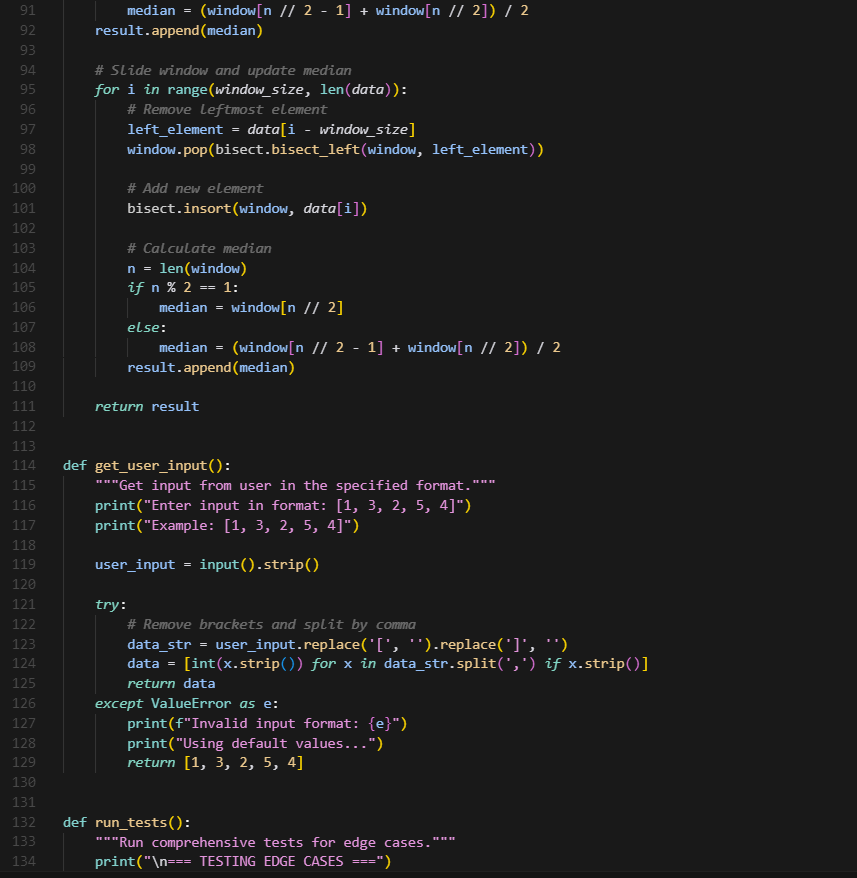
Prompt :

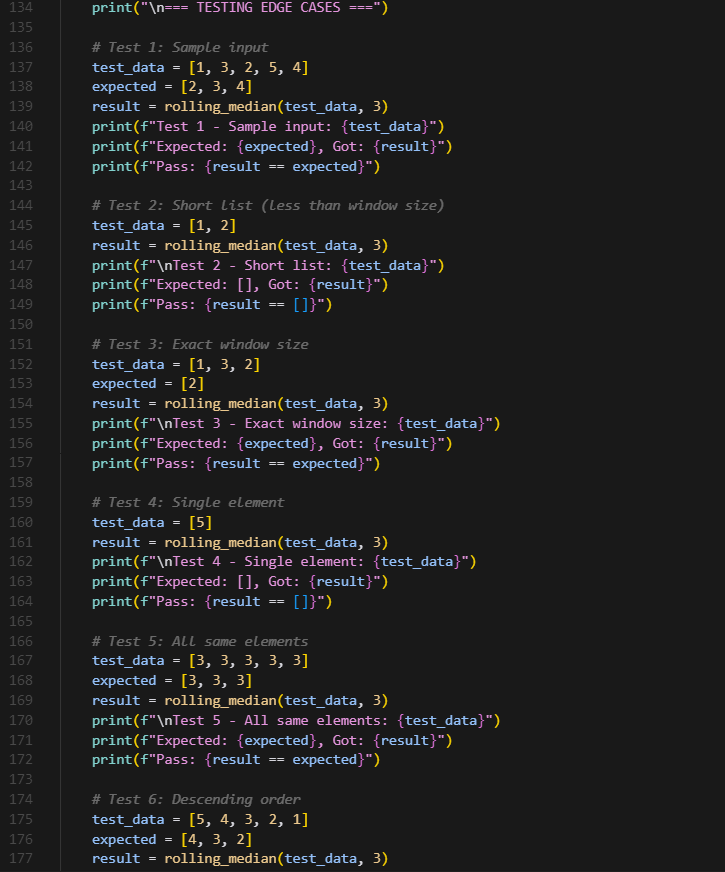
Implement rolling median algorithm for agritech anomaly detection with window size 3. Return List[float] of median values for each sliding window. Input: [1, 3, 2, 5, 4]. Output: [2, 3, 4]. Handle edge cases: short lists, exact window size. Use efficient approach (bisect/sorting).

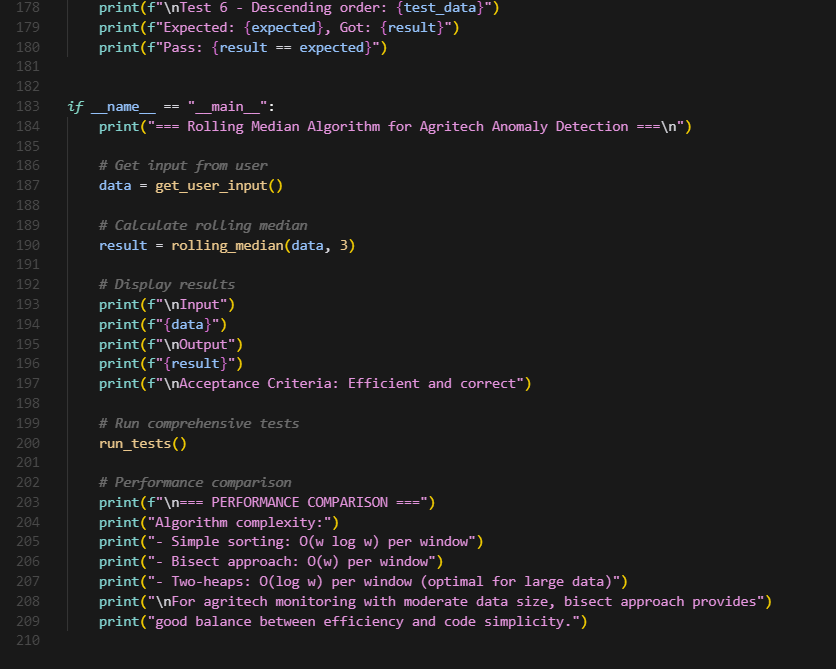
Code :











Output :

