**6 kyu**

**Tick Toward**

28491% of 98113 of382[ptolemybarnes](https://www.codewars.com/users/ptolemybarnes)

Python

* [TRAIN AGAIN](https://www.codewars.com/kata/tick-toward/train/python)
* [NEXT KATA](https://www.codewars.com/trainer/python)

Details

[Solutions](https://www.codewars.com/kata/tick-toward/solutions/python)

[Forks (6)](https://www.codewars.com/kata/tick-toward/forks/python)

[Discourse (49)](https://www.codewars.com/kata/tick-toward/discuss/python)

* Add to Collection
* |
* Share this kata:

Define a function that generates medial values between two coordinates and returns them in an array from start to target. For example, if the starting point is [1,1] and the target point is [3,2] then the shortest route from start to target would be [[1,1], [2,2], [3,2]]. The route should go only through integral coordinates.

Note: you should move diagonally until the path from your current position to the target can be represented as a fully horizontal/vertical line.

Examples:

tick\_toward((5,5), (5,7)) == [(5,5), (5,6), (5,7)]

tick\_toward((3,2), (4,5)) == [(3,2), (4,3), (4,4), (4,5)]

tick\_toward((5,1), (5,-2)) == [(5,1), (5,0), (5,-1), (5,-2)]

Note: tuples will be used for representing coordinates in Python.

<https://www.codewars.com/kata/tick-toward/python>

1. **def** tick\_toward(start, target):

4. min\_x = min(start[0], target[0])
5. max\_x = max(start[0], target[0])
7. c\_x = []
8. **for** i **in** range(min\_x, max\_x + 1):
9. c\_x.append(i)
11. min\_y = min(start[1], target[1])
12. max\_y = max(start[1], target[1])

15. c\_y = []
16. **for** i **in** range(min\_y, max\_y + 1):
17. c\_y.append(i)

20. **if** (start[0] > target[0]): c\_x.reverse()
21. **if** (start[1] > target[1]): c\_y.reverse()
23. length = max(len(c\_x), len(c\_y))
25. lista = []
27. **for** i **in** range(0, length):
28. **if**(i >= len(c\_x)):
29. *#lista.append(Punto(c\_x[len(c\_x) - 1], c\_y[i]))*
30. lista.append( (c\_x[len(c\_x) - 1], c\_y[i]  )  )
31. **elif**(i >= len(c\_y)):
32. *#lista.append(Punto(c\_x[i], c\_y[len(c\_y) - 1]))*
33. lista.append(  (c\_x[i], c\_y[len(c\_y) - 1])   )
34. **else**:
35. lista.append((c\_x[i], c\_y[i]))
37. **return** lista