PYTHON EXERCISE

RANDOM WALKS IN MONTE CARLO

01 – BASIC LIBRARY FOR RANDOM WALKS

The first task is to implement a library / set of utility functions that provide the necessary methods to perform random walks as a module that is reused in the other tasks.

- Declare a tuple of options on where to walk at each crossing. The options are N, E, S, W for the four cardinal directions.
- Implement a generator function `def generate_walk(blocks = 1):` that generates walks of length #blocks consisting of the previously defined options.
- Implement a function `def decode_walk(walk):` that takes a #walk and decodes it into vector (tuple) that contains the change in the position after the walk (dx, dy).
- Implement a distance function `def distance_manhattan(start, end):` that calculates the Manhattan distance[1] between a #start and #end position.
- Implement `def do_walk(blocks, dist = distance_manhattan)` that generates a walk of length #blocks, calculates the distance from the starting position and returns a tuple containing the walk and the walked distance.
- Put everything together in a function for walk analysis that generates #repetitions walks from length 1 to #max_blocks. The function returns a dictionary which uses the maximum length as key and contains the generated walks and distances as tuple.
  `def monte_carlo_walk_analysis(max_blocks, repetitions = 10000):`

[1] https://en.wikipedia.org/wiki/Manhattan_distance