PYTHON EXERCISE
RANDOM WALKS IN MONTE CARLO

04 – MEMORY USAGE

This task consists of two subtasks. The first one is to research and describe in your own words how garbage collection works in Python. Additionally, provide an example that demonstrates the usage of the sys module[1] for determining reference counts and the garbage collector module[2].

When profiling our simple library for random walk generation with tracemalloc[3], you will notice that the generated walks take up lots of memory. However, the generated walks are not always necessary to be included at all times (depending on the analysis). Therefore, we adapt our function for performing walks

```
def do_walk(blocks, dist = distance_manhattan, gen_walk = True):
```

by including an additional parameter `gen_walk` that specifies whether the walks are actually stored (anywhere) and returned. If `gen_walk` is set to false only the walked distance is returned.

Then, a function `def monte_carlo_walk(max_blocks,repetitions=100000):` has to be created that works in the same way as `monte_carlo_walk_analysis`, but does not store the generated walks (only the walked distances).

Provide a script that uses tracemalloc to show the allocated memory when generating 10,000 walks for each maximum length up to 20. The script should highlight the differences when generating the walks with `monte_carlo_walk_analysis` or with `monte_carlo_walk`. tracemalloc also provides the line numbers of the sources files, where the memory has been allocated. Discuss and explain why the top three memory allocations, you were responsible for, occur at the described locations.

Hints:

- Provide a detailed documentation of the performed analysis and possible improvements.
- The source file should be called Memory.py

[1] https://docs.python.org/3/library/sys.html
[2] https://docs.python.org/3/library/gc.html
[3] https://docs.python.org/3/library/tracemalloc.html