Activity 1. Measuring execution times

1. How many more years can we continue counting in milliseconds starting from January 1st, 1970? Using a 64-bit format to store we will run out dates roughly by year 292276973.
2. What does t=0 mean? It means the execution time was smaller than a millisecond.
3. From what size of the problem do times start to be reliable? At about n=280 times start to be higher than 50 milliseconds, although with some still being under.

Activity 2. Problem size growth?

1. What happens if n1 = 5\*n0? The execution times have a quite steeper incline per iteration.
2. Are results the expected from O(n)? Although the plotted data may not reflect it, the times are expected from an O(n) functions, it doesn’t look that way because of the different increase of loads.
3. Graph

Activity 3. Taking small execution times

1. Measurements table
2. Main components of the computer that did the work: Processor and (RAM) memory.
3. Do values meet expectations?

Activity 4. Operations on matrices

1. Measurements table

Refer to excel file in the same folder.

1. Main components of the computer that did the work: Processor and (RAM)
2. Do values meet expectations?

The shape of the curves do match those expected, and one taking longer than the other does also.

Activity 5. Benchmarking

1. Why are there differences in execution times?

Because the compilers result in different code, and also because the virtual machines of the different languages might have different performance themselves.

1. Is there any analogy between the behaviours of the two implementations?

Both have the same complexity, so although times differ from Java and Python, the resulting curve of execution would have the same shape.