|  |  |
| --- | --- |
| **Value of n** | **Time(s)** |
| 5 | 0.0000050067901611328125 |
| 15 | 0.00000476837158203125 |
| 30 | 0.0000057220458984375 |
| 60 | 0.00000762939453125 |
| 120 | 0.000008344650268554688 |
| 350 | 0.000027418136596679688 |
| 850 | 0.0001494884490966797 |
| 1500 | 0.0006921291351318359 |
| 1600 | ------- |
| 1700 | --------- |
| 2500 | ---------- |

**Empirical analysis of Factorial Calculation**

**Iterative:**

|  |  |
| --- | --- |
| **Value of n** | **Time(s)** |
| 5 | 0.000001430511474609375 |
| 15 | 0.0000016689300537109375 |
| 30 | 0.000003814697265625 |
| 60 | 0.0000054836273193359375 |
| 120 | 0.00001049041748046875 |
| 350 | 0.00012135505676269531 |
| 850 | 0.00037550926208496094 |
| 1500 | -------- |
| 1600 | ------- |
| 1700 | --------- |
| 2500 | ---------- |

**Recursive:**

The iterative method for calculating factorials is more efficient and capable of handling larger values of (n) compared to the recursive method, which is more prone to hitting Python's recursion limit for large (n) values. The iterative method shows faster execution times for small ( n), but encounters overflow issues around (n = 1600), while the recursive method faces performance constraints and recursion limit errors around (n = 850). In summary, the iterative approach is generally more robust for handling large values but both methods have their limits with very large (n).

**Sami Rafat Amarneh**

**202210344**