Part 1

What is the big O notation for calc_factorial?

O(n)

What is the big O notation for calc_stats?

O(n)

What is the big O notation for print_triangle?

O(n^2)

What is the big O notation for display_letters_in_names?

O(n)

What is the big O notation for average = sum(numbers) / len(numbers)?

O(n)

What is the order of O(n^2), O(log n), and O(n) (best performance first and worst performance last)?

O(log n), O(n), O(n^2)

Part 2

Do the actual results agree with the big O predictions made earlier? If not, what do you think the big O should be?

1 is O(n) and that checks out 2 is O(n^2) and that checks out 3 is O(log n) and that checks out

Which function (algorithm1, algorithm2, algorithm3) has the best performance and which one the worst performance?

3 is the best and 2 is the worst, just looking at the time.

Looking at the results, why do we say that big O only applies when n is "large"

You can barely count the difference in milliseconds on the first round, but as n got higher the difference was much easier to tell apart.