As computer scientists, we write and execute programs all the time.

For example, think of a program that looks for patterns or computes statistics over a large database, say Amazon's database.

For further example, think of a program that controls and supports a massive online game, say World of Warcraft.

The programs that support such applications should, not only run correctly (i.e., no bugs), but run very fast, in time, and require a reasonable amount of memory, as the size of the input grows.

(e.g., Amazon's database grows all the time, from thousands, to millions, to hundreds of millions of transactions, and the number of players of a massive online game, double, quadruple, etc., as the game gains popularity.)
The point is that, in computer science applications, we must quantify the performance of a program in terms of the size of the input.

When you write or execute a program, you should know how much time/memory the program requires when the input is of size 100, 1000, 1,000,000, 100,000,000.

In general, when the input size is $\infty$, you say your program required $f(n)$ time/memory.

Program performance is a function of the input size $n$. 
ORDER OF GROWTH
is a convenient short way
to classify performance
in broad categories.

Example:
Suppose the performance of your program is
\[ f(n) = n^2 + 100 \log n + 3 \]
and the performance of your competitor's program is
\[ g(n) = 1,000,000 n^4 + 100,000,000 n^3. \]
Whose program is "better"?
\[ g(n) = 1,000,000 \cdot n^4 + 100,000,000 \cdot n^3 \]

has these large numbers (million, 100 million), but, its leading term is \( n^4 \).

\[ f(n) = n^2 + 100 \log n + 3^n \]

does not have any huge constants, like millions or billions, but its leading term is \( 3^n \).
running time $f(n)$ vs $g(n)$

$f(n) = n^2 + 100 \log(n) + 3^n$

$g(n) = 1,000,000 n^4 + 100,000,000 n^3$

maybe $g(n)$ will be bigger than $f(n)$ for some values of $n$ relatively small, say up to at most a hundred ($n_0 \leq 100$).

but then the exponential $3^n$ will pick up and make $f(n)$ much-much bigger, always, than $g(n)$ whose running time is determined by some $n^4$. 
Do a sanity test as an exercise.

Say \( g(n) = 10^6 \cdot n^4 + 10^8 \cdot 3 \)

\( f(n) = n^2 + 100 \cdot \log n + 3^n \)

Let us see

\( g(1) = 10^6 + 10^8 \) (in the hundreds of millions)

\( f(1) = 1 + 3 = 4 \)

\( g(2) = 10^6 \cdot 16 + 10^8 \cdot 8 \) (still in the hundreds of millions)

\( f(2) = 4 + 100 + 9 = 113 \)

\( g(100) = 10^6 \cdot (100)^4 + 10^8 \cdot (100)^3 \)

\( f(100) = 3^{100} \gg 10^{30} \)

Can you compare easily \( 2 \cdot 10^{14} \) with \( 3^{100} \)?

\( 3^{100} > 3^{3 \times 30} = (3^3)^{30} = 27^{30} \gg 10^{30} \)

So already by \( n = 100 \), \( 3^n \) has picked up growth versus \( 100,000,000 \cdot n^4 \) hands down.