Assumptions:

- All primitive operations take constant amounts of time
- following operations require constant # of primitive ops.
  - reading/writing/modifying primitive data types.
  - arithmetic & logical operations on primitive data types
  - method calls
- following ops scale linearly with size (# of primitive data types)
  - initializing arrays & strings
  - creating <u>new</u> object instance.

Section 3 Big O notation

Throughout today, $f, g, h : \underbrace{\mathbb{N}}_{\text{instance size}} \longrightarrow \underbrace{\mathbb{R}^+}_{\text{running time of procedure on a particular machine}}$

Def: $f, g : \mathbb{N} \to \mathbb{R}^+$, We write $f = O(g)$
if $\exists N \in \mathbb{N}$ & $C \in \mathbb{R}$, such that
$\forall n \geq N$, $f(n) \leq C \cdot g(n)$.

"f scales at most as quickly as g."

Ex: $f(n) = 3n^2 + 12$, $g(n) = n^2$.
So $g(n) = O(3n^2+12)$, $f(n) = O(n^2)$.

Properties of O

- If f is bounded, $f = O(1)$.
- If $f(n) \leq g(n)$ $\forall n$, $f = O(g)$
- If $f = O(g)$, then $c \cdot f = O(g)$ $\forall c \in \mathbb{R}$.
- If $f = O(g)$, then
  $f + O(g) = O(g)$
  $g + O(f) = O(g)$
- If $f_1 = O(g_1)$, $f_2 = O(g_2)$, then
  $f_1 f_2 = O(g_1 g_2)$.

Example: $f(n) = 10n^2 + 100n + 1000 = O(n^2)$.

$f(n) = 10n^2 + 100n + \underbrace{1000}_{O(1)}$. (prop. 1)

$\Rightarrow O(f) = O(10n^2 + 100n)$.

$\Rightarrow O(f) = O(n^2 + n)$ (prop. 3)

But $O(n) \leq O(n^2)$, so $O(f) = O(n^2$