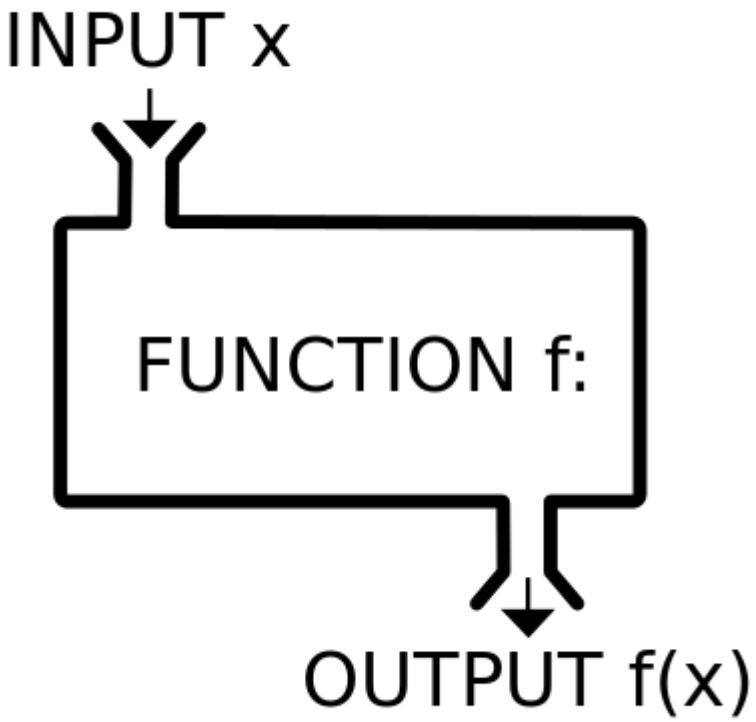
Functions and Their Growth

This lesson discusses the building blocks for analyzing algorithms.

The yardstick to measure the performance of algorithms is specified in terms of functions. For the mathematically uninitiated, we explain functions below.

Functions

Think of a function like a *machine* or a *blackbox* that takes inputs from one end and spits outputs from the other end.



Function is just a black-box that takes in an input and spits out a result denoted by f(x)

Let's consider the following function:

$$f(n) = n^2$$

This is the traditional format of expressing a function, with n as the input fed to the "machine". The output of the "machine" is expressed in terms of the input; in the above function, the output is $\boldsymbol{n^2}$ defined in terms of the input $\boldsymbol{n}$. The input n variable can take on different values and the output will vary accordingly. The below table captures the output values generated when **n** takes on different interger values.

$f(n)$	n^2
f (0)	0
f (1)	1
f (2)	4
f (3)	9
f (4)	16
f (5)	25

Comparing Functions

Now you know what a function is. Let's see how functions compare with each other. Consider the following two functions:

$$f(n) = n^2$$

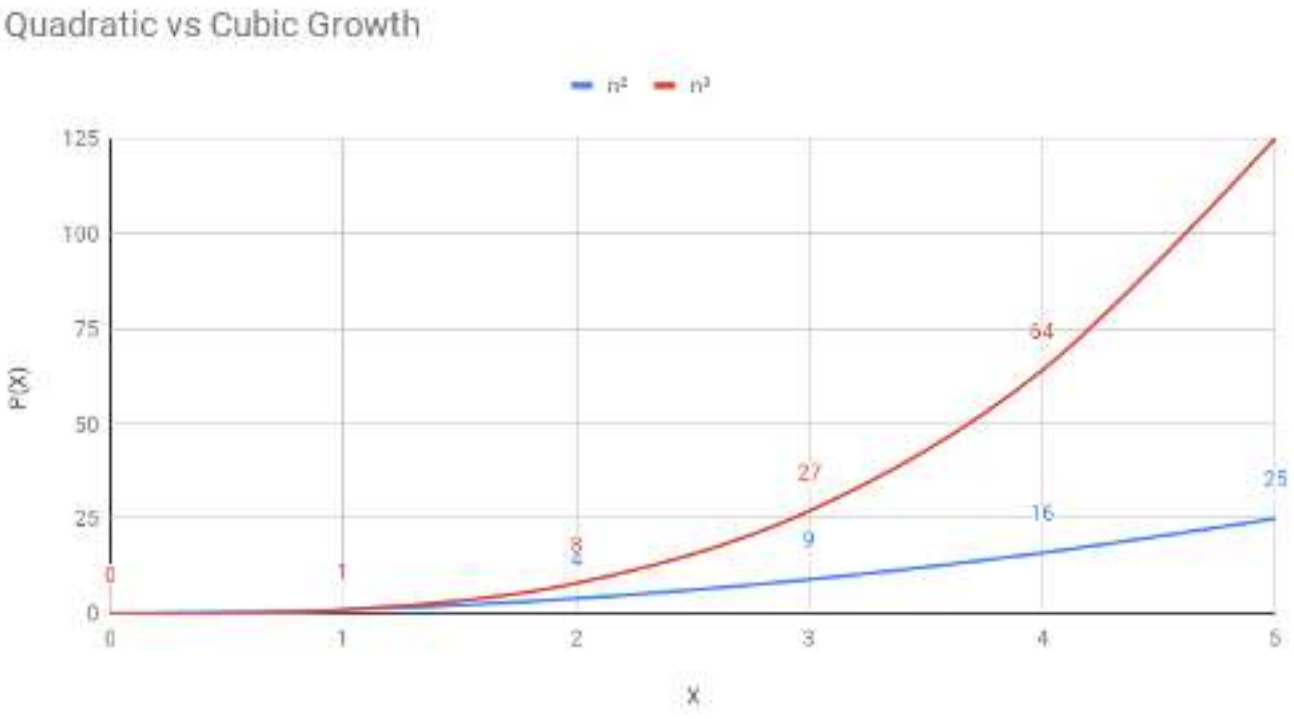
$$f(n) = n^3$$

If we restrict the values n can assume to only non-negative numbers, then we can determine which of the two functions will produce a numerically bigger output than the other. *The function which produces a bigger number will*

output than the other. ***The function which produces a bigger number will be said to grow faster than the other.***

$f(n)$	n^2	n^3
f (0)	0	0
f (1)	1	1
f (2)	4	8
f (3)	9	27
f (4)	16	64
f (5)	25	125

We can observe from the output that the function $f(n) = n^3$ grows faster than the other function $f(n) = n^2$ when n > 1.



Relating Back to Insertion Sort

In the previous section, we worked out an algebraic expression for the total number of instructions executed when sorting an input array of size n. That

expression is expressed as a function of n below:

$$f(n) = [2 * (n + 1) + 2n] + [2n + 7[\frac{n(n-1)}{2}]]$$

We have captured the time complexity of insertion sort as a function of the input size. In upcoming sections, we will now be able to wield developed mathematical tools to gain further insights.

Comparing functions may remind you of our race-track example at the beginning of the course. We are eventually headed in that direction, where we can model the performance of algorithms as functions, whose input n represents the size of the input to the algorithms. We will not be directly interested in the values the functions spit out. Rather, how they grow and the growth of functions is dependent on the values they produce.