It's sort of a broad term, so it's hard to come up with a complete definition. What I would say is that, at least in this context, it's something that programmers use to "guide" their algorithm in the correct direction. You often need a heuristic because the regular version of the algorithm is too slow.

For example, let's say you want to find a path from one place to another. The obvious way to do it is to just fan out, checking each "node" on the map to see if it's blocked or not. Eventually, you'll find a way around the obstacle to target. That's called Dijkstra's algorithm (or, really, a mega-simplified version of it), and it looks sort of like [this](http://upload.wikimedia.org/wikipedia/commons/2/23/Dijkstras_progress_animation.gif).

If we use a heuristic, though, we can make things faster. In this case, what we do is use a heuristic that approximates how far each node is from the target. The algorithm will favor checking nodes with a lower heuristic score (ie closer to the target) first, rather than just checking all the nodes blindly. So, the result is something like [this](http://upload.wikimedia.org/wikipedia/commons/5/5d/Astar_progress_animation.gif), where you can see that it's rushing straight towards the target rather than blindly checking everywhere.

That's the heuristic at work; it's guiding the algorithm in what we can assume is probably the right direction, making it much faster. In this case it's obvious because there is literally a direction that the algorithm should go in, but heuristics can be much more abstract than that.