Why Study Theory of Computation?

Theory of Computation (ToC) is a part of theoretical computer science. ToC is usually split up into three parts: *automata theory*, *computability theory* and *computational complexity theory*.

1. Computability theory deals with the question of Which problems can be solved by computers and which ones cannot? Interestingly we do not need computers to answer these questions. Rather we can do that by using abstract *mathematical* models of computers.
2. Automata theory is the study of abstract mathematical systems or machines (called automata) and the computational problems that can be solved using these machines.
3. Computational complexity theory concerns whether a problem can be solved at all on a computer as well as how efficiently the problem can be solved. The above said three parts are linked by the question: *What are the fundamental capabilities and limitations of computers?*

This course largely concerns automata theory as well as computability theory. You have studied computational complexity theory as part of your algorithms course.

In essence, ToC is primarily concerned with the study of how problems can be solved using *computational* means or algorithms. While solving problems, we might encounter many practical problems with similar characteristics. Quite often in such scenarios, it is often beneficial not to solve them one by one, but to find a general algorithm that would enable us to solve all these problems. However, one should not be wasting time to look for a general algorithm for those problems for which generality is not possible. Consequently, it is desirable to know which problem can be algorithmically solved and which cannot be. Understanding which problems can be algorithmically solved is one of the main objectives of theory of computation.

In situations when an algorithm is, in principle, possible, sometimes, the only possible algorithm requires so much computation time -- e.g., longer than the lifetime of the Universe -- that they are not practically feasible. It is therefore desirable to know if the given problem can be feasibly solved. This is also one of the main objectives of theory of computation. To cite interesting examples, When we write a program, it is desirable to be able to check that this program always halts, and that it always produces a correct result. It may therefore seem reasonable to look for an algorithm that, given a program, checks whether it halts or not -- but theory of computation proves that such algorithms are not possible!! Similarly, it may seem reasonable to look for a general algorithm that, given a program that always halts, checks whether this program always produces the correct result -- but theory proves that such algorithms are not possible either.

References:

1. https://web2.qatar.cmu.edu/~oelagama/Freshmen_Immigration/Theory_of_Computation.html
2. http://www.cs.utep.edu/vladik/cs5315.16/why.html