In [computer science](http://en.wikipedia.org/wiki/Computer_science), **functional programming** is a [programming paradigm](http://en.wikipedia.org/wiki/Programming_paradigm), a style of building the structure and elements of computer programs, that treats [computation](http://en.wikipedia.org/wiki/Computation) as the evaluation of [mathematical functions](http://en.wikipedia.org/wiki/Function_(mathematics)) and avoids [state](http://en.wikipedia.org/wiki/Program_state) and [mutable](http://en.wikipedia.org/wiki/Immutable_object) data. Functional programming emphasizes [functions](http://en.wikipedia.org/wiki/Function_(computer_science)) that produce results that depend only on their inputs and not on the program state—i.e. [pure](http://en.wikipedia.org/wiki/Pure_function) [mathematical functions](http://en.wikipedia.org/wiki/Function_(mathematics)). It is a [declarative programming](http://en.wikipedia.org/wiki/Declarative_programming) paradigm, which means programming is done with [expressions](http://en.wikipedia.org/wiki/Expression_(computer_science)). In functional code, the output value of a function depends only on the arguments that are input to the function, so calling a function *f* twice with the same value for an argument *x* will produce the same result *f(x)* both times. Eliminating [side effects](http://en.wikipedia.org/wiki/Side_effect_(computer_science)), i.e. changes in state that do not depend on the function inputs, can make it much easier to understand and predict the behavior of a program, which is one of the key motivations for the development of functional programming.