Introduction to Semantic Versioning

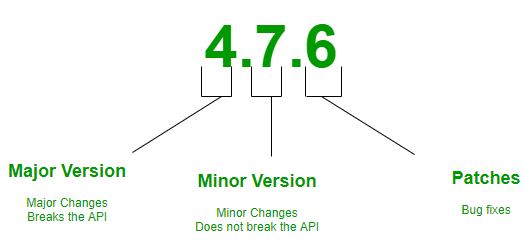
* Difficulty Level : [Easy](https://www.geeksforgeeks.org/easy)
* Last Updated : 25 Mar, 2020

Semantic versioning (also referred as **SemVer**) is a versioning system that has been on the rise over the last few years. It has always been a problem for software developers, release managers and consumers. Having a universal way of versioning the software development projects is the best way to track what is going on with the software as new plugins, addons, libraries and extensions are being built almost everyday.

Semantic Versioning is a 3-component number in the format of **X.Y.Z**, where :

* X stands for a major version.
* Y stands for a minor version.
* Z stands for a patch.

So, SemVer is of the form **Major.Minor.Patch**.



**Working :** The goal of SemVer was to bring some sanity to the management of rapidly moving software release targets. As discussed above, 3 numbers i.e, Major, Minor and Patch are required to identify a software version. For example, if we take version 5.12.2, then it has a major version of 5, a minor version of 12 and a patch version of 2. Below given are the scenarios when you should bump the value of X, Y and Z.

* Bump the value of **X** when breaking the existing API.
* Bump the value of **Y** when implementing new features in a backward-compatible way.
* Bump the value of **Z** when fixing bugs.

Valid identifiers are in the set **[A-Za-z0-9]** and cannot be empty. Pre-release metadata is identified by appending a hyphen to the end of the SemVer sequence. Thus a pre-release for version 1.0.0 could be 1.0.0-alpha.1. Then if another build is needed, it would become 1.0.0-alpha.2, and so on. Note that names cannot contain leading zeros, but hyphens are allowed in names for pre-release identifiers.

**Advantages of SemVer :**

* You can keep track of every transition in the software development phase.
* Versioning can do the job of explaining the developers about what type of changes have taken place and the possible updates that should take place in the software.
* It helps to keep things clean and meaningful.
* It helps other people who might be using your project as a dependency.

**Points to keep in mind :**

* The first version starts at 0.1.0 and not at 0.0.1, as no bug fixes have taken place, rather we start off with a set of features as first draft of the project.
* Before 1.0.0 is only the Development Phase, where you focus on getting stuff done.
* SemVer does not cover libraries tagged **0.\*.\***. The first stable version is 1.0.0.