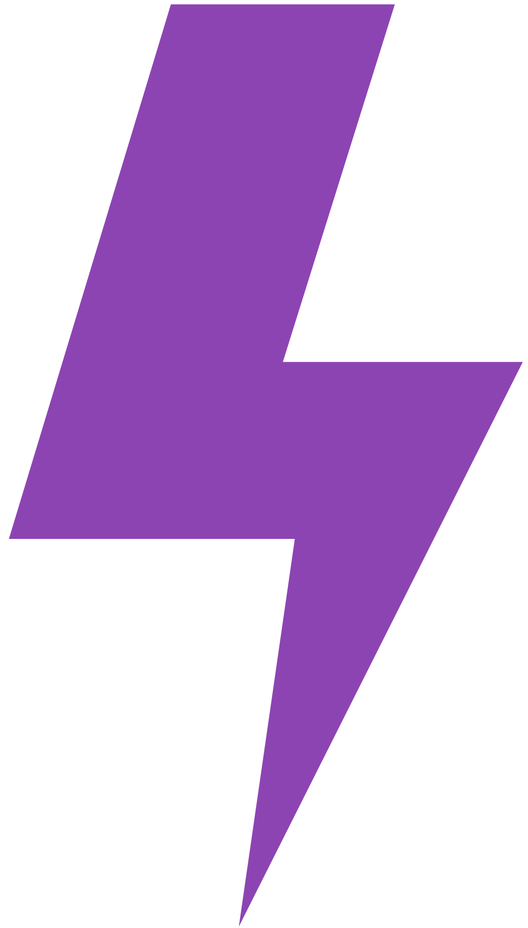
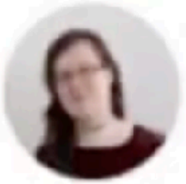
Comparing Version Numbers



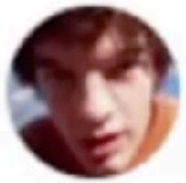


Leah Neukirchen
@LeahNeukirchen

You won't believe how much software breaks because Python 3.10 has a two-digit minor version.

4:43 PM · Sep 27, 2021 · Twitter Web App

152 Retweets **35** Quote Tweets **1,330** Likes



Tim Yates
@cimbuldev

Replying to @LeahNeukirchen

I used to use “compare two software version numbers” as an interview question. The results were horrifying.

6:11 PM · Sep 27, 2021 · Twitter for iPhone

5 Retweets **119** Likes

How would you compare them then?

Semantic Versioning

PyCon CZ 2023

```python
def compare_versions(v1, v2):
    """Compare two version strings (in format x.y.z)."""

    if v1 == v2:
        return RESULT.EQUAL

    v1t = [int(num) for num in v1.split('.')]
    v2t = [int(num) for num in v2.split('.')]

    for v1v, v2v in zip(v1t, v2t):
        if v1v > v2v:
            return v1
        elif v2v > v1v:
            return v2
```

Well ...

A pre-release version MAY be denoted by appending a hyphen and a series of dot separated identifiers immediately following the patch version.

1.5.2-rc.1

When major, minor, and patch are equal, a pre-release version has lower precedence than a normal version:

Example: 1.0.0-alpha < 1.0.0.

... and

Precedence for two pre-release versions with the same major, minor, and patch version MUST be determined by comparing each dot separated identifier from left to right until a difference is found as follows:

1. Identifiers consisting of only digits are compared numerically.

2. Identifiers with letters or hyphens are compared lexically in ASCII sort order.

3. Numeric identifiers always have lower precedence than non-numeric identifiers.

4. A larger set of pre-release fields has a higher precedence than a smaller set, if all of the preceding identifiers are equal.

Example: 1.0.0-alpha < 1.0.0-alpha.1 < 1.0.0-alpha.beta < 1.0.0-beta < 1.0.0-beta.2 < 1.0.0-beta.11 < 1.0.0-rc.1 < 1.0.0.

there could be build metadata

Build metadata MAY be denoted by appending a plus sign and a series of dot separated identifiers immediately following the patch or pre-release version. Identifiers MUST comprise only ASCII alphanumerics and hyphens [0-9A-Za-z-]. Identifiers MUST NOT be empty. Build metadata MUST be ignored when determining version precedence.

1.5.2-rc1+amd

At least it's well defined

```
^(0|[1-9]\d*)\.(0|[1-9]\d*)\.(0|[1-9]\d*)(?:-((?:0|[1-9]\d*|\d*[a-zA-Z-][0-9a-zA-Z-]*)(?:\.(?:0|[1-9]\d*|\d*[a-zA-Z-][0-9a-zA-Z-]*))*))?(?:\+([0-9a-zA-Z-]+(?:\.[0-9a-zA-Z-]+)*))?$
```

Not everyone uses SemVer

and not everything that looks like SemVer adheres to rules

Python

```
[N!]N(.N)*[{a|b|rc}N][.postN][.devN]
```

CalVer

for example
Ubuntu

2023.04

LaTeX

3.141592653

Idiosyncratic Version Number

Perl has two!

1.20 (one point two)

1.2 == 1.20

v1.20 (one, twenty)

v1.2 < v1.20

3.0

Vista

95

11

XP

Xbox

Xbox 360

Xbox One

Xbox Series X

Whatever this is with Microsoft

What I'm trying to say is

It's complicated

If you're asked this in **an interview**

Ask for a version definition

Don't try to come up with regex

Unit tests are lovely

If you need to do this in **real life**

Hope the versions follow a standard

Document it well

Good luck!

Thanks!

I'm Juhis

hamatti.org

Mastodon:
@hamatti@hamatti.org