InfoWorld

Visual Studio Code vs. Atom: How they stack up

The open-source, ultra-configurable code editors from Microsoft and GitHub have a lot in common. Here are the big differences you need to know

By Serdar Yegulalp
Senior Writer, InfoWorld

JUL 3, 2019

If you're a fan of Microsoft Visual Studio Code—and it seems more people are every day—it's because the popular code editor offers a heap of appealing features. It's endlessly customizable, highly consistent across platforms, and progressing at a rapid clip with monthly updates.

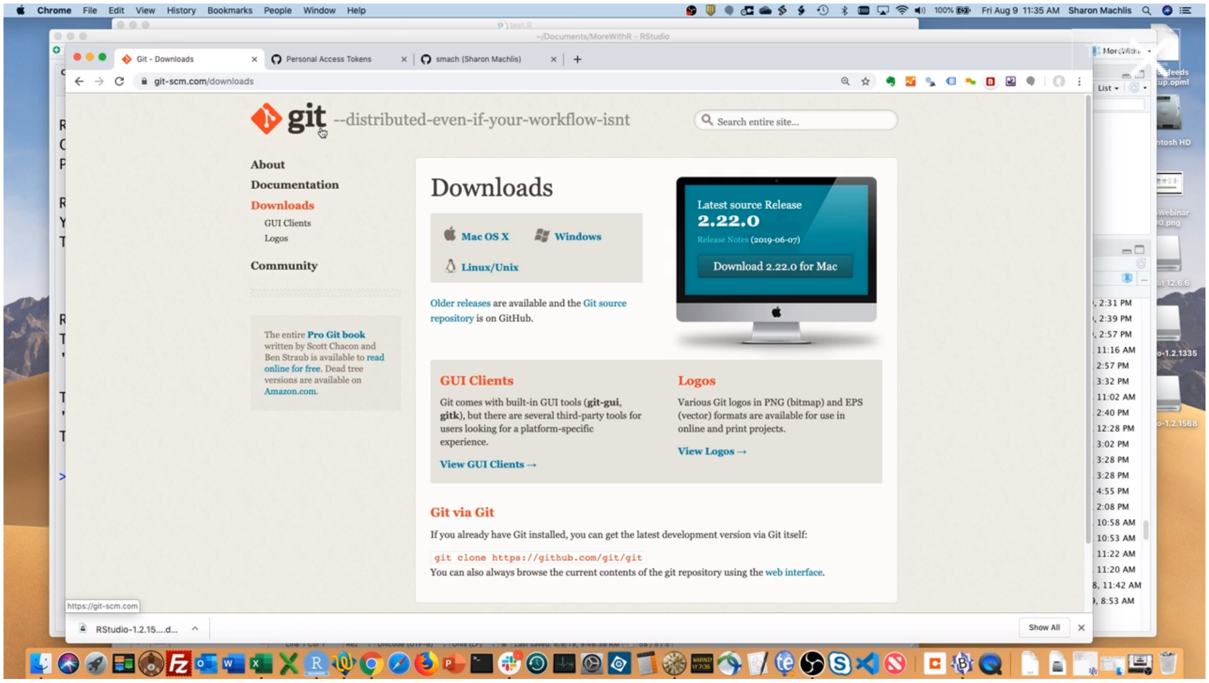
But Visual Studio Code is hardly the only popular code editor out there. In fact, the market is filled with highly customizable editing apps, not least of which is "hackable" Atom, a tool developed by GitHub that commands a faithful following of users. Both Visual Studio Code and Atom are built with similar components, mainly the Electron system for building desktop applications with web technologies.

[Using Visual Studio Code? Don't miss these 10 Visual Studio Code extensions for every developer. • Learn what's new in the latest version of Visual Studio Code. | Keep up with hot topics in programming with InfoWorld's App Dev Report newsletter.]

Trying to decide between Visual Studio Code and Atom ? Here are some of the key differences.

Visual Studio Code vs. Atom: Origins and development

Visual Studio Code and Atom have much in common. Both were built using GitHub's Electron framework for writing desktop apps using JavaScript and HTML and deploying them with the Node.js runtime. Atom began development at GitHub, debuting in 2014, while Visual Studio Code originated at Microsoft, appearing in 2015. And then Microsoft purchased GitHub in 2018.



Downloads

Older releases are available and the Git source repository is on GitHub.

The entire Pro Git book written by Scott Chacon and Ben Straub is available to read online for free. Dead tree versions are available on Amazon.com.

GUI Clients

Git comes with built-in GUI tools (**git-gui**, **gitk**), but there are several third-party tools for users looking for a platform-specific experience.

Logos

Various Git logos in PNG (bitmap) and EPS (vector) formats are available for use in online and print projects.

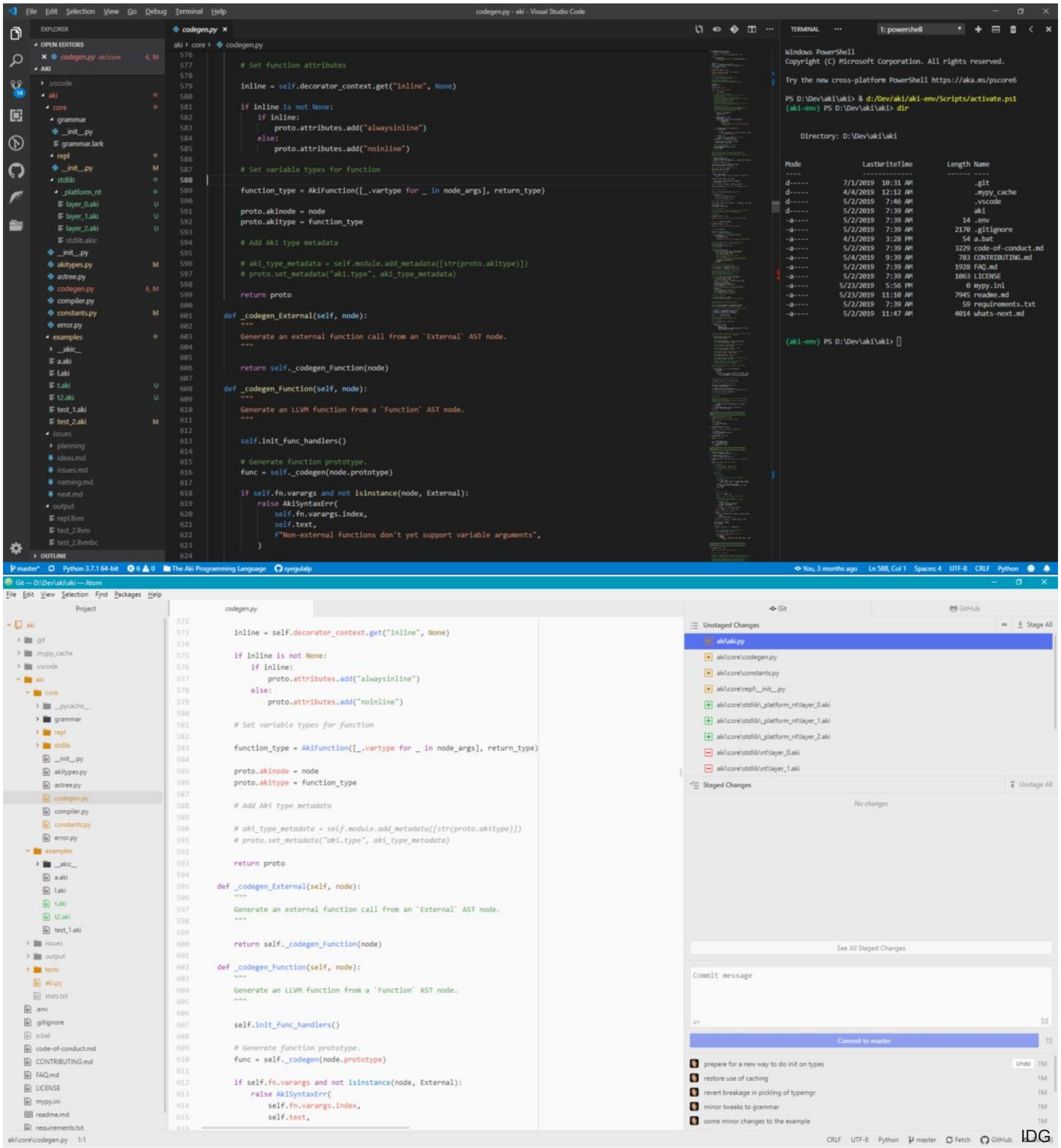
Git via Git

If you already have Git installed, you can get the latest development version via Git itself:

```
git clone https://github.com/git/git
```

You can also always browse the current contents of the git repository using the web interface.

Now that both Electron-based code editors belong to Microsoft, should we expect Atom to be deprecated over time? The short answer is “not yet, at least.” Development on Atom has continued apace by the same team, with new versions appearing regularly since the GitHub sale. And so far, Atom’s development track hasn’t been explicitly guided by Microsoft, making it a possible alternative for those who aren’t fond of Visual Studio Code’s more direct links to Redmond (e.g., silently sending usage telemetry).



```
        # Set function attributes

        inline = self.decorator_context.get("inline", None)

        if inline is not None:
            if inline:
                proto.attributes.add("alwaysinline")
            else:
                proto.attributes.add("noinline")

        # Set variable types for function

        function_type = AkiFunction([_.vartype for _ in node_args], return_type)

        proto.akinode = node
        proto.akitype = function_type

        # Add Aki type metadata

        # aki_type_metadata = self.module.add_metadata([str(proto.akitype)])
        # proto.set_metadata("aki.type", aki_type_metadata)

        return proto

    def _codegen_External(self, node):
        """
        Generate an external function call from an `External` AST node.
        """

        return self._codegen_Function(node)

    def _codegen_Function(self, node):
        """
        Generate an LLVM function from a `Function` AST node.
        """

        self.init_func_handlers()

        # Generate function prototype.
        func = self._codegen(node.prototype)

        if self.fn.varargs and not isinstance(node, External):
            raise AkiSyntaxErr(
                self.fn.varargs.index,
                self.text,
                f"Non-external functions don't yet support variable arguments",
            )
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS D:\Dev\aki\aki> & d:/Dev/aki/aki-env/Scripts/activate.ps1
(aki-env) PS D:\Dev\aki\aki> dir


    Directory: D:\Dev\aki\aki


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----         7/1/2019  10:31 AM                .git
d-----         4/4/2019  12:12 AM                .mypy_cache
d-----         5/2/2019   7:46 AM                .vscode
d-----         5/2/2019   7:39 AM                aki
-a----         5/2/2019   7:39 AM             14 .env
-a----         5/2/2019   7:39 AM           2170 .gitignore
-a----         4/1/2019   3:28 PM             54 a.bat
-a----         5/2/2019   7:39 AM           3229 code-of-conduct.md
-a----         5/4/2019   9:39 AM            783 CONTRIBUTING.md
-a----         5/2/2019   7:39 AM           1928 FAQ.md
-a----         5/2/2019   7:39 AM           1063 LICENSE
-a----        5/23/2019   5:56 PM              0 mypy.ini
-a----        5/23/2019  11:10 AM           7945 readme.md
-a----         5/2/2019   7:39 AM             59 requirements.txt
-a----         5/2/2019  11:47 AM           4014 whats-next.md


(aki-env) PS D:\Dev\aki\aki> []
```

```
        inline = self.decorator_context.get("inline", None)

        if inline is not None:
            if inline:
                proto.attributes.add("alwaysinline")
            else:
                proto.attributes.add("noinline")

        # Set variable types for function

        function_type = AkiFunction([_.vartype for _ in node_args], return_type)

        proto.akinode = node
        proto.akitype = function_type

        # Add Aki type metadata

        # aki_type_metadata = self.module.add_metadata([str(proto.akitype)])
        # proto.set_metadata("aki.type", aki_type_metadata)

        return proto

    def _codegen_External(self, node):
        """
        Generate an external function call from an `External` AST node.
        """

        return self._codegen_Function(node)

    def _codegen_Function(self, node):
        """
        Generate an LLVM function from a `Function` AST node.
        """

        self.init_func_handlers()

        # Generate function prototype.
        func = self._codegen(node.prototype)

        if self.fn.varargs and not isinstance(node, External):
            raise AkiSyntaxErr(
                self.fn.varargs.index,
                self.text,
```

Unstaged Changes

- aki\aki.py
- aki\core\codegen.py
- aki\core\constants.py
- aki\core\repl__init__.py
- aki\core\stdlib_platform_nt\layer_0.aki
- aki\core\stdlib_platform_nt\layer_1.aki
- aki\core\stdlib_platform_nt\layer_2.aki
- aki\core\stdlib\nt\layer_0.aki
- aki\core\stdlib\nt\layer_1.aki

Staged Changes

No changes

Commit message

prepare for a new way to do init on types	1M
restore use of caching	1M
revert breakage in pickling of typemgr	1M
minor tweaks to grammar	1M
some minor changes to the example	1M

Both Visual Studio Code and Atom use the Electron desktop application system, but each has different philosophies for which components are included by default and in what *form.*

Whether fallout from the Microsoft acquisition or not, Facebook's retiring its Nuclide project in late 2018 was definitely a blow to Atom. Nuclide was an open source extension for Atom that provided a suite of IDE-like facilities for developing projects using React Native, Hack, and Flow. On the plus side, parts of Nuclide are enjoying a second life in other editors—including, you guessed it, Visual Studio Code. (Note that third parties have also developed a "de-Microsofted" version of Visual Studio Code, VSCodium, free of Microsoft branding, telemetry, and licensing.)

Visual Studio Code vs. Atom: Customization and extensibility

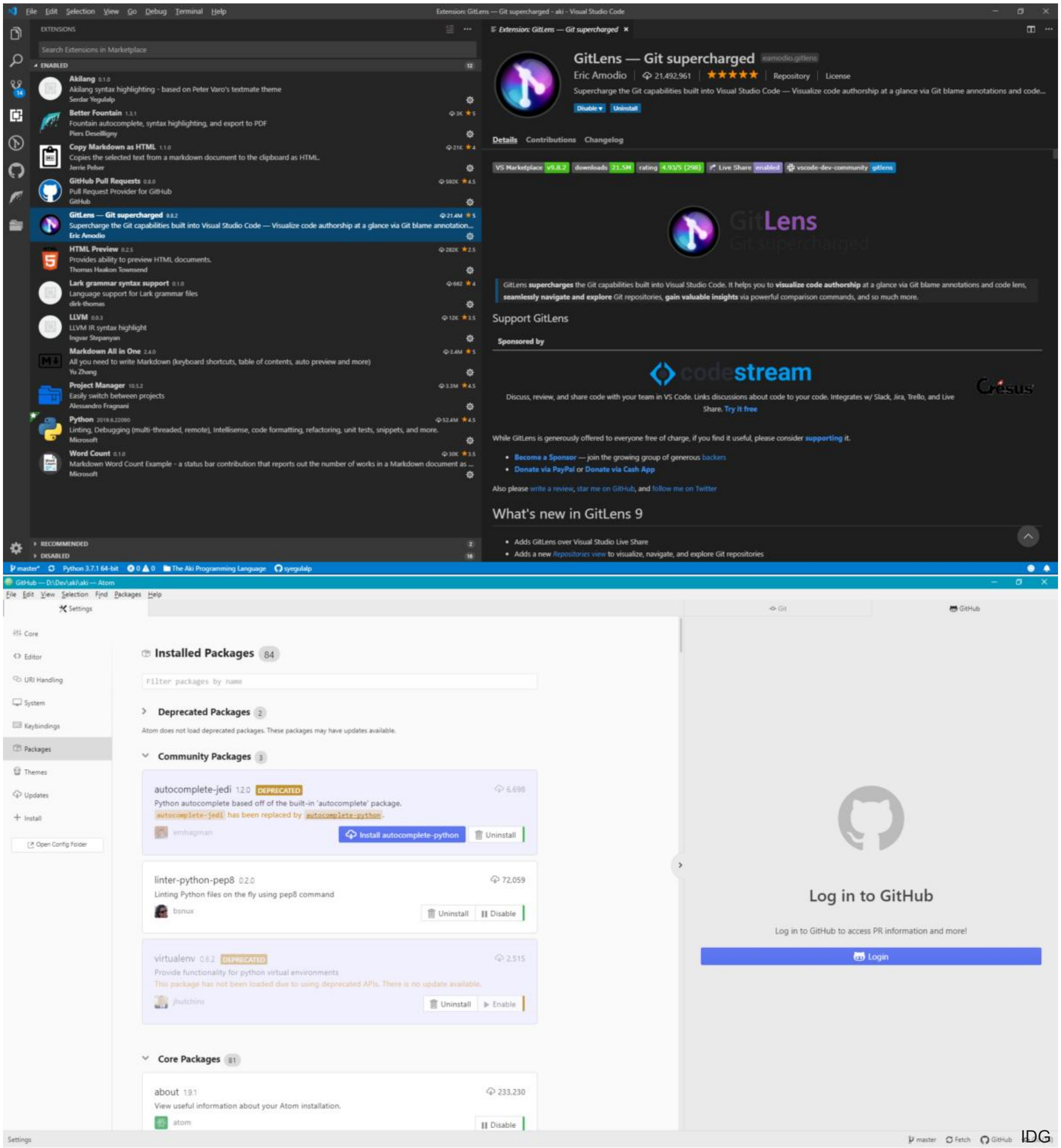
Both Atom and Visual Studio Code are designed to be customizable and extensible via third-party add-on packages. In this respect they're about even. Both have large and well-organized indexes of extensions and themes. Both allow you to search, install, and manage add-ons directly inside the program itself. One minor difference is themes. In Visual Studio Code, themes are considered an extension like any other. In Atom, themes are a different class of extension, managed in their own distinct part of the UI.

Another area where Atom differs is its hackability. Atom's online documentation has an entire section named, bluntly enough, Hacking Atom, which walks the prospective Atom hacker through many common customizations. Visual Studio Code has a guide to creating extensions, but nothing like the top-down hacker's tour Atom provides.

Visual Studio Code vs. Atom: Plug-ins and integration

Atom was designed to be highly hackable and user-configurable. To that end, many of Atom's core functions are provided as plug-ins. A default roster of plug-ins provided out of the box includes Git/GitHub integration and editing functions like working with whitespace and tabs.

Visual Studio Code, by contrast, builds more functionality directly in. For instance, some Git integration is available out of the box in Visual Studio Code as a native part of the editor. However, Visual Studio Code's native functionality can be extended or eclipsed with plug-ins. In fact, because Visual Studio Code's native Git integration is minimal, you'll need one of the third-party Git extensions like GitLens for more serious work.


File Edit Selection View Go Debug Terminal Help
Extension: GitLens — Git supercharged - aki - Visual Studio Code
EXTENSIONS
Search Extensions in Marketplace
ENABLED 12
RECOMMENDED 2
DISABLED 18
master* Python 3.7.1 64-bit 0 0 The Aki Programming Language syegulalp
Extension: GitLens — Git supercharged
Disable ▾ Uninstall
Details Contributions Changelog
VS Marketplace v9.8.2 downloads 21.5M rating 4.93/5 (298) Live Share enabled vscode-dev-community gitlens
GitLens Git supercharged
codestream
Crésus
GitHub — D:\Dev\aki\aki — Atom
File Edit View Selection Find Packages Help
Settings
Core
Editor
URI Handling
System
Keybindings
Packages
Themes
Updates
Install
Open Config Folder
Git
GitHub
Login
Settings
master Fetch GitHub

Akilang 0.1.0
Akilang syntax highlighting - based on Peter Varo's textmate theme
Serdar Yegulalp

Better Fountain 1.3.1 — 3K ★5
Fountain autocomplete, syntax highlighting, and export to PDF
Piers Deseilligny

Copy Markdown as HTML 1.1.0 — 21K ★4
Copies the selected text from a markdown document to the clipboard as HTML.
Jerrie Pelser

GitHub Pull Requests 0.8.0 — 592K ★4.5
Pull Request Provider for GitHub
GitHub

GitLens — Git supercharged 9.8.2 — 21.4M ★5
Supercharge the Git capabilities built into Visual Studio Code — Visualize code authorship at a glance via Git blame annotation...
Eric Amodio

HTML Preview 0.2.5 — 282K ★2.5
Provides ability to preview HTML documents.
Thomas Haakon Townsend

Lark grammar syntax support 0.1.0 — 662 ★4
Language support for Lark grammar files
dirk-thomas

LLVM 0.0.3 — 12K ★3.5
LLVM IR syntax highlight
Ingvar Stepanyan

Markdown All in One 2.4.0 — 3.4M ★5
All you need to write Markdown (keyboard shortcuts, table of contents, auto preview and more)
Yu Zhang

Project Manager 10.5.2 — 3.3M ★4.5
Easily switch between projects
Alessandro Fragnani

Python 2019.6.22090 — 52.4M ★4.5
Linting, Debugging (multi-threaded, remote), Intellisense, code formatting, refactoring, unit tests, snippets, and more.
Microsoft

Word Count 0.1.0 — 30K ★3.5
Markdown Word Count Example - a status bar contribution that reports out the number of works in a Markdown document as ...
Microsoft

GitLens — Git supercharged eamodio.gitlens
Eric Amodio | 21,492,961 | ★★★★★ | Repository | License
Supercharge the Git capabilities built into Visual Studio Code — Visualize code authorship at a glance via Git blame annotations and code...

GitLens **supercharges** the Git capabilities built into Visual Studio Code. It helps you to **visualize code authorship** at a glance via Git blame annotations and code lens, **seamlessly navigate and explore** Git repositories, **gain valuable insights** via powerful comparison commands, and so much more.

Support GitLens

Sponsored by

Discuss, review, and share code with your team in VS Code. Links discussions about code to your code. Integrates w/ Slack, Jira, Trello, and Live Share. Try it free

While GitLens is generously offered to everyone free of charge, if you find it useful, please consider **supporting** it.

- **Become a Sponsor** — join the growing group of generous backers
- **Donate via PayPal** or **Donate via Cash App**

Also please write a review, star me on GitHub, and follow me on Twitter

What's new in GitLens 9

- Adds GitLens over Visual Studio Live Share
- Adds a new *Repositories view* to visualize, navigate, and explore Git repositories

Installed Packages 84

Filter packages by name

Deprecated Packages 2

Atom does not load deprecated packages. These packages may have updates available.

Community Packages 3

autocomplete-jedi 1.2.0 DEPRECATED — 6,698
Python autocomplete based off of the built-in 'autocomplete' package.
`autocomplete-jedi` has been replaced by `autocomplete-python`.
emfsagman
Install autocomplete-python | Uninstall

linter-python-pep8 0.2.0 — 72,059
Linting Python files on the fly using pep8 command
bsnux
Uninstall | Disable

virtualenv 0.6.2 DEPRECATED — 2,515
Provide functionality for python virtual environments
This package has not been loaded due to using deprecated APIs. There is no update available.
jhutchins
Uninstall | Enable

Core Packages 81

about 1.9.1 — 233,230
View useful information about your Atom installation.
atom
Disable

Log in to GitHub

Log in to GitHub to access PR information and more!

IDG

With Visual Studio Code, the directory of extensions includes themes as well as language support and other tools. Atom keeps themes distinct from extensions that add language *support or that modify the editor's other behaviors.*

Visual Studio Code vs. Atom: Usage and market share

Ever since it first appeared, Visual Studio Code has eaten away at the marketshare of many other editors, Atom included. According to Triplebyte, by the end of 2018 Visual Studio Code was used by 22% of the candidate developers it interviewed over the course of the year; Atom, 6%. Those numbers had grown from around 5% and 11%, respectively, in 2017.

Don't take this as gospel that Atom is on its way out, though. Atom's design, development process, and feature mix appeal to an audience all its own. But the rise of Visual Studio Code isn't due to Microsoft's backing alone—it's because Visual Studio Code is a genuinely powerful, flexible, and useful tool.

Serdar Yegulalp is a senior writer at InfoWorld, focused on machine learning, containerization, devops, the Python ecosystem, and periodic reviews.