OrdinaryIndustries

The Ultimate VS Code Setup for Python

 Jack Fields Follow 10 min read · Jul 28, 2023

Having a well-optimized setup for your Visual Studio Code (VS Code) is nothing short of a game-changer when it comes to unleashing your coding potential. Imagine an environment tailored precisely to your needs, where every keystroke brings forth a symphony of efficiency and creativity. A good setup not only streamlines your workflow but also boosts productivity, making the coding journey an exhilarating and enjoyable experience. Whether you're a seasoned developer or just starting your coding adventure, setting up VS Code right can be the secret ingredient that takes your skills to the next level. So, let's dive in and explore the art of crafting a sublime coding haven within the virtual walls of Visual Studio Code!

Find more of my writing on Medium and Substack! You can also find me on Twitter @ordinaryinds.

A programmer in front of screens. Created with ChatGPT.

Why VS Code?

VS Code has become one of the most popular code editors for Python development, and for several good reasons. Here are some compelling arguments for using VS Code in Python development:

- Lightweight code editor that launches quickly, making it ideal for developers who want a seamless coding experience without resource-heavy overhead.

- Available on multiple platforms, including Windows, macOS, and Linux.
- Boasts an extensive extension ecosystem.
- Built-in debugger that works seamlessly with Python.
- Supports various Python formatters and linters.
- Integrates well with version control systems like Git.
- Built-in terminal.
- Intuitive user interface.
- Vast and active community of developers.

In this article we will explore installing VSCode and then customize it to be powerhouse for Python development.

Installation

Installing VS Code is simple. Download and run the installer from the official website here. Check out our complete guide here for system-specific instructions.

How to Edit Settings

Now, let's take a look at how settings are modified. This can be different from programs you have used previously.

In VS Code, there are two primary options for editing settings: the Graphical User Interface (GUI) and the `settings.json` file. Each option offers unique benefits to cater to different preferences and use cases.

Settings.json File:

The `settings.json` file is a plain text JSON file that stores all the configuration settings for Visual Studio Code. It is the underlying configuration file that the GUI settings modify. You can skip the GUI and directly edit this file instead.

To edit settings via the `settings.json` file:

1. Open the Command Palette by pressing `Ctrl + Shift + P` (Windows/Linux) or `Cmd + Shift + P` (Mac).
2. Type “User Settings” and select the JSON option from the command list.

Remember the command palette as we will use it quite a bit throughout this guide.

Graphical User Interface (GUI):

The GUI method allows users to modify settings through the VS Code interface without directly interacting with the underlying JSON configuration file.

To access the GUI settings click the gear icon in the lower-left corner of the VS Code window. If you're a keyboard shortcut fan then use the shortcut `Ctrl + ,` (Windows/Linux) or `Cmd + ,` (Mac).

This guide uses the `settings.json` method but each item can be found in the GUI by searching the key we provide. For example, to find rulers in the GUI you can search `editor.rulers`.

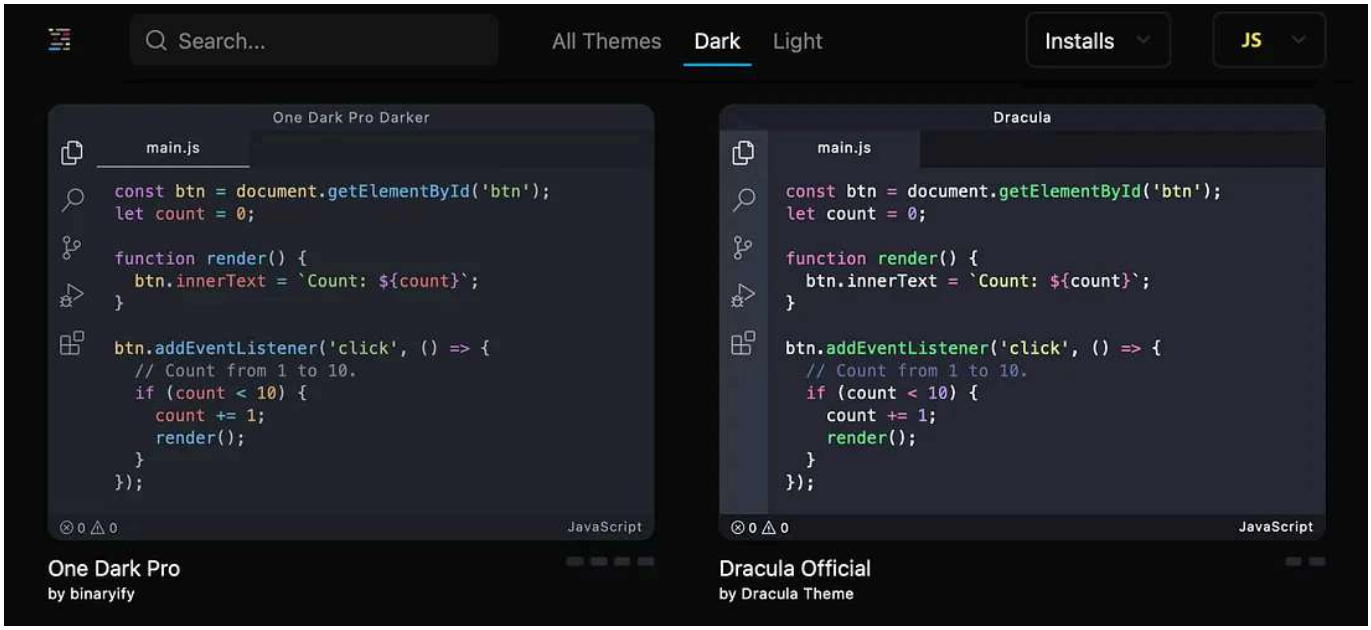
Themes

In this section we look at themes and how to adjust the visual styling of VS Code. Choosing a visual theme for VS Code may seem like it's merely about making your editor look pretty. In truth, it is an important decision that significantly impacts the overall development experience and productivity. Here are several reasons why selecting a suitable visual theme is crucial:

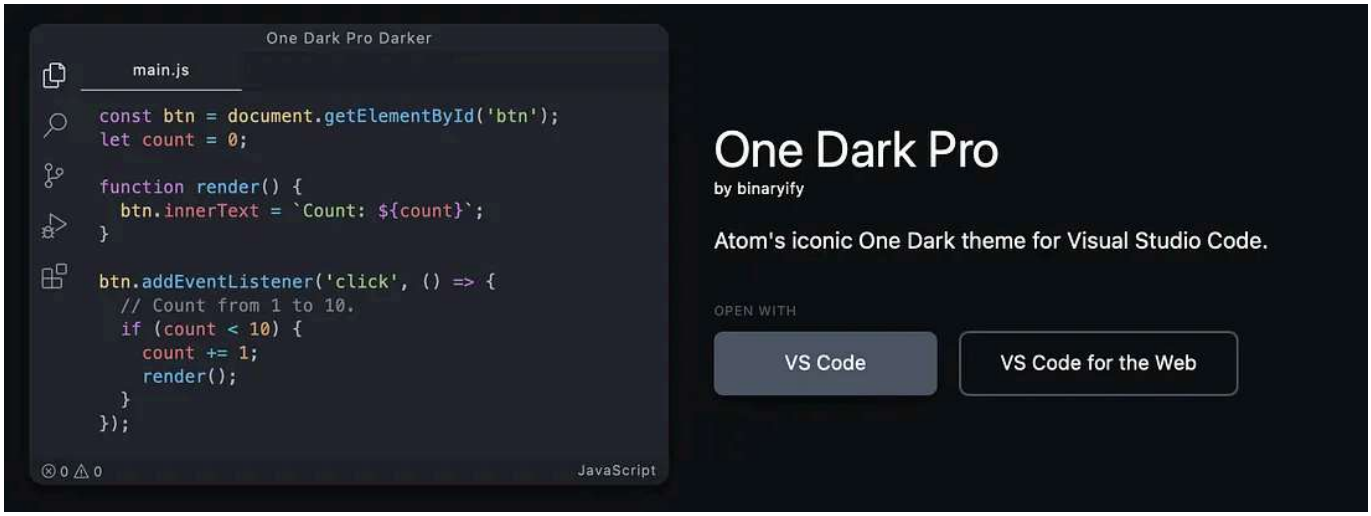
1. Reduced Eye Strain: A well-designed visual theme with appropriate color choices, contrast, and syntax highlighting can reduce eye strain during long coding sessions.

2. Improved Code Readability: Visual themes play a critical role in enhancing code readability. Themes that offer clear differentiation between various code elements such as keywords, functions, variables, and comments make it easier to comprehend code at a glance.
3. Personalization and Motivation: Visual themes allow developers to create an environment that resonates with their personality and preferences. A visually appealing and personalized coding space can boost motivation and inspire creativity, making coding sessions more engaging and enjoyable.
4. Accessibility and Inclusivity: Choosing an accessible theme that considers color contrast and readability is essential for ensuring inclusivity. A theme that accommodates diverse needs, including those of developers with visual impairments or color blindness, fosters an inclusive coding environment.

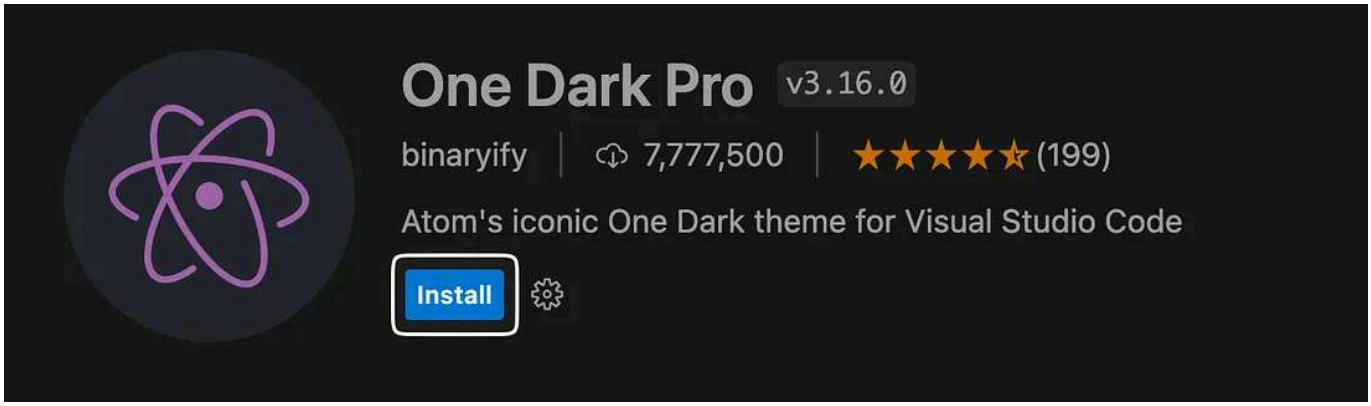
Browsing themes in VS Code can be frustrating at times because there isn't a gallery that you can quickly scan. Instead, you have to find each theme and open it in the extension marketplace. We found VS Code Themes which provides a beautiful way to quickly browse through the various color themes you can choose from.



We like One Dark Pro. However it can't be stated enough how important it is for you to find a theme that fits what YOUR needs.



Once you choose a theme you can select it, then choose the VS Code button. This opens the theme extension in the extension marketplace.

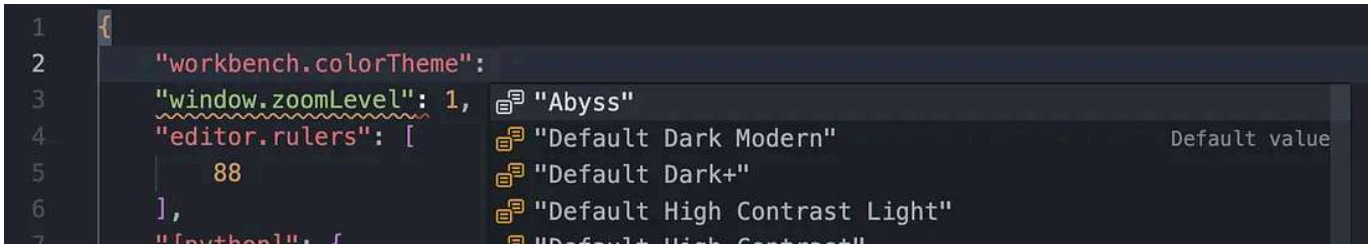


From the marketplace page select install. Once installed, the theme is automatically enabled.

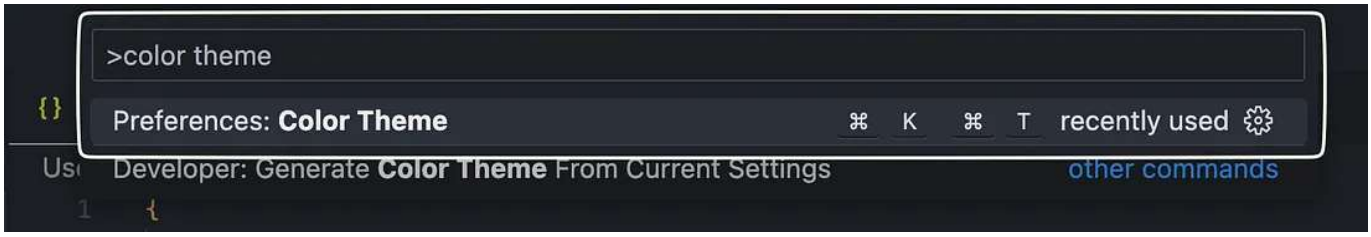
Some themes offer multiple options. Here you can see One Dark Pro has 4 different color themes to choose from.

```
{
    "workbench.colorTheme": "One Dark Pro Darker"
}
```

You can also set this in `settings.json` .



Add the `workbench.colorTheme` key and VS Code will display a list of themes to choose from. If you want to choose a different theme that you have installed you can do so from the command palette.



Open the command palette and type “color theme”. Choose the Preferences: Color Theme option. Of course, you can also do this in `settings.json` .

We have a few small tweaks we recommend to make your editor a bit more digestible. Feel free to use any that you find useful. These options

```
{
    "window.zoomLevel": 1,
    "workbench.tree.indent": 20,
    "editor.minimap.renderCharacters": false,
    "editor.minimap.size": "fill",
    "files.autoSave": "afterDelay",
    "files.autoSaveDelay": 1000,
    "editor.cursorStyle": "block-outline",
    "editor.cursorBlinking": "smooth"
}
```

`window.zoomLevel` sets the UI scale small enough to get a lot of content in but large enough to be legible. You can zoom in and out using `Cmd +` and `Cmd -` for macOS or `Ctl +` and `Ctl -` on Windows and Linux.

`workbench.tree.indent` adds more horizontal inset to the file explorer tree. This makes it easier to see nested content.

`editor.minimap.renderCharacters` makes the minimap show lines of shapes instead of actual text. Let's be real, no one is reading the minimap text.

`editor.minimap.size` adjust the sizing of the mini map so that it's usable but doesn't take up a ton of space.

`files.autoSave` will automatically save files you have modified recently. You can set it to save when changing tabs, when changing windows, or after a delay. If you choose the delay option you have to set `files.autoSaveDelay` to a number of seconds.

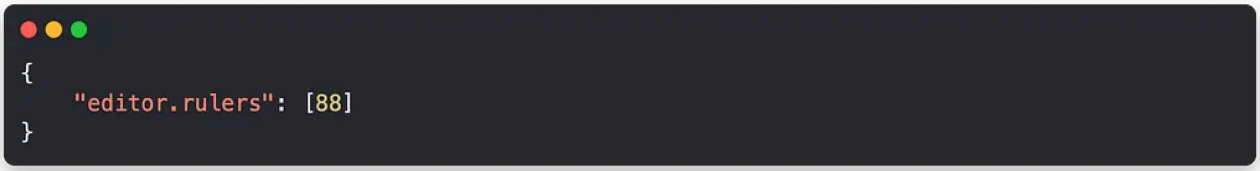
`editor.cursorStyle` and `editor.cursorBlinking` control the type of cursor you use. We prefer the old-school block and block-outline options.

Rulers

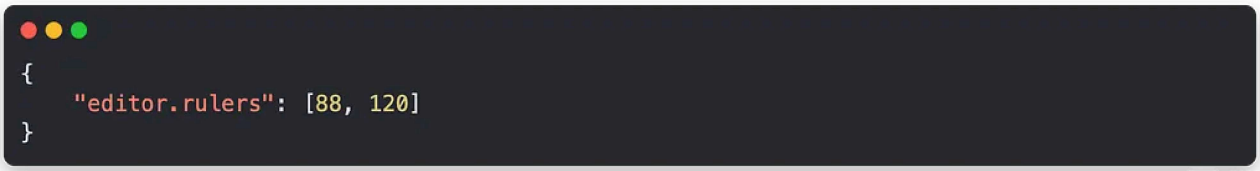
Rulers act as subtle guides that prevent code from sprawling too far, encouraging conciseness and readability. Embracing the power of rulers in your VS Code environment empowers you to write clean, well-organized code that not only meets industry standards but also becomes a joy to work with for both you and your fellow developers. So, let your code gracefully

dance along the rulers, and witness how this seemingly small addition can make a significant impact on the overall elegance and professionalism of your projects.

Adding a ruler to VS Code is a straightforward process. Open your `settings.json` and add the following line:



Here, we used 88 as this aligns with the Black formatter we will discuss later in this article. If your linter or formatter prefers a different line length then use that value. If you would like multiple rulers you can do that too:



The result is a vertical line at the chosen line length.

```python
from django.contrib.auth.models import AbstractUser, BaseUserManager
from django.db import models


class User(AbstractUser):
    """User base model."""

    class Role(models.TextChoices):
        ADMIN = "ADMIN", "Admin"
        WRITER = "WRITER", "Writer"

    base_role = Role.WRITER

    role = models.CharField(max_length=50, choices=Role.choices)

```

Linting

A linter is an essential tool that plays a vital role in improving the overall quality and maintainability of your code. It helps you adhere to the Python coding standards and best practices by analyzing your code for potential errors, style inconsistencies, and syntax issues. By highlighting these problems in real-time as you write your code, a linter assists you in identifying and rectifying mistakes early in the development process, reducing the chances of introducing bugs and enhancing the readability of your codebase. Using a linter not only fosters cleaner and more standardized coding practices within your team but also contributes to more efficient collaboration, making it an indispensable tool for any Django developer working with Python in the VS Code environment.

The number of options for linters is ever-growing. Here are a few of the most popular:

- Flake8
- PyLint

- Ruff

We are partial to Ruff. It is up to 100x faster than other linters, supports the latest version of Python, and has almost identical rules as Flake8. Of course, if you have a preference toward another linter use it. Much of the configuration will be similar.

Setting up Ruff couldn't be easier. Simply install the Ruff extension from the marketplace here . It starts working immediately. Regardless of the linter you choose add the following to your Python scope in `settings.json`.

These two options will enable linting when you save a file. Through our testing we found a few config options specific to Ruff that make us even more productive. Let's take a look at these.

Import auto-sorting

The PEP8 style guide details how imports should be organized. This helps keep your code both readable and maintainable. Auto-sorting will structure your Python imports first by their source (e.g. standard library, third-party, local application) then alphabetically within those sections.

Enable it by adding the `editor.codeActionsOnSave` section to your Python scope in `settings.json` .

```
{
    "[python]": {
        "editor.codeActionsOnSave": {
            "source.organizeImports": true
        }
    }
}
```

Here you can see an example of import sorting before:

```
import os
from blog.models import Post
from django.db import models
from django.contrib.auth.models import AbstractUser, BaseUserManager
```

and after:

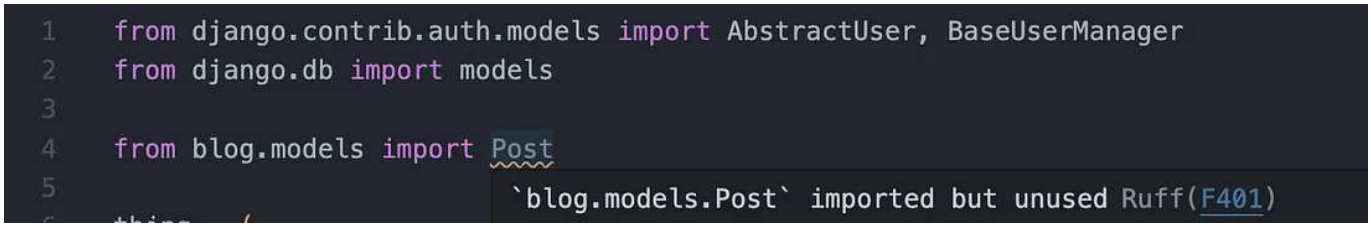
```
import os

from django.contrib.auth.models import AbstractUser, BaseUserManager
from django.db import models

from blog.models import Post
```

Auto-fixing Violations

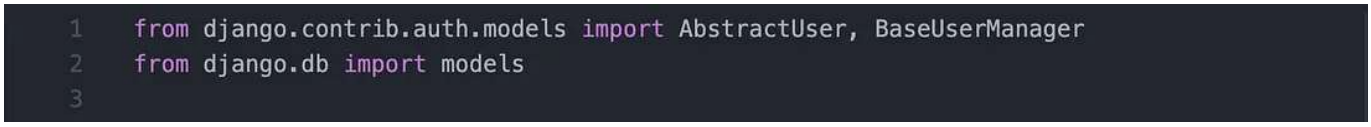
Finding violations in your code is only good if you fix them. You must take action on each one. Ruff offers a way to automatically fix violations. Enable it by adding the `source.fixAll` key to the `editor.codeActionsOnSave` section of your Python scope.

```
{
   "[python]": {
      "editor.codeActionsOnSave": {
         "source.fixAll": true
      }
   }
}
```

Here you can see an example of auto-fixing violations. Ruff identifies that the `Post` class is imported but not being used.



Ruff fixes this by removing the `Post` import.



Formatting

A code formatter is a tool that analyzes and automatically restructures source code according to predefined coding styles and conventions. It aims to achieve consistency in code layout, indentation, line breaks, and other formatting aspects without altering the code's functionality. By enforcing consistent code styles across a project, formatters enhance code readability, make it easier to understand, and minimize potential bugs caused by inconsistent formatting.

Much like linters there are many formatters to choose from. Here are a few of the most popular:

- Black
- YAPF
- autopep8

We prefer the style choices made by Black. Black's approach follows the principle of "the uncompromising code formatter," and it gains widespread recognition for its ability to produce consistent code that is highly readable.

As an example, Black uses an 88 character line length. This is in contrast to other formatters that use 80 characters. The decision to use a 10% longer limit makes a huge difference in file length. It also allows for less line wrapping which produces more readable code. Small but impactful choices like this make Black an easy choice for our development. As always, choose the formatter that aligns with your needs.

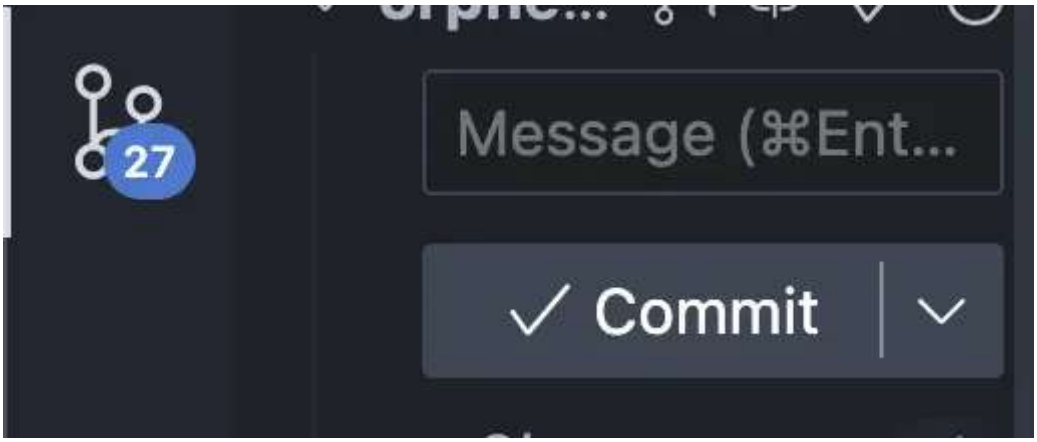
The Black extension can be found here. Install it as you would any other extension. Once installed, open `settings.json` and add these two options to your python scope.

```
"[python]": {
    "editor.defaultFormatter": "ms-python.black-formatter",
    "editor.formatOnSave": true
}
```

These lines set the default formatter to Black and enable format on save. This means that Black will format your code only when you save a file.

Git

VS Code integrations remarkably well with version control systems like git. Simply having git installed is enough for the editor to do it's magic. If you need to install git checkout this guide provided by git themselves. Once setup you will have a source control button your primary side bar.



The source control button the primary sidebar.

Conclusion

You now have a fully configured VS Code with a gorgeous color theme, linting, formatting, and installed a few helpful extensions to make your coding hours more productive and enjoyable! By leveraging the power of Visual Studio Code and its versatile configurations, developers can unlock

their full potential and take their Python development to new heights. As the development landscape continues to evolve, the adaptability and feature-rich nature of VS Code will undoubtedly keep it at the forefront of modern coding environments, empowering developers to write clean, efficient, and robust Python applications with ease.

If you liked this content and want to see more like it follow us on Twitter @OrdinaryInds.

Ordinary Industries

Vscode Development Python Coding Software Engineering

Published in OrdinaryIndustries

9 followers · Last published Jul 28, 2023

Follow

OI posts about software engineering, programming,

Written by Jack Fields

201 followers · 12 following

Follow

Formerly Apple. Indie app dev head of Ordinary Industries.

Responses (11)

Write a response

What are your thoughts?

der andere Dirk
Apr 30, 2024

Source code in screen shots. Rly?

4 Reply

Michael Young
Aug 8, 2023

Handy tips here. I would suggest making the suggested settings available as code blocks to make them easier to copy and paste those settings.

2 2 replies Reply

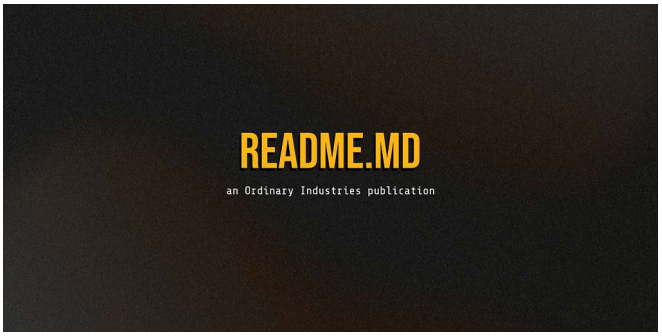
Fardin Delfani
Apr 19

hi

Reply

See all responses

More from Jack Fields and OrdinaryIndustries



In Kernel Extension by Jack Fields

Kernel Extension: Issue #1

A new kind of iOS dev newsletter that breaks out of the main thread

Jan 31

In Kernel Extension by Jack Fields

README.md

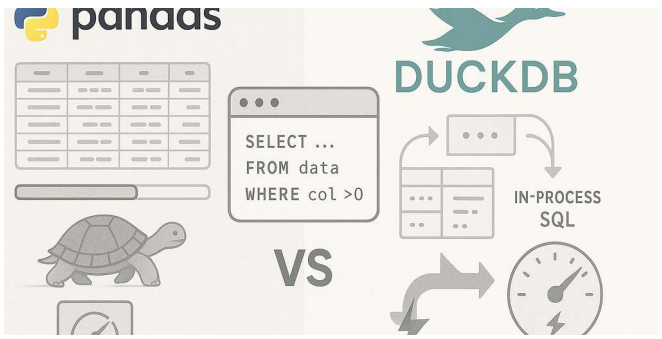
Kernel Extension (Kext) is a publication for iOS developers who want more than just...

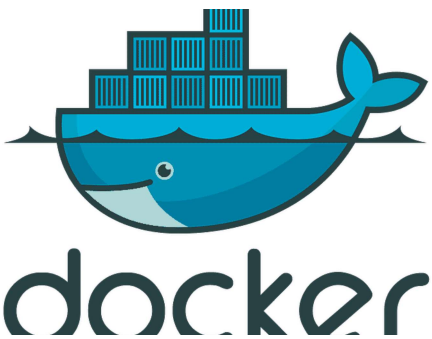
Feb 16

See all from Jack Fields

See all from OrdinaryIndustries

Recommended from Medium





Nikulsinh Rajput

Why DuckDB Replaced Pandas for All My Data Projects

Blazing-fast in-process SQL that scales without setup.

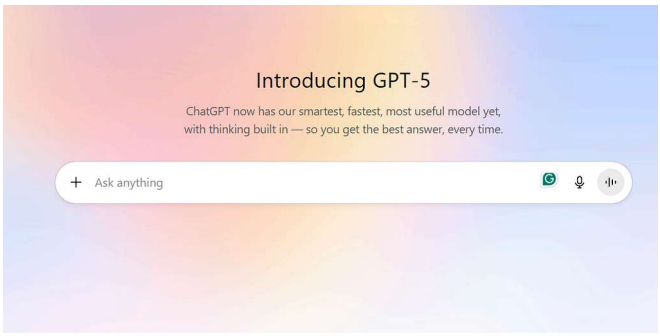
Aug 3 605 18

Abhinav

Docker Is Dead—And It's About Time

Docker changed the game when it launched in 2013, making containers accessible and...

Jun 8 4.7K 111



Mayuresh K

Mastering Git Worktree: A Developer's Guide to Multiple...

Have you ever been in the middle of implementing a complex feature when a...

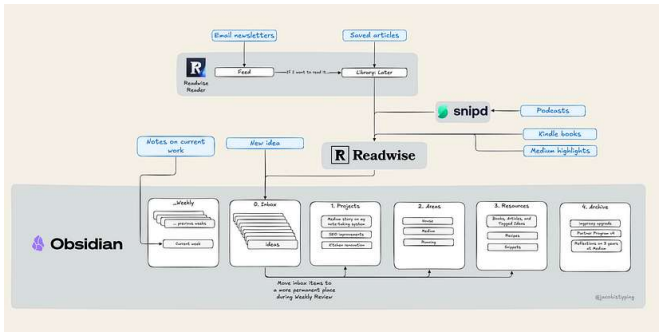
Mar 7 14 1

Alberto Romero

GPT-5 Is Here: There's Only One Feature Worth Writing About

My short review of OpenAI's new flagship model

5d ago 1.6K 61



Jacob Bennett

How I never forget anything as a staff software engineer

The engineer's over-engineered knowledge management system

ThreadSafe Diaries

He Was a Senior Developer, Until We Read His Pull Request

When experience doesn't translate to expertise, and how one code review change...

Jul 31 1.8K 32

Aug 3 5.6K 179

See more recommendations