Chapter 3

IDE

3.1 What Are IDEs and Code Editors?

An IDE (or Integrated Development Environment) is a program dedicated to software development. As the name implies, IDEs integrate several tools specifically designed for software development. These tools usually include:

- An editor designed to handle code (with, for example, syntax highlighting and auto-completion)
- Build, execution, and debugging tools
- Some form of source control

Most IDEs support many different programming languages and contain many more features. They can, therefore, be large and take time to download and install. You may also need advanced knowledge to use them properly.

In contrast, a dedicated code editor can be as simple as a text editor with syntax highlighting and code formatting capabilities. Most good code editors can execute code and control a debugger. The very best ones interact with source control systems as well. Compared to an IDE, a good dedicated code editor is usually smaller and quicker, but often less feature rich.

3.2 Requirements for a Good Python Coding Environment

So what things do we really need in a coding environment? Feature lists vary from app to app, but there are a core set of features that makes coding easier:

- Save and reload code files
  If an IDE or editor won't let you save your work and reopen everything later, in the same state it was in when you left, it's not much of an IDE.
- Run code from within the environment
  Similarly, if you have to drop out of the editor to run your Python code, then it's not much more than a simple text editor.
- Debugging support
  Being able to step through your code as it runs is a core feature of all IDEs and most good code editors.
- Syntax highlighting
  Being able to quickly spot keywords, variables, and symbols in your code makes reading and understanding code much easier.

- Automatic code formatting
  Any editor or IDE worth it's salt will recognize the colon at the end of a while or for statement, and know the next line should be indented.

Of course, there are lots of other features you might want, like source code control, an extension model, build and test tools, language help, and so on. But the above list is what I'd see as " core features" that a good editing environment should support.

With these features in mind, let's take a look at some general purpose tools we can use for Python development.

3.3 General Editors and IDEs with Python Support

3.3.1 Eclipse + PyDev

Category: IDE
Website: www.eclipse.org
Python tools: PyDev, www.pydev.org

If you've spent any amount of time in the open-source community, you've heard about Eclipse. Available for Linux, Windows, and OS X at, Eclipse is the de-facto open-source IDE for Java development. It has a rich marketplace of extensions and add-ons, which makes Eclipse useful for a wide range of development activities.

One such extension is PyDev, which enables Python debugging, code completion, and an interactive Python console. Installing PyDev into Eclipse is easy: from Eclipse, select Help, Eclipse Marketplace, then search for PyDev. Click Install and restart Eclipse if necessary.

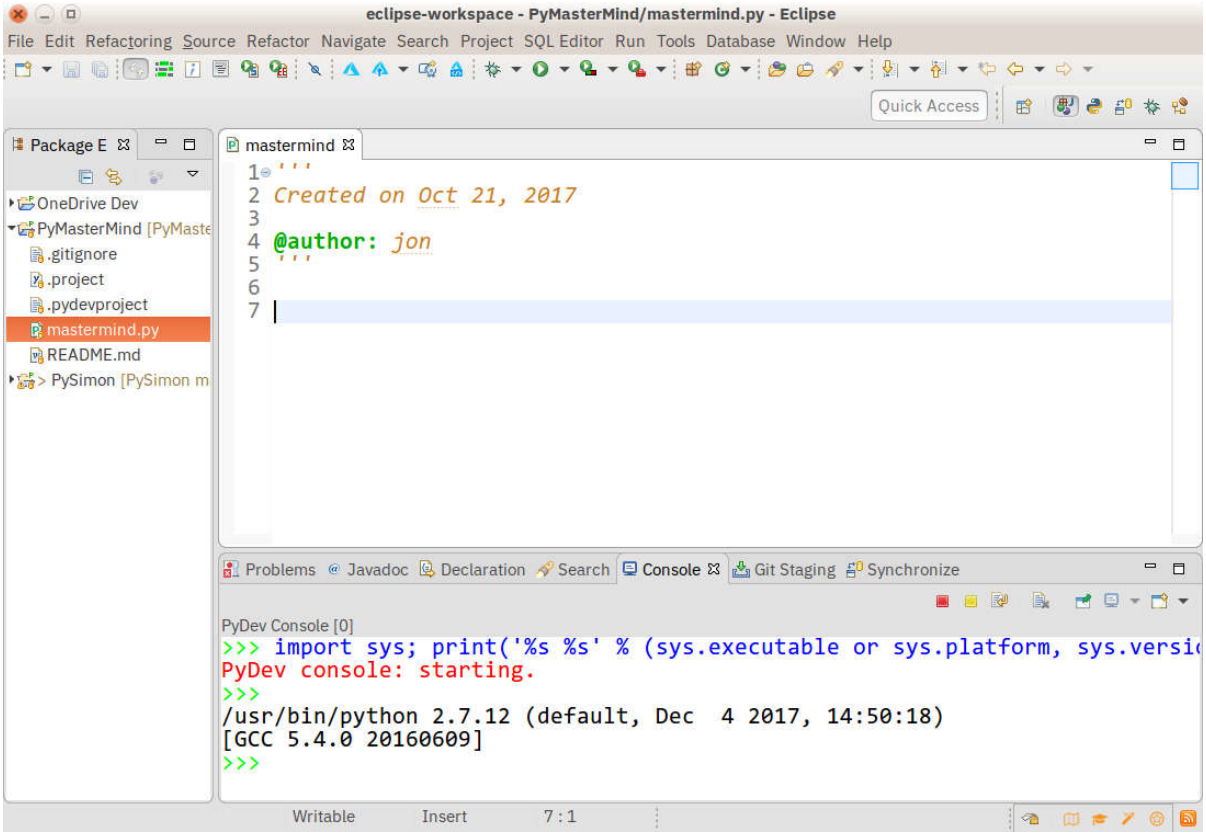

Figure 3.1: Eclipse + PyDev

Pros: If you've already got Eclipse installed, adding PyDev will be quicker and easier. PyDev is very accessible for the experienced Eclipse developer.

Cons: If you're just starting out with Python, or with software development in general, Eclipse can be a lot to handle. Remember when I said IDEs are larger and require more knowledge to use properly? Eclipse is all that and a bag of (micro)chips.

3.3.2 Sublime Text

Category: Code Editor
Website: http://www.sublimetext.com

Written by a Google engineer with a dream for a better text editor, Sublime Text is an extremely popular code editor. Supported on all platforms, Sublime Text has built-in support for Python code editing and a rich set of extensions (called packages) that extend the syntax and editing features.

Installing additional Python packages can be tricky: all Sublime Text packages are written in Python itself, and installing community packages often requires you to execute Python scripts directly in Sublime Text.

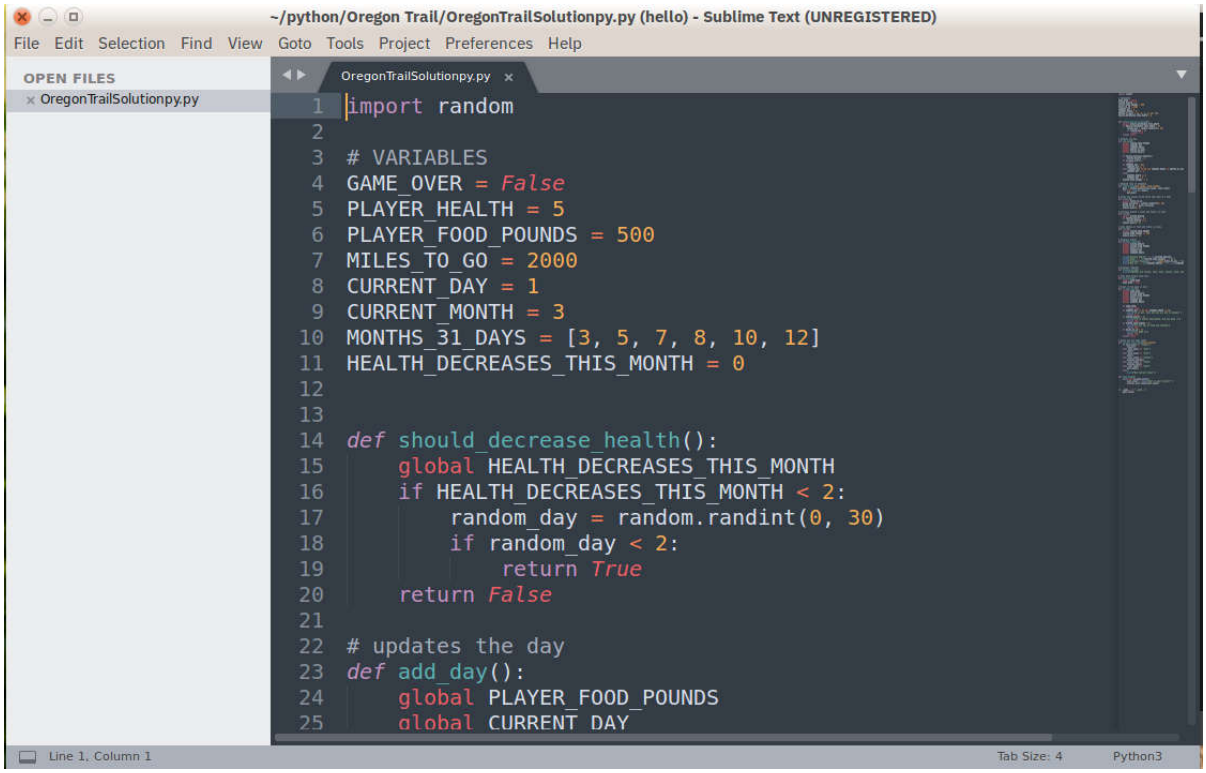

Figure 3.2: Sublime Text

Pros: Sublime Text has a great following in the community. As a code editor, alone, Sublime Text is fast, small, and well supported.

Cons: Sublime Text is not free, although you can use the evaluation version for an indefinite period of time. Installing extensions can be tricky, and there's no direct support for executing or debugging code from within the editor.

3.3.3 Atom

Category: Code Editor
Website: https://atom.io/

Available on all platforms, Atom is billed as the "hackable text editor for the 21st Century." With a sleek interface, file system browser, and marketplace for extensions, open-source Atom is built using Electron, a framework for creating desktop applications using JavaScript, HTML, and CSS. Python language support is provided by an extension that can installed when Atom is running.

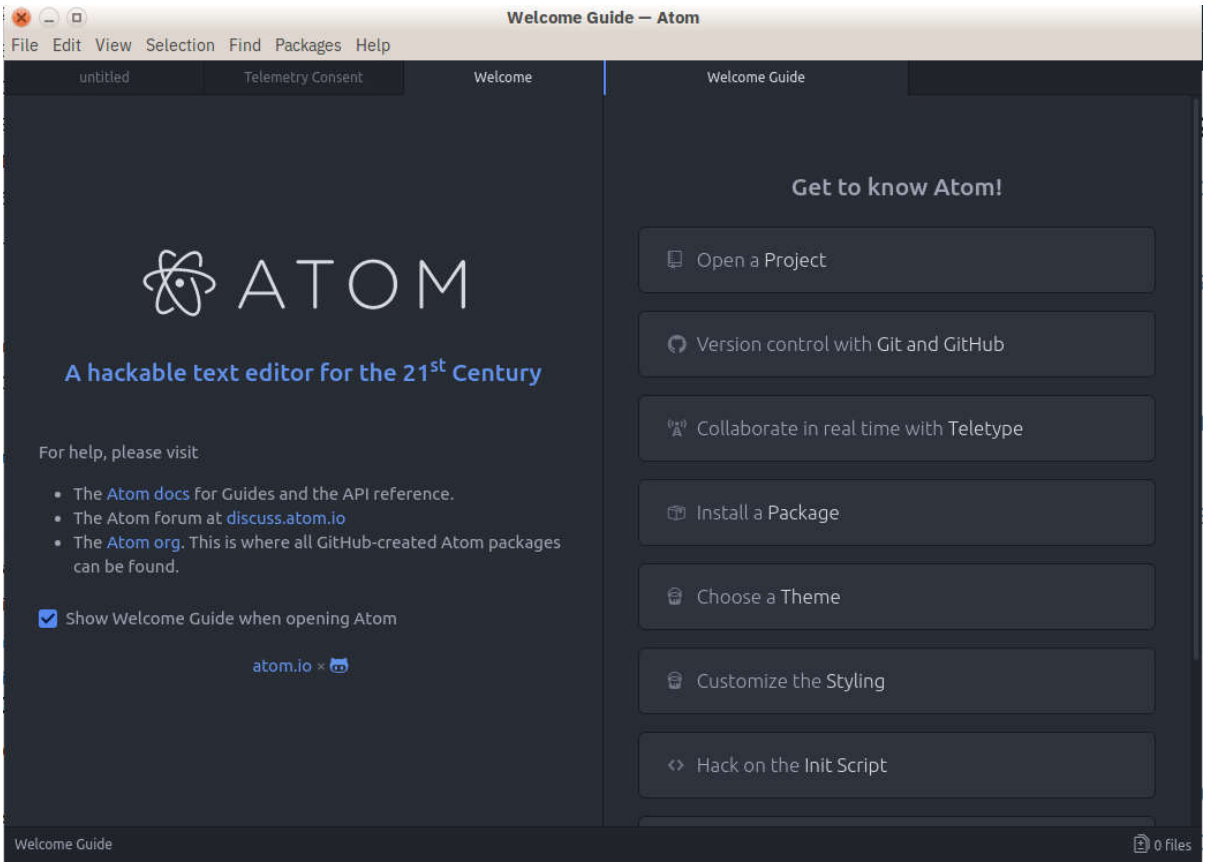

Figure 3.3: Atom

Pros: It has broad support on all platforms, thanks to Electron. Atom is small, so it downloads and loads fast.

Cons: Build and debugging support aren't built-in but are community provided add-ons. Because Atom is built on Electron, it's always running in a JavaScript process and not as a native application.

3.3.4 GNU Emacs

Category: Code Editor
Website: https://www.gnu.org/software/emacs/

Back before the iPhone vs Android war, before the Linux vs Windows war, even before the PC vs Mac war, there was the Editor War, with GNU Emacs as one of the combatants. Billed as " the extensible, customizable, self-documenting, real-time display editor," GNU Emacs has been around almost as long as UNIX and has a fervent following.

Always free and available on every platform (in one form or another), GNU Emacs uses a form of the powerful Lisp programming language for customization, and various customization scripts exist for Python development.

Pros: You know Emacs, you use Emacs, you love Emacs. Lisp is a second language, and you know the power it gives you means you can do anything.

Cons: Customization means writing (or copy/pasting) Lisp code into various script files. If it's not already provided, you may have to learn Lisp to figure out how to do it.

Plus, you know that Emacs would be a great operating system, if it only had a good text editor...

3.3.5 Vi / Vim

Category: Code Editor
Website: https://www.vim.org/

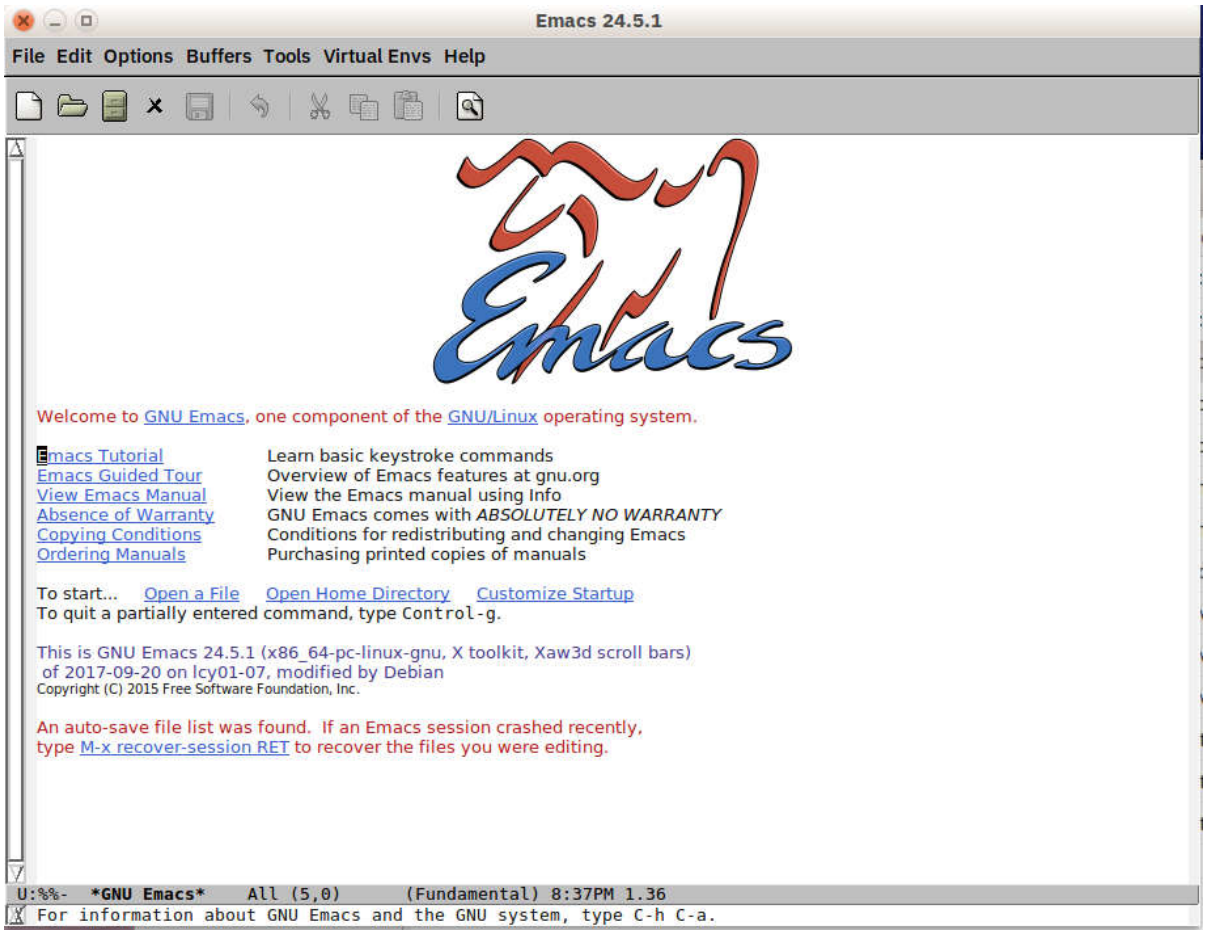

Figure 3.4: GNU Emacs

On the other side of the Text Editor War stands VI (aka VIM). Included by default on almost every UNIX system and Mac OS X, VI has an equally fervent following.

VI and VIM are modal editors, separating the viewing of a file from the editing of a file. VIM includes many improvements on the original VI, including an extensibility model and in-place code building. VIMScripts are available for various Python development tasks.

Pros: You know VI, you use VI, you love VI. VIMScripts don't scare you, and you know you bend it to your will.

Cons: Like Emacs, you're not comfortable finding or writing your own scripts to enable Python development, and you're not sure how a modal editor is supposed to work.

Plus, you know that VI would be a great text editor, if only it had a decent operating system.

3.3.6 Visual Studio

Category: IDE
Website: https://www.visualstudio.com/vs/
Python tools: Python Tools for Visual Studio, aka PTVS

Built by Microsoft, Visual Studio is a full-featured IDE, in many ways comparable to Eclipse. Built for Windows and Mac OS only, VS comes in both free (Community) and paid (Professional and Enterprise) versions. Visual Studio enables development for a variety of platforms and comes with its own marketplace for extensions.

Python Tools for Visual Studio (aka PTVS) enables Python coding in Visual Studio, as well as Intellisense for Python, debugging, and other tools.

Pros: If you already have Visual Studio installed for other development activities, adding PTVS is quicker and easier.

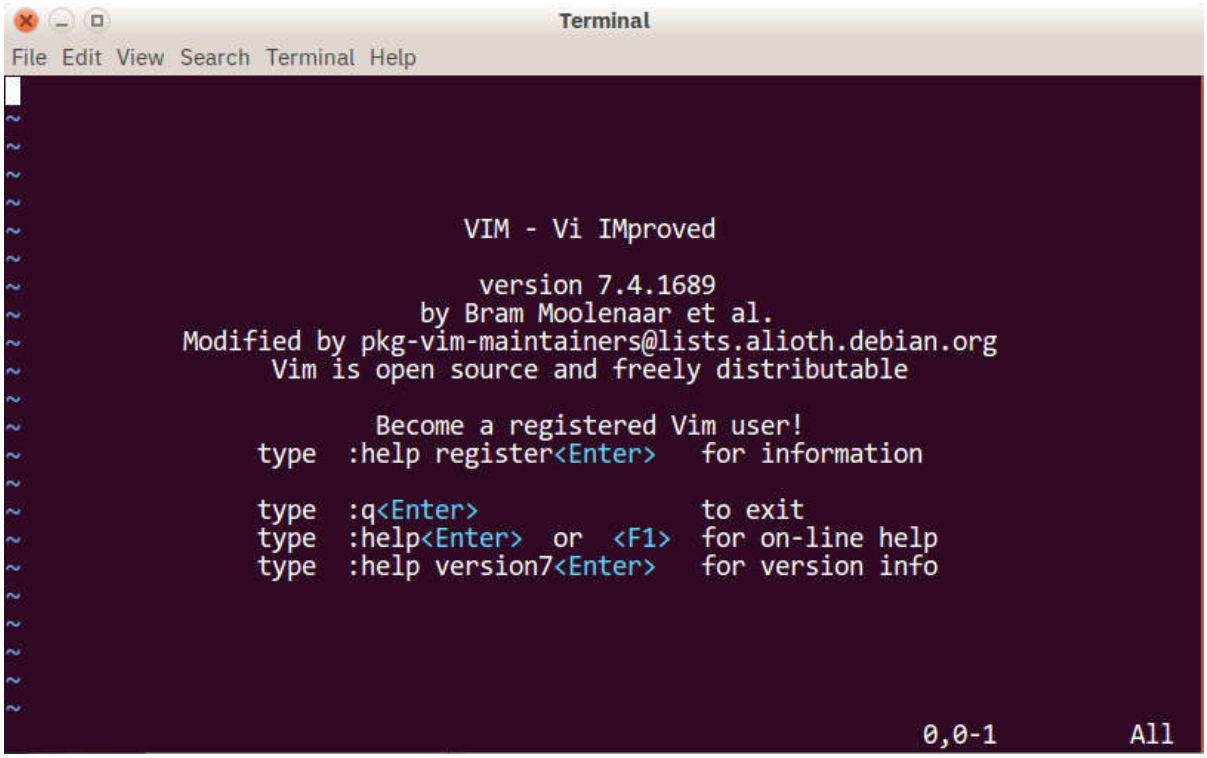

Figure 3.5: Vi / Vim

Cons: Visual Studio is a big download for just Python. Plus, if you're on Linux, you're out of luck: there's no Visual Studio install for that platform.

3.3.7 Visual Studio Code

Category: Code Editor
Website: https://code.visualstudio.com/
Python tools: https://marketplace.visualstudio.com/items?itemName=ms-python.python

Not to be confused with full Visual Studio, Visual Studio Code (aka VS Code) is a full-featured code editor available for Linux, Mac OS X, and Windows platforms. Small and light-weight, but full-featured, VS Code is open-source, extensible, and configurable for almost any task. Like Atom, VS Code is built on Electron, so it has the same advantages and disadvantages that brings.

Installing Python support in VS Code is very accessible: the Marketplace is a quick button click away. Search for Python, click Install, and restart if necessary. VS Code will recognize your Python installation and libraries automatically.

Pros: Thanks to Electron, VS Code is available on every platform, surprisingly full-featured despite having a small footprint, and open-source.

Cons: Electron means VS Code is not a native app. Plus, some people may have principled reasons to not use Microsoft resources.

3.4 Python-Specific Editors and IDEs

3.4.1 PyCharm

Category: IDE
Website: https://www.jetbrains.com/pycharm/

One of the best (and only) full-featured, dedicated IDEs for Python is PyCharm. Available in both paid (Professional) and free open-source (Community) editions, PyCharm installs quickly and easily on Windows, Mac OS X, and Linux platforms.

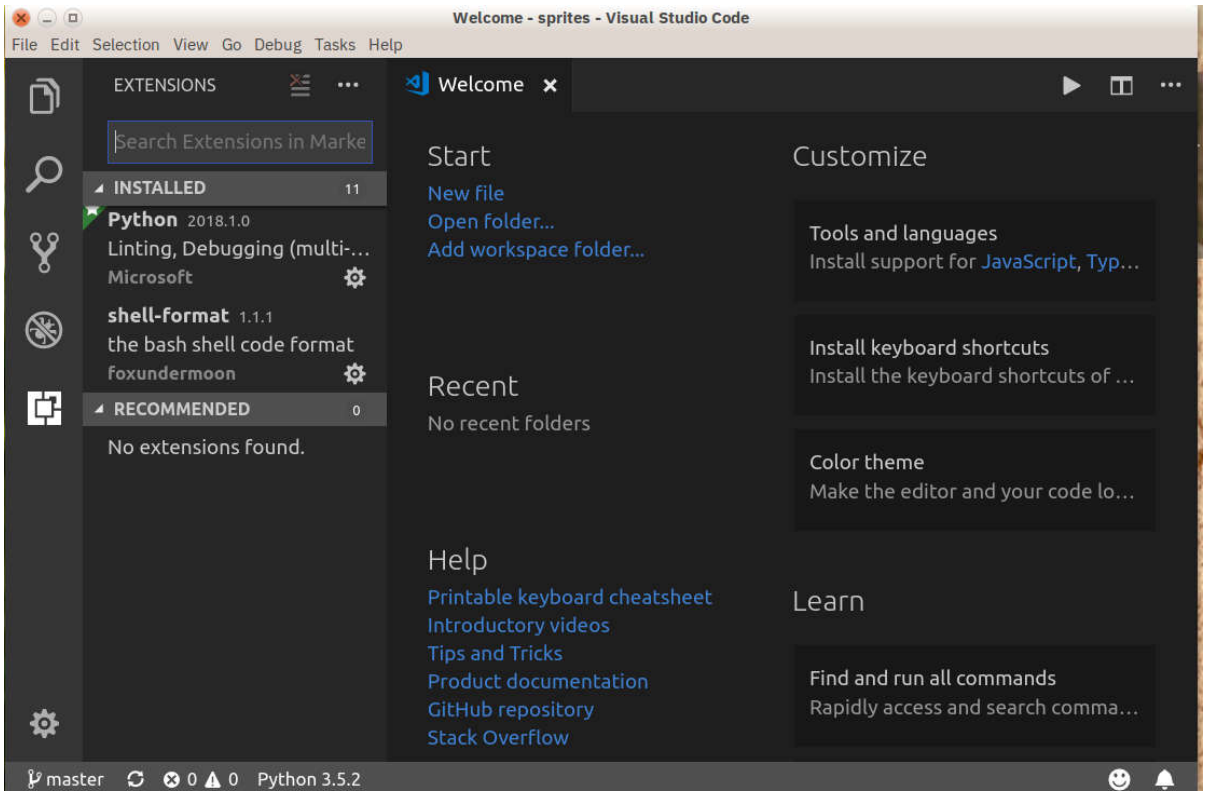

Figure 3.6: Visual Studio

Out of the box, PyCharm supports Python development directly. You can just open a new file and start writing code. You can run and debug Python directly inside PyCharm, and it has support for source control and projects.

Pros: It's the de facto Python IDE environment, with tons of support and a supportive community. It edits, runs, and debugs Python out of the box.

Cons: PyCharm can be slow to load, and the default settings may need tweaking for existing projects.

3.4.2 Spyder

Category: IDE
Website: https://github.com/spyder-ide/spyder

Spyder is an open-source Python IDE that's optimized for data science workflows. Spyder comes included with the Anaconda package manager distribution, so depending on your setup you may already have it installed on your machine.

What's interesting about Spyder is that it's target audience is data scientists using Python. You'll notice this throughout. For example, Spyder integrates well with common Python data science libraries like SciPy, NumPy, and Matplotlib.

Spyder features most of the " common IDE features" you might expect, such as a code editor with robust syntax highlighting, Python code completion, and even an integrated documentation browser.

A special feature that I haven't seen in other Python editing environments is Spyder's " variable explorer" that allows you to display data using a table-based layout right inside your IDE. Personally, I usually don't have a need for this but it does look neat. If you regularly do data science work using Python, you might fall in love with this unique feature. The IPython/Jupyter integration is nice as well.

Overall, I'd say that Spyder feels more basic than other IDEs. I like to view it more as a special purpose tool rather than something I use as my primary editing environment every day. What is nice about this Python IDE is that it is available for free on Windows, macOS, and Linux and that it is fully open-source software.

Pros: You're a data scientist using the Anaconda Python distribution.

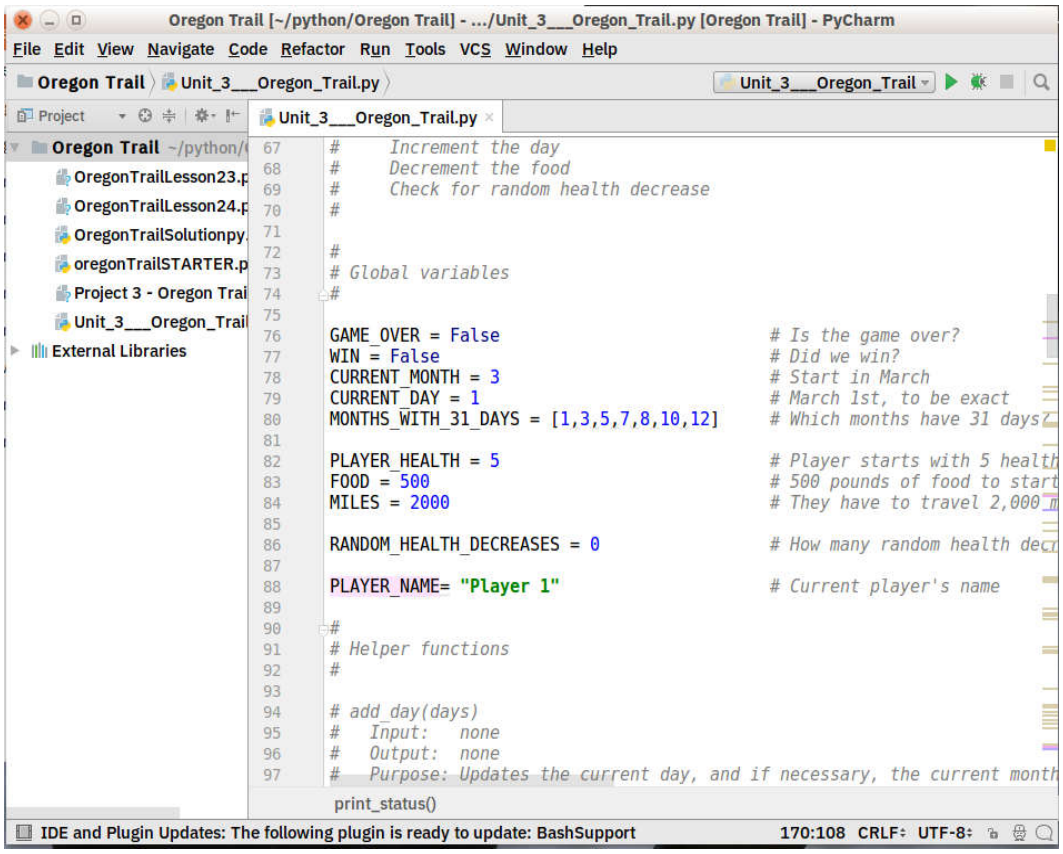

Figure 3.7: PyCharm

Cons: More experienced Python developers might find Spyder too basic to work with on a daily basis and instead opt for a more complete IDE or customized editor solution.

3.4.3 Thonny

Category: IDE
Website: http://thonny.org/

A recent addition to the Python IDE family, Thonny is billed as an IDE for beginners. Written and maintained by the Institute of Computer Science at the University of Tartu in Estonia, Thonny is available for all major platforms, with installation instructions on the site.

By default, Thonny installs with its own bundled version of Python, so you don't need to install anything else new. More experienced users may need to tweak this setting so already installed libraries are found and used.

Pros: You're a beginning Python user, and want an IDE that's ready to roll.

Cons: More experienced Python developers will find Thonny too basic for most uses, and the built-in interpreter is something to work around, not with. Plus, as a new tool, there may be issues you find which may not have immediate solutions.

3.5 Which Python IDE is Right for You?

Only you can decide that, but here are some basic recommendations:

- New Python developers should try solutions with as few customizations as possible. The less gets in the way, the better.
- If you use text editors for other tasks (like web pages or documentation), look for code editor solutions.

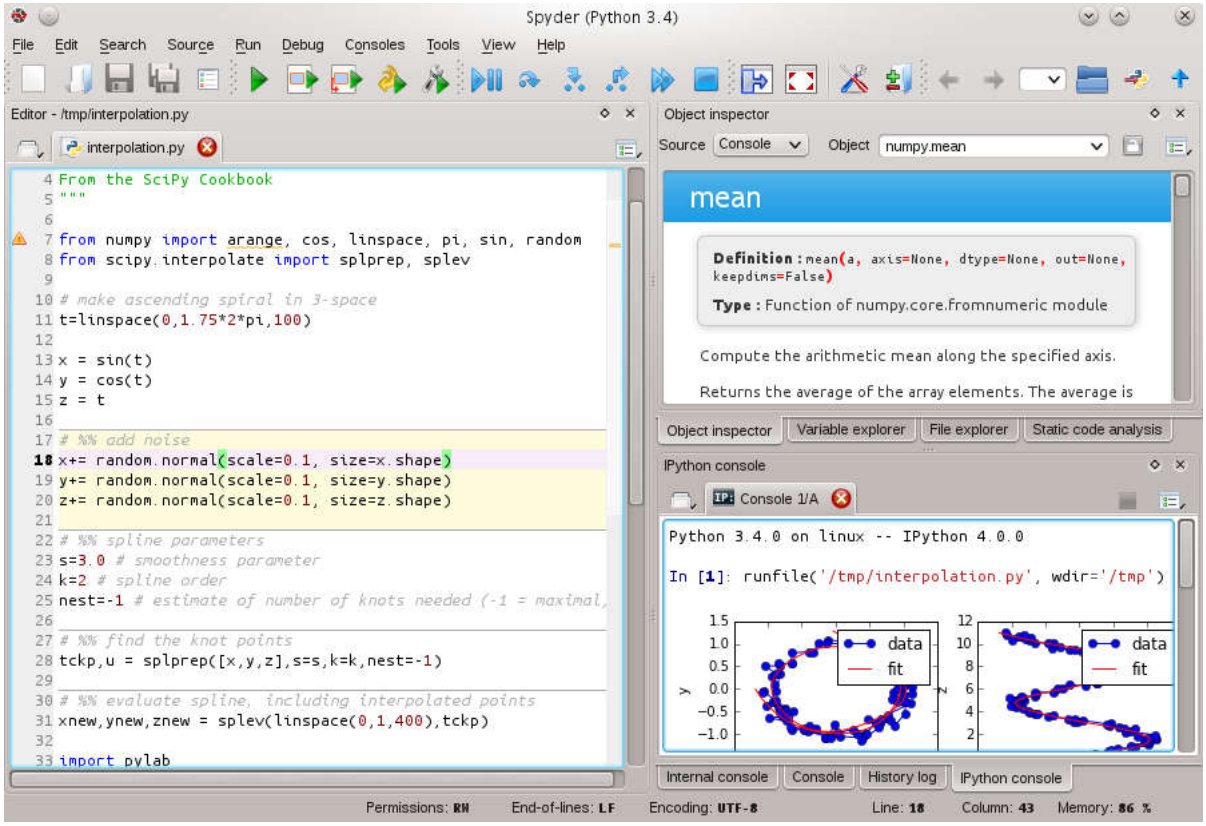

Figure 3.8: Spyder

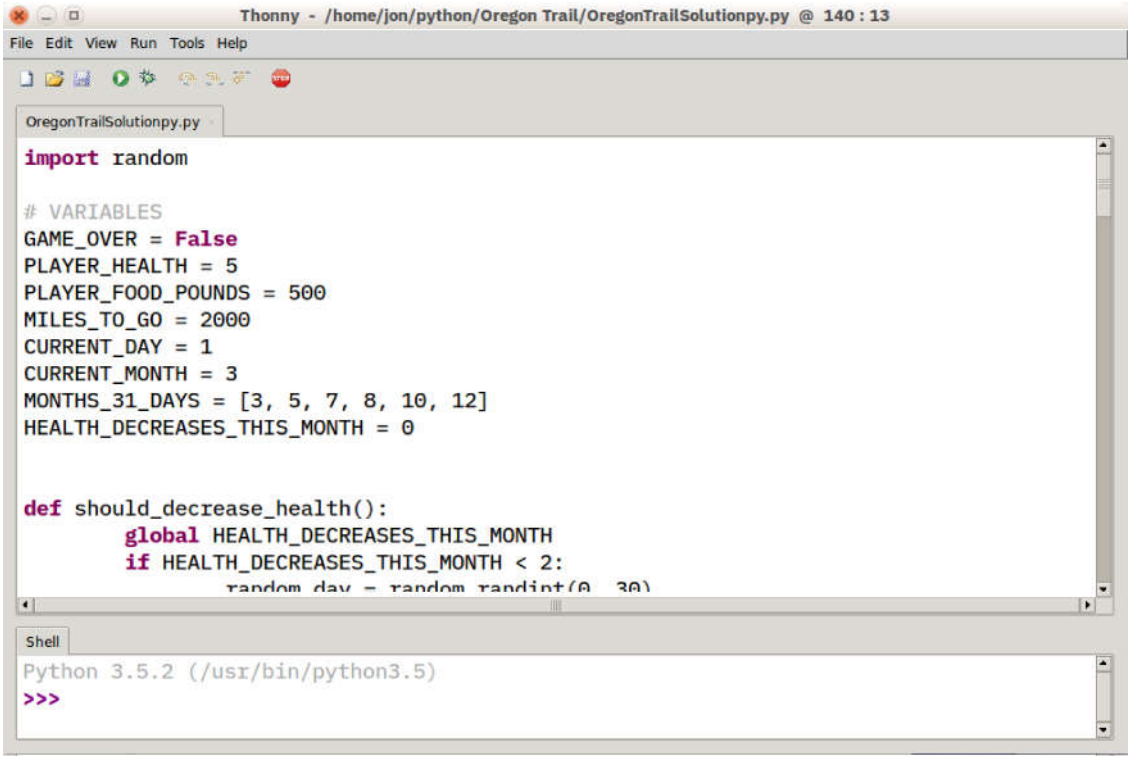

Figure 3.9: Thonny

- If you're already developing other software, you may find it easier to add Python capabilities to your existing toolset.