Python IDEs and Code Editors (Guide)

Table of Contents

Writing Python using IDLE or the Python Shell is great for simple things, but those tools quickly turn larger programming projects into frustrating pits of despair. Using an IDE, or even just a good dedicated code editor, makes coding fun—but which one is best for you?

Fear not, Gentle Reader! We are here to help explain and demystify the myriad of choices available to you. We can't pick what works best for you and your process, but we can explain the pros and cons of each and help you make an informed decision.

To make things easier, we'll break our list into two broad categories of tools: the ones built exclusively for Python development and the ones built for general development that you can use for Python. We'll call out some Whys and Why Nots for each. Lastly, none of these options are mutually exclusive, so you can try them out on your own with very little penalty.

But first…

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.

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.

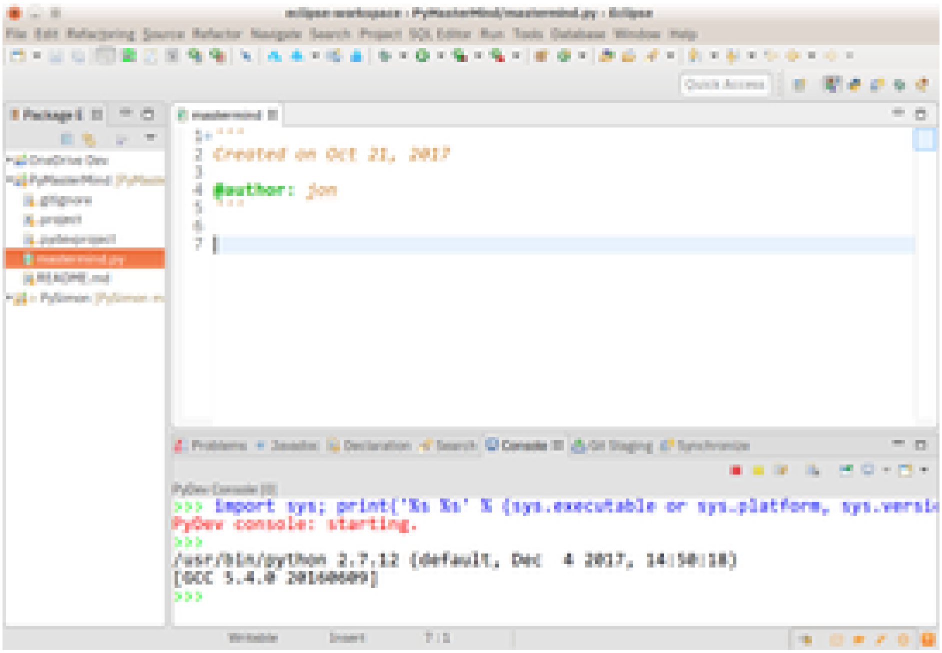
General Editors and IDEs with Python Support

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.



```
'''
Created on Oct 21, 2017

@author: jon
'''

```

```
>>> import sys; print('%s %s' % (sys.executable or sys.platform, sys.versi
PyDev console: starting.
>>>
/usr/bin/python 2.7.12 (default, Dec  4 2017, 14:50:18)
[GCC 5.4.0 20160609]
>>>
```

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

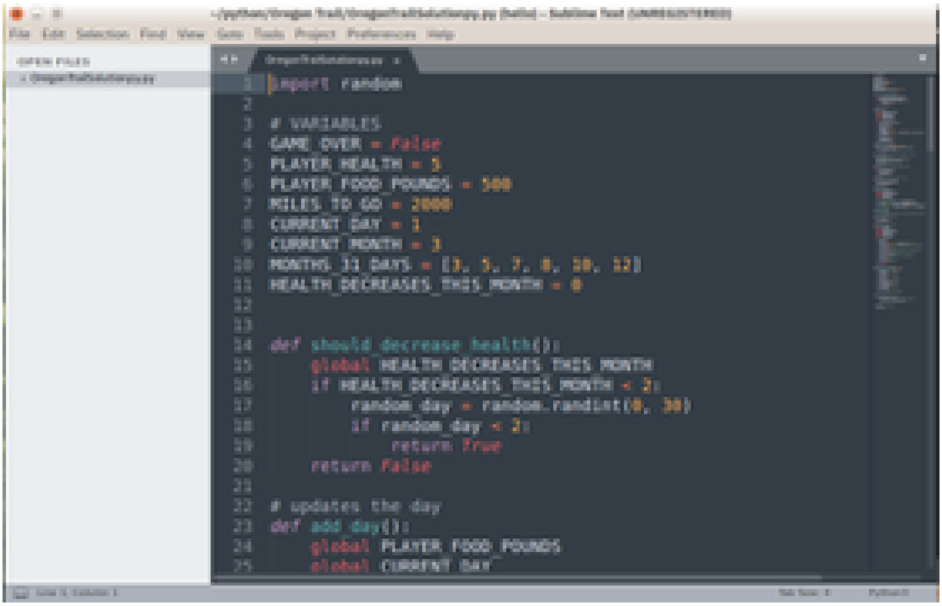
Cons: If you’re just starting out with Python, or with software development in general, Eclipse can be a lot to handle. Remember when I said IDEs are larger and require more knowledge to use properly? Eclipse is all that and a bag of (micro)chips.

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.



```python
import random

# VARIABLES
GAME_OVER = False
PLAYER_HEALTH = 5
PLAYER_FOOD_POUNDS = 500
MILES_TO_GO = 2000
CURRENT_DAY = 1
CURRENT_MONTH = 3
MONTHS_31_DAYS = [1, 5, 7, 8, 10, 12]
HEALTH_DECREASES_THIS_MONTH = 0


def should_decrease_health():
    global HEALTH_DECREASES_THIS_MONTH
    if HEALTH_DECREASES_THIS_MONTH < 2:
        random_day = random.randint(0, 30)
        if random_day < 2:
            return True
    return False

# updates the day
def add_day():
    global PLAYER_FOOD_POUNDS
    global CURRENT_DAY
```

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

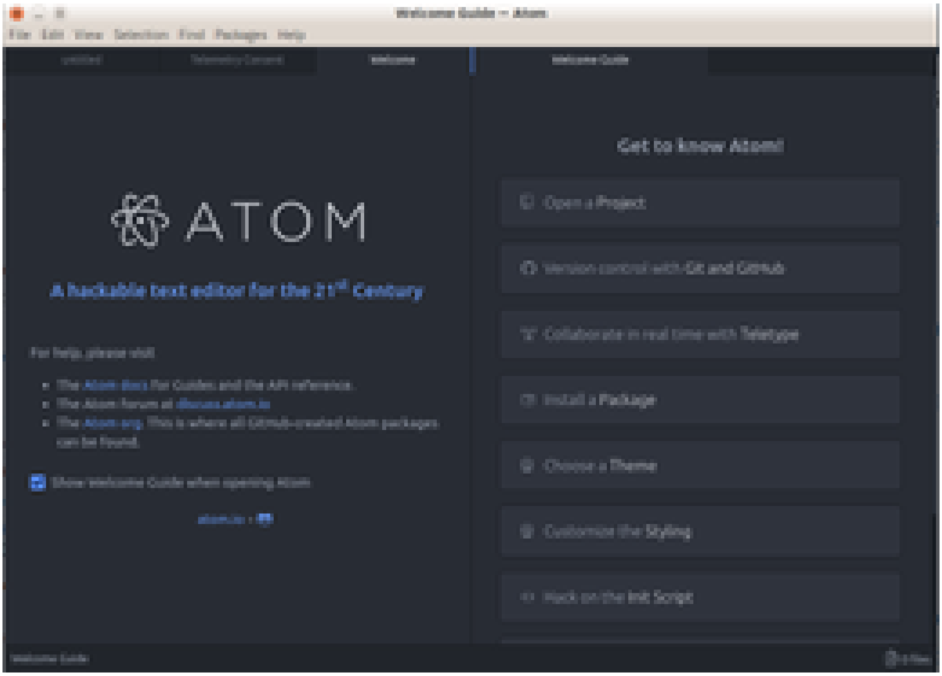
Cons: Sublime Text is not free, although you can use the evaluation version for an indefinite period of time. Installing extensions can be tricky, and there’s no direct support for executing or debugging code from within the editor.

To make the most of your Sublime Text setup, read our [Python + Sublime Text setup guide]() and consider our [in-depth video course]() that shows you how to craft an effective Python development setup with Sublime Text 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 be installed when Atom is running.



For help, please visit:

- The Atom docs for Guides and the API reference.
- The Atom forum at discuss.atom.io
- The Atom org. This is where all GitHub-created Atom packages can be found.

[✓] Show Welcome Guide when opening Atom

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

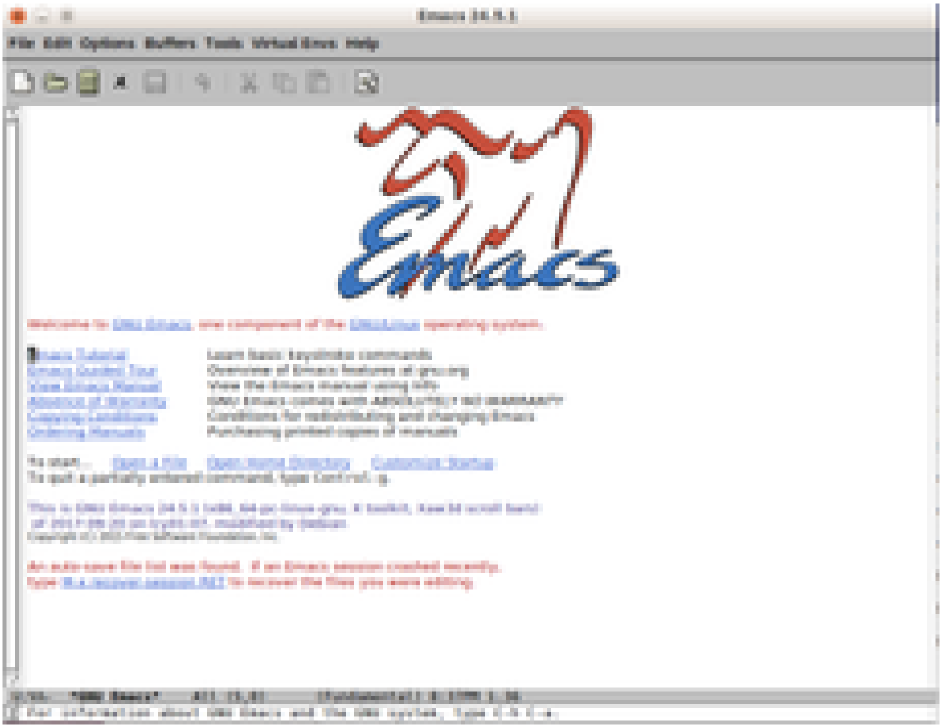
Cons: Build and debugging support aren’t built-in but are community provided add-ons. Because Atom is built on Electron, it’s always running in a JavaScript process and not as a native application.

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.



Welcome to GNU Emacs, one component of the GNU/Linux operating system.

Emacs Tutorial   Learn basic keystroke commands
Emacs Guided Tour   Overview of Emacs features at gnu.org
View Emacs Manual   View the Emacs manual using Info
Absence of Warranty   GNU Emacs comes with ABSOLUTELY NO WARRANTY
Copying Conditions   Conditions for redistributing and changing Emacs
Ordering Manuals   Purchasing printed copies of manuals

To start...   Open a File   Open Home Directory   Customize Startup
To quit a partially entered command, type Control-g.

This is GNU Emacs 24.5.1 [illegible]
[illegible]
Copyright (C) 2015 Free Software Foundation, Inc.

An auto-save file list was found. If an Emacs session crashed recently,
type M-x recover-session RET to recover the files you were editing.

[illegible]
For information about GNU Emacs and the GNU system, type C-h C-a.

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…

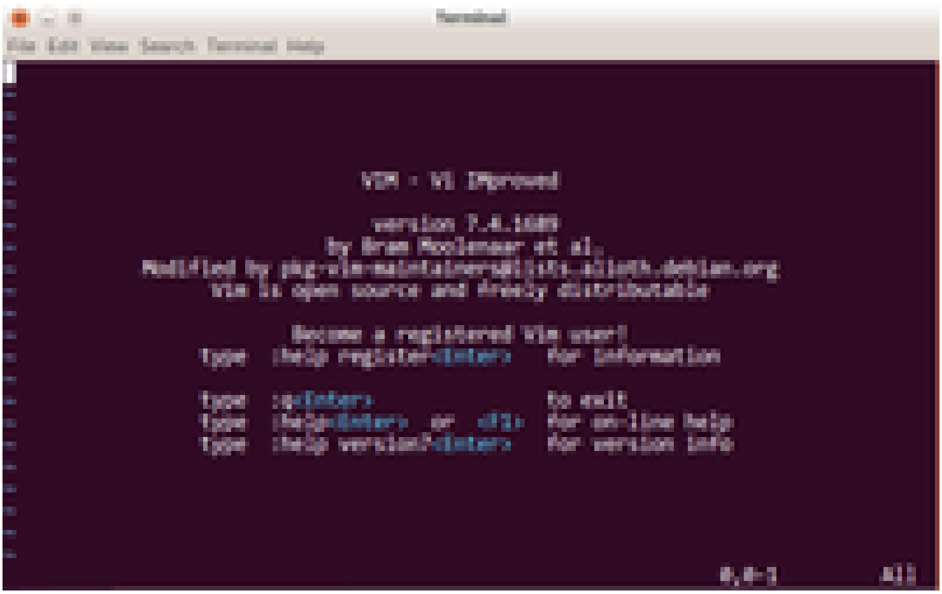
Be sure to consult our Python + Emacs setup guide to make the most of this setup.

Vi / Vim

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

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.



```
                        VIM - Vi IMproved

                        version 7.4.1689
                   by Bram Moolenaar et al.
      Modified by pkg-vim-maintainers@lists.alioth.debian.org
           Vim is open source and freely distributable

                  Become a registered Vim user!
         type  :help register<Enter>   for information

         type  :q<Enter>               to exit
         type  :help<Enter>  or  <F1>  for on-line help
         type  :help version7<Enter>   for version info

                                                 0,0-1         All
```

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.

If you're going with this combination, check out our Python + VIM setup guide with tips and plugin recommendations.

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.

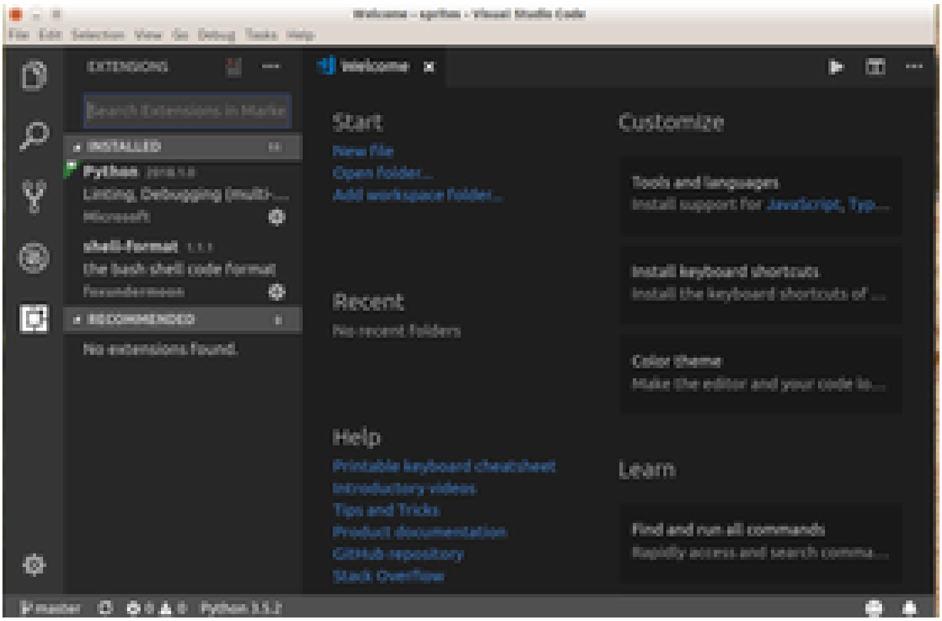
Cons: Visual Studio is a big download for just Python. Plus, if you're on Linux, you're out of luck: there's no Visual Studio install for that platform.

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.


Welcome - sprlbm - Visual Studio Code
File Edit Selection View Go Debug Tasks Help
EXTENSIONS
Search Extensions in Marke
INSTALLED
Python 2018.1.0
Linting, Debugging (multi-...
Microsoft
shell-format 1.1.1
the bash shell code format
foxundermoon
RECOMMENDED
No extensions found.
Welcome
Start
New file
Open folder...
Add workspace folder...
Recent
No recent folders
Help
Printable keyboard cheatsheet
Introductory videos
Tips and Tricks
Product documentation
GitHub repository
Stack Overflow
Customize
Tools and languages
Install support for JavaScript, Typ...
Install keyboard shortcuts
Install the keyboard shortcuts of ...
Color theme
Make the editor and your code lo...
Learn
Find and run all commands
Rapidly access and search comma...
master
Python 3.5.2

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.

Be sure to consult our tutorial on using Visual Studio Code for Python development, as well as the companion video course, to make the most of this setup.

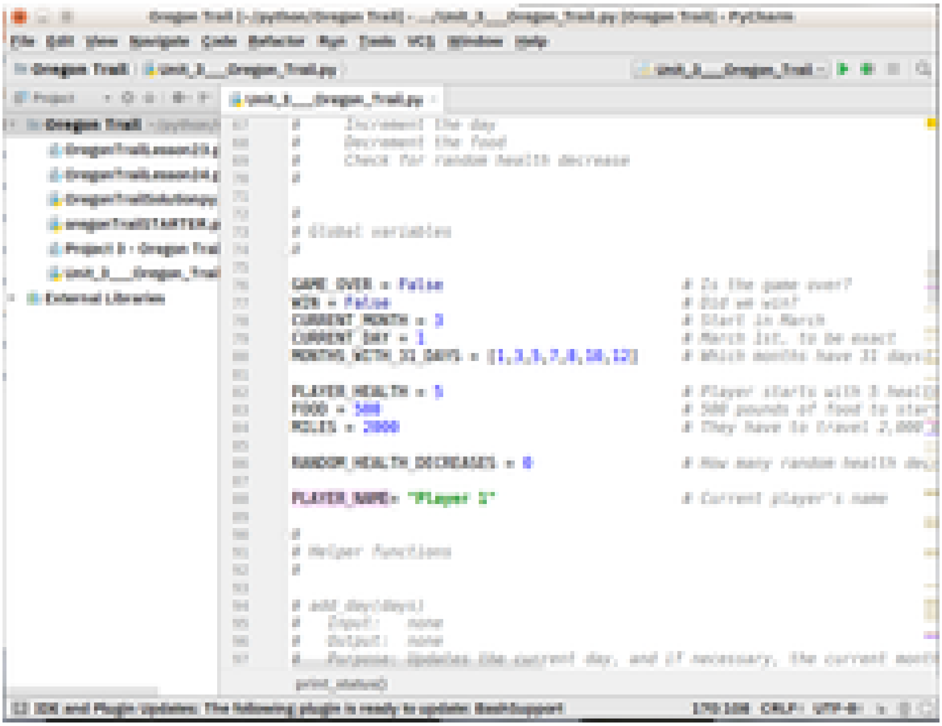
Python-Specific Editors and IDEs

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.

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.



```
#       Increment the day
#       Decrement the food
#       Check for random health decrease
#


#
# Global variables
#

GAME_OVER = False                                # Is the game over?
WIN = False                                      # Did we win?
CURRENT_MONTH = 3                                # Start in March
CURRENT_DAY = 1                                  # March 1st, to be exact
MONTHS_WITH_31_DAYS = [1,3,5,7,8,10,12]          # Which months have 31 days

PLAYER_HEALTH = 5                                # Player starts with 5 heal
FOOD = 500                                       # 500 pounds of food to star
MILES = 2000                                     # They have to travel 2,000

RANDOM_HEALTH_DECREASES = 0                      # How many random health de

PLAYER_NAME= "Player 1"                          # Current player's name


#
# Helper functions
#


# add_day(days)
#    Input:   none
#    Output:  none
#    Purpose: Updates the current day, and if necessary, the current mont
```

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.

Spyder

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

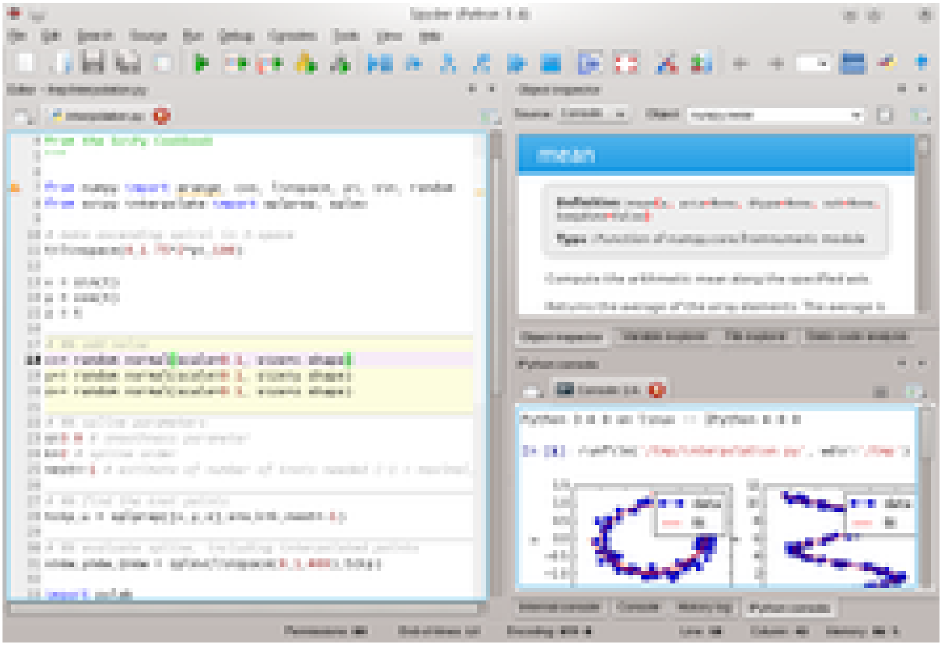
Spyder is an open-source Python IDE that’s optimized for data science workflows. Spyder comes included with the Anaconda package manager distribution, so depending on your setup you may already have it installed on your machine.

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.

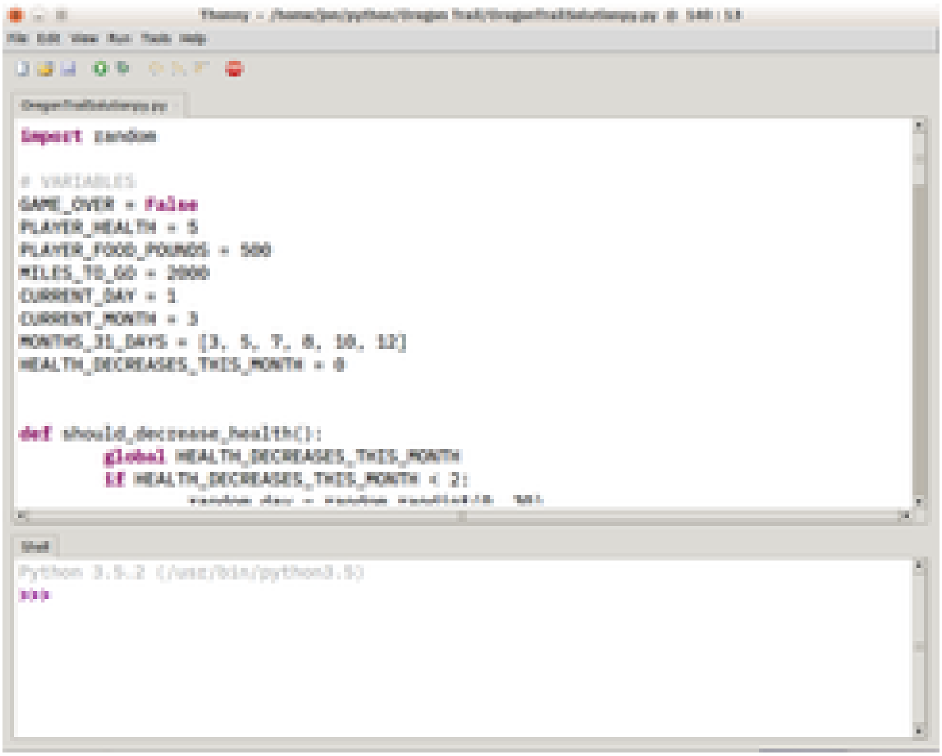
Cons: More experienced Python developers might find Spyder too basic to work with on a daily basis and instead opt for a more complete IDE or customized editor solution.

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.



```
import random

# VARIABLES
GAME_OVER = False
PLAYER_HEALTH = 5
PLAYER_FOOD_POUNDS = 500
MILES_TO_GO = 2000
CURRENT_DAY = 1
CURRENT_MONTH = 3
MONTHS_31_DAYS = [3, 5, 7, 8, 10, 12]
HEALTH_DECREASES_THIS_MONTH = 0


def should_decrease_health():
        global HEALTH_DECREASES_THIS_MONTH
        if HEALTH_DECREASES_THIS_MONTH < 2:
                random_day = random.randint(0, 30)
```

```
Python 3.5.2 (/usr/bin/python3.5)
>>>
```

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.

If you’re interested in using Thonny as your Python editor, be sure to read our dedicated article on Thonny which goes into more depth and shows you additional features.