- **Keyword**: best Python Debugging Tools
- **Length**: 1700 - 2200
- **Slug**: best-python-debuggin-tools
- **Target URL**: https://www.bairesdev.com/blog/best-python-debugging-tools/
- **Meta Description**: Discover the top debugging tools for Python development, including PyCharm, PDB, and IPython, which help you quickly identify and fix errors in your code
- **Keyphrase Group:** best python debugging tools, python debugger, python debugging tool, python IDE debugging, online python debugging
- **Article Type: templated: Listicle**

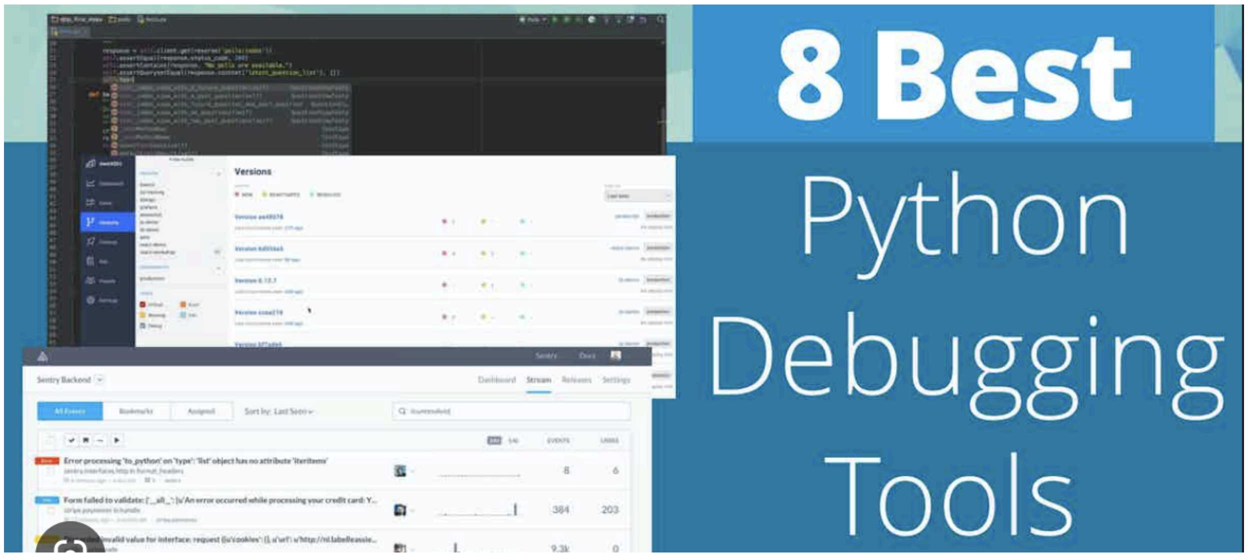
Suggested Pages/Posts For Cross-Linking

Please note, these are suggested pages / posts for cross-linking which are highly relevant. Not all pages need to be linked, but the writer should bear these page topics in mind when writing and add a link if the topic is touched upon.

- https://www.bairesdev.com/blog/python-and-big-data-trend/ | python for big data, python and big data,
- https://www.bairesdev.com/blog/python-for-web-development/ | python for web development, python and web development, *variations* **[DONE]**
- https://www.bairesdev.com/blog/top-object-oriented-programming-languages/ | best oop languages, best object oriented programming language **[DONE]**

Best Python Debugging Tools in 2023

Learn about the top Python debugging tools and their key features to choose the right tool for you.



[Image Source - **Please Replace**]

Debugging is an essential part of the software development process and can be time-consuming and challenging. Fortunately, there are **Python debugging services** and several tools available for Python developers to help them find and fix errors more efficiently. In this article, we will cover the **top Python debugging tools**, each with its own unique features and benefits, making it easier for developers to find and fix bugs in their code.

What Is Python Debugging?

Python was released in 1991 by Guido van Rossum and has become one of the best object-oriented programming languages. Over the years, as the use of Python grew, so did the need for Python debugging tools and techniques.

Python debugging refers to identifying and fixing errors in Python code. This can include finding syntax, semantic, and runtime errors that prevent the code from functioning. Various techniques and tools are available for debugging Python code.

The typical process involved in debugging Python code includes the following steps:

1. **Identify the error:** This can be done by reading error messages, reviewing log files, or testing the code and observing any unexpected behavior.
2. **Reproduce the error:** To debug the code, you need to be able to reproduce the error consistently.

3. **Isolate the source of the error:** Once you have reproduced the error, you need to isolate the source of the problem by using techniques such as adding print statements, using a debugger, or examining stack traces.
4. **Form a hypothesis:** From the information gathered, form a hypothesis about what might be causing the error.
5. **Test the hypothesis:** Use your hypothesis to guide your debugging process by making changes to the code and observing the results.
6. **Repeat the process:** If the hypothesis is incorrect, repeat the process from Step 3 until the error is fixed.
7. **Verify the fix:** Once you have fixed the error, verify that the code is working correctly by repeating the steps that initially caused the error.
8. **Document the fix:** Document the error, the steps you took to fix it, and any relevant information that will help you or others avoid similar issues in the future.

With the growing popularity of Python and its increasing use in various industries, there has been a continuous effort to improve and enhance the debugging tools and techniques available to Python developers.

Today, a rich ecosystem of debugging tools and techniques is available to help Python developers, but why are debuggers important? Let’s find out.

Why Is a Python Debugger Important?

A Python debugger is important because it helps developers find and fix errors in their code more efficiently. Without a debugger, finding and fixing bugs can be time-consuming and challenging, especially as the code grows in size and complexity.

A debugger allows developers to pause the execution of their code and inspect variables, expressions, and the program’s state at any given point. This makes it easier to isolate the source of an error and understand how the code is behaving.

Using a Python debugger has its share of benefits, including:

- **Time savings:** Debugging can be a time-consuming task, but with a debugger, the process is streamlined, allowing developers to find and fix errors more quickly.
- **Improved code quality:** Debugging helps ensure that the code is free of errors and running smoothly, resulting in improved code quality.
- **Better understanding of code:** By being able to pause the execution of the code and inspect variables, expressions, and the state of the program, a debugger can help developers gain a deeper understanding of how their code is working.
- **Debugging of complex errors:** Debuggers can isolate and debug complex errors that may not be immediately obvious.
- **Improved collaboration:** Debuggers can help facilitate collaboration among team members by providing a common tool and process for finding and fixing errors.

- **Improved efficiency:** Debuggers often provide features such as breakpoints, step-by-step execution, and the ability to examine the call stack, making the debugging process more efficient and intuitive.

A Python debugging tool is essential for any Python developer looking to improve the quality and efficiency of their code and in the code refactoring process. But before choosing a debugger, there are things to consider when choosing the best tool to debug your code. Let's take a look at a few of those considerations.

Things To Consider When Choosing the Best Tool for Debugging Python Code

Choosing the right tool for debugging code is an important decision for any Python developer, as it can greatly impact the efficiency and effectiveness of the debugging process. There are a few factors to consider when evaluating which debugger is best for your needs, including:

- **Ease of use:** Look for a debugger that is easy to use. A good debugger should provide an intuitive and streamlined experience for finding and fixing errors in code.
- **Integration with development environment:** Consider a debugger that integrates well with your preferred development environment and provides a seamless experience.
- **Debugging capabilities:** Look for a debugger that provides the necessary debugging capabilities for your needs, such as breakpoints, step-by-step execution, the ability to inspect variables and expressions, and more.
- **Debugging performance:** Consider a debugger that performs well and does not slow down the debugging process.
- **Support and community:** Consider a debugger with a strong community of users and developers and robust support and documentation.
- **Cost:** Consider the cost of the debugger, especially if you are on a budget or working on an open-source project.
- **Compatibility:** Consider a debugger compatible with the version of Python you are using along with your operating and hardware system.

Different types of customers, industries, and companies will have additional considerations to evaluate when choosing the best Python debugger tool. Here are a few examples:

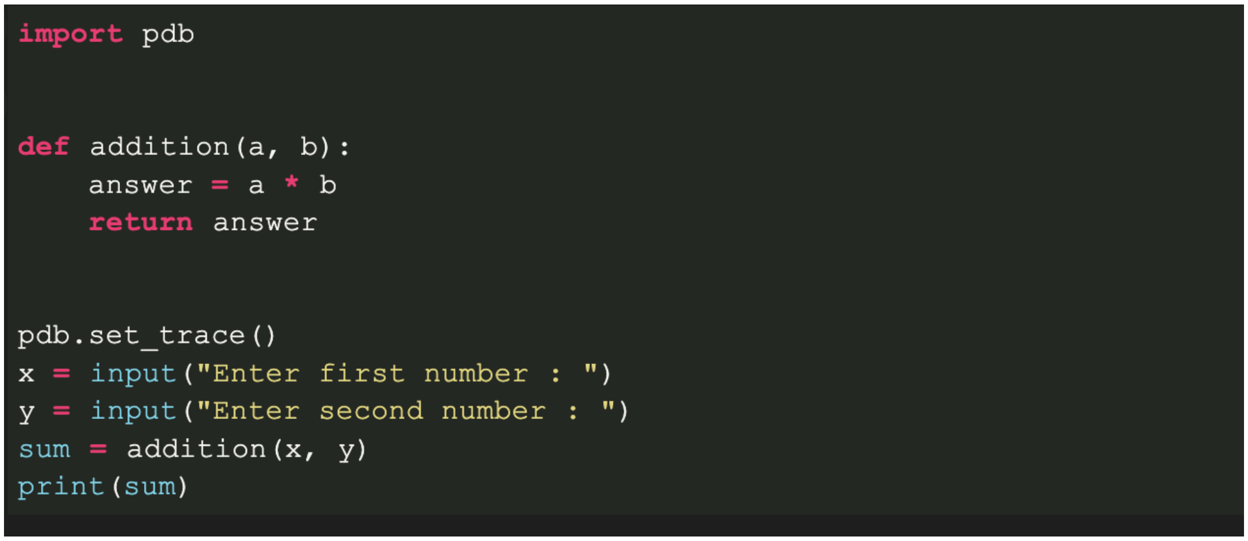
- **Enterprise customers:** Security and stability are often top priorities for enterprise customers, so they may look for a debugger with a proven track record that is widely used in their industry. Additionally, they may require a debugger that integrates well with their existing tools and systems and provides robust support and maintenance.
- **Small businesses:** Small businesses may have limited budgets, so cost may be a significant consideration when choosing a debugger. They may also prioritize ease of use and the ability to quickly find and fix errors, as they may have limited resources for debugging.

- **Open-source projects:** For open-source projects, compatibility with multiple platforms may be necessary. They may also prioritize a debugger with a large and active community and strong support and documentation.
- **Educational institutions:** A debugger that is easy to use and helps students learn and understand code is essential. They may also prioritize a debugger that has good documentation and a supportive community to help students quickly find answers to their questions.
- **Game developers:** Game developers may prioritize a debugger that provides detailed information about performance, memory usage, and other game-specific metrics. They may also need a debugger that integrates well with the game engine they are using.

These are just a few examples of the criteria different types of customers, industries, and companies may consider when evaluating the best Python debugger tool. Ultimately, the choice will depend on each individual's or organization's specific needs and requirements. Now that you know what to consider when choosing a debugging tool, let's dive into the top debugging tools for Python.

The 8 Top Debugging Tools for Python

#1 PDB



```python
import pdb


def addition(a, b):
    answer = a * b
    return answer


pdb.set_trace()
x = input("Enter first number : ")
y = input("Enter second number : ")
sum = addition(x, y)
print(sum)
```

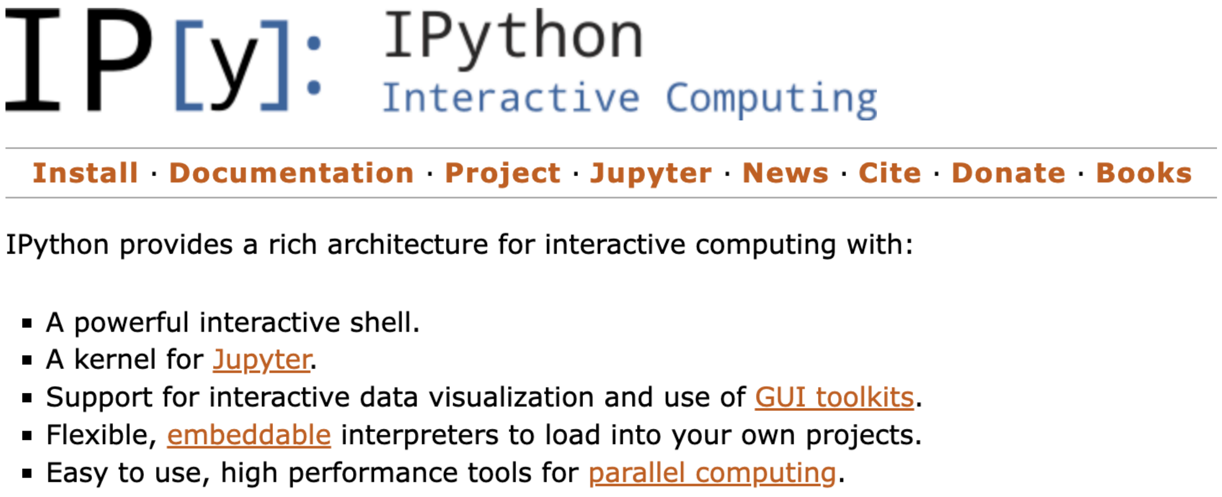
The Python debugger (PDB) is a built-in debugger for Python that provides a simple and straightforward way to debug Python code and is included in the standard library. It provides a simple command-line interface for debugging, allowing you to step through your code, set breakpoints, and inspect variables. This tool is suitable for simple debugging tasks and best suited for developers comfortable with command-line tools.

Key Debugging Features

- Step-by-step debugging

- Breakpoints
- Interactive debugging
- Built into Python

#2 IPython

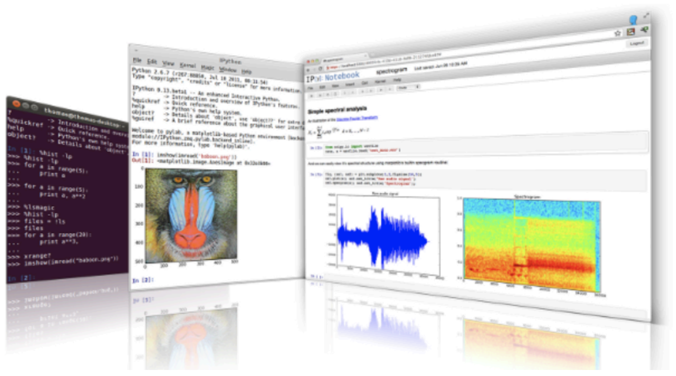


IP[y]: IPython Interactive Computing

Install · Documentation · Project · Jupyter · News · Cite · Donate · Books

IPython provides a rich architecture for interactive computing with:

- A powerful interactive shell.
- A kernel for Jupyter.
- Support for interactive data visualization and use of GUI toolkits.
- Flexible, embeddable interpreters to load into your own projects.
- Easy to use, high performance tools for parallel computing.



IPython is an interactive shell for Python that provides advanced debugging capabilities, including step-by-step debugging, interactive debugging, and post-mortem debugging. IPython also allows you to access shell commands from within the debugger, making it a versatile tool for a wide range of debugging needs.

Key Debugging Features

- Advanced debugging capabilities
- Step-by-step debugging
- Interactive debugging
- Post-mortem debugging
- Enhanced command-line interface

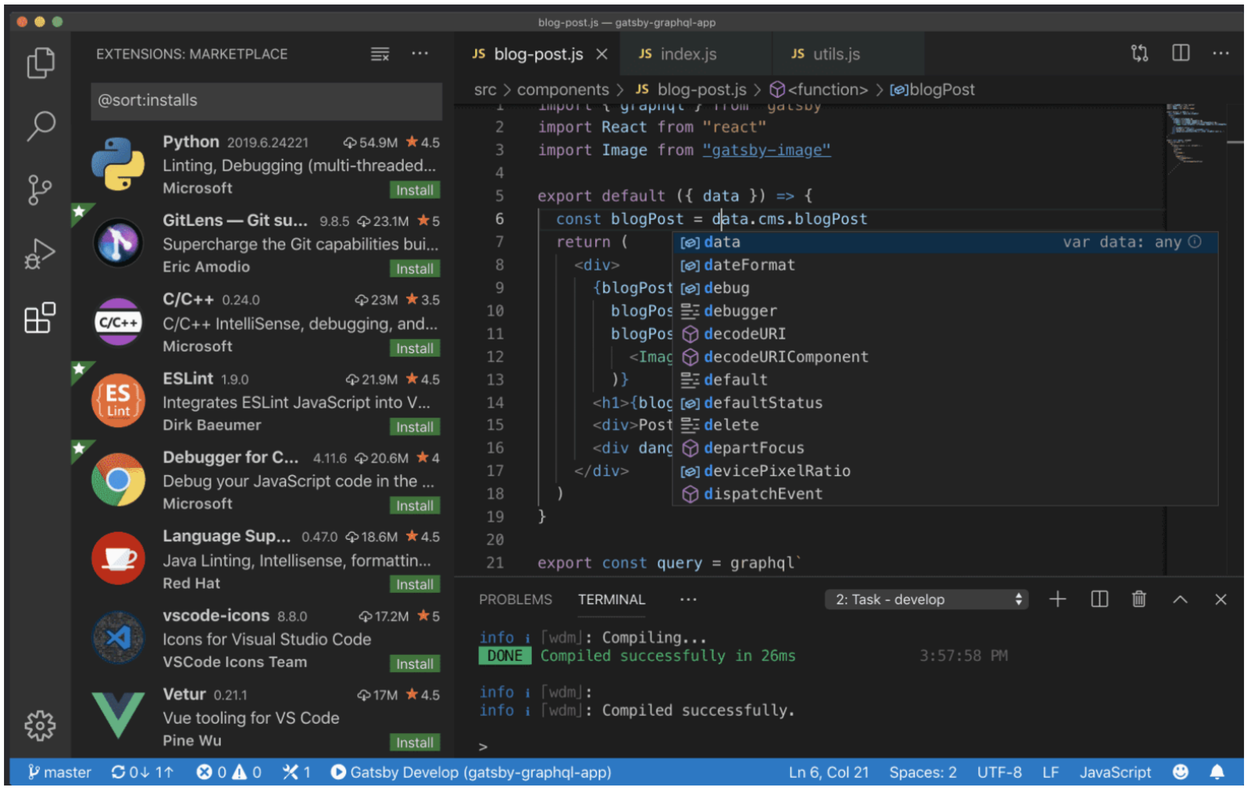
#3 PyCharm

PyCharm is a popular integrated development environment (IDE) for Python that includes a powerful debugger that provides advanced features such as multi-threaded debugging, remote debugging, and debugging of Python web applications. The intelligent help capabilities also provide intelligent auto-completion and error checking, which reduces problems significantly. Overall, PyCharm includes many other useful tools besides debugging tools, making it a valuable addition to a developer's toolset.

Key Debugging Features

- Integrated debugger
- Multi-threaded debugging
- Remote debugging
- Debugging of Python web applications
- Advanced code editor

#4 Visual Studio Code Python Extension


blog-post.js — gatsby-graphql-app
EXTENSIONS: MARKETPLACE
@sort:installs
Python 2019.6.24221 54.9M ★4.5
Linting, Debugging (multi-threaded...
Microsoft Install
GitLens — Git su... 9.8.5 23.1M ★5
Supercharge the Git capabilities bui...
Eric Amodio Install
C/C++ 0.24.0 23M ★3.5
C/C++ IntelliSense, debugging, and...
Microsoft Install
ESLint 1.9.0 21.9M ★4.5
Integrates ESLint JavaScript into V...
Dirk Baeumer Install
Debugger for C... 4.11.6 20.6M ★4
Debug your JavaScript code in the ...
Microsoft Install
Language Sup... 0.47.0 18.6M ★4.5
Java Linting, Intellisense, formattin...
Red Hat Install
vscode-icons 8.8.0 17.2M ★5
Icons for Visual Studio Code
VSCode Icons Team Install
Vetur 0.21.1 17M ★4.5
Vue tooling for VS Code
Pine Wu Install
blog-post.js
index.js
utils.js
src > components > blog-post.js > <function> > blogPost
PROBLEMS
TERMINAL
2: Task - develop
master
0↓ 1↑
0 0
1
Gatsby Develop (gatsby-graphql-app)
Ln 6, Col 21
Spaces: 2
UTF-8
LF
JavaScript

```
import { graphql } from "gatsby"
import React from "react"
import Image from "gatsby-image"

export default ({ data }) => {
  const blogPost = data.cms.blogPost
  return (
    <div>
      {blogPost
        blogPos
        blogPos
          <Imag
        )}
      <h1>{blog
      <div>Post
      <div dang
    </div>
  )
}

export const query = graphql`
```

```
data                var data: any
dateFormat
debug
debugger
decodeURI
decodeURIComponent
default
defaultStatus
delete
departFocus
devicePixelRatio
dispatchEvent
```

```
info ｉ ｢wdm｣: Compiling...
 DONE  Compiled successfully in 26ms              3:57:58 PM

info ｉ ｢wdm｣:
info ｉ ｢wdm｣: Compiled successfully.

>
```

Visual Studio Code is a popular code editor that provides a comprehensive set of debugging features for Python. This tool includes step-by-step debugging, breakpoints, variable inspection, and additional tools for code highlighting, linting, and auto-completion. Visual Studio Code is a good option for developers looking for an intelligent code editor with built-in debugging capabilities.

Key Debugging Features

- Comprehensive set of debugging features for Python
- Integrated with the code editor
- Supports step-by-step debugging
- Breakpoints
- Variable inspection

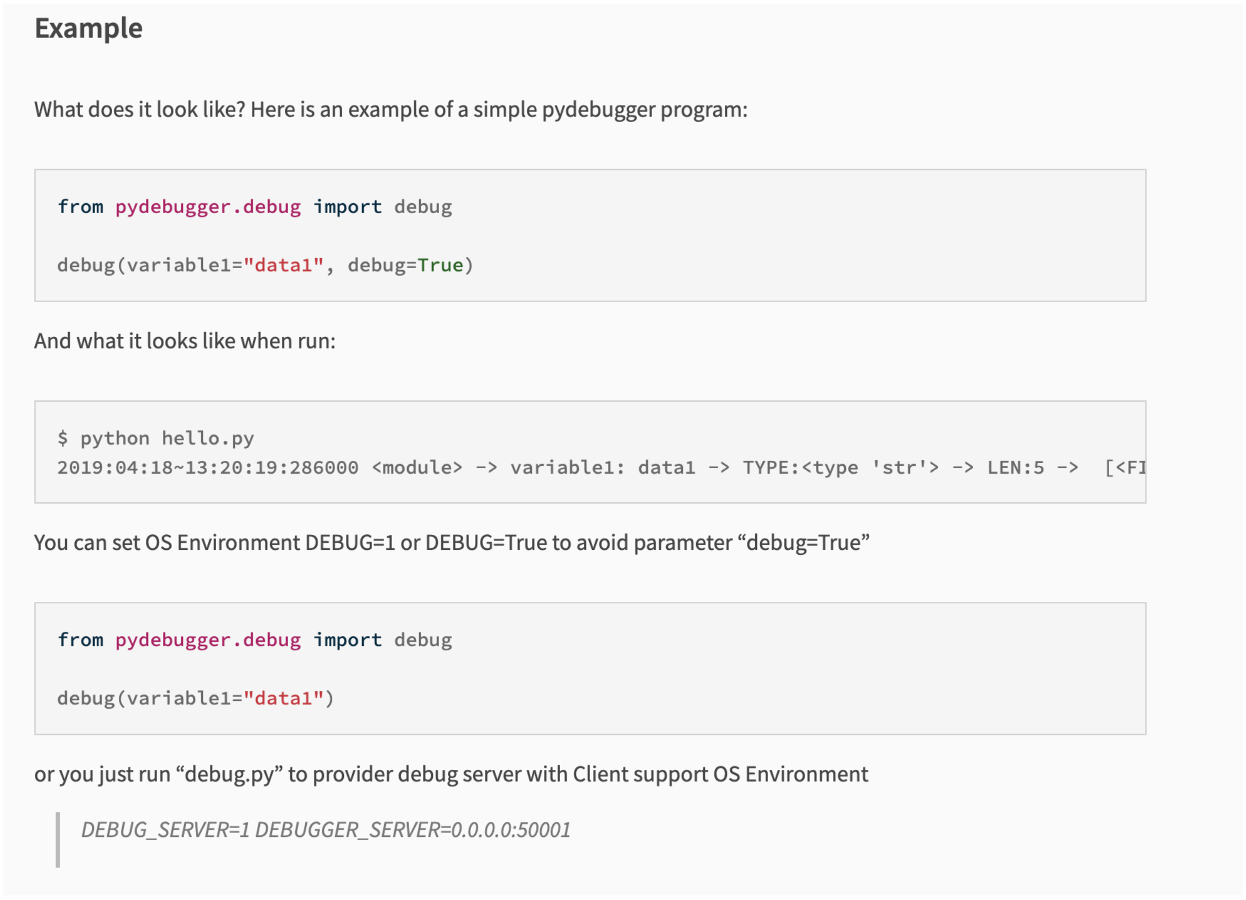
#5 PyDev

PyDev is a Python IDE for Eclipse that supports Django applications and includes code analysis, testing, and refactoring tools, and provides an integrated debugger, including multi-threaded debugging, remote debugging, and debugging of Django applications. This debugger is recommended for users who need a debugger when using Python for web development. It is a good choice for developers who prefer a full-featured IDE with a strong emphasis on debugging capabilities.

Key Debugging Features

- Integrated debugger
- Multi-threaded debugging
- Remote debugging
- Debugging of Django applications
- Eclipse-based IDE

#6 PyDebugger



Example

What does it look like? Here is an example of a simple pydebugger program:

```
from pydebugger.debug import debug

debug(variable1="data1", debug=True)
```

And what it looks like when run:

```
$ python hello.py
2019:04:18~13:20:19:286000 <module> -> variable1: data1 -> TYPE:<type 'str'> -> LEN:5 ->  [<FI
```

You can set OS Environment DEBUG=1 or DEBUG=True to avoid parameter “debug=True”

```
from pydebugger.debug import debug

debug(variable1="data1")
```

or you just run “debug.py” to provider debug server with Client support OS Environment

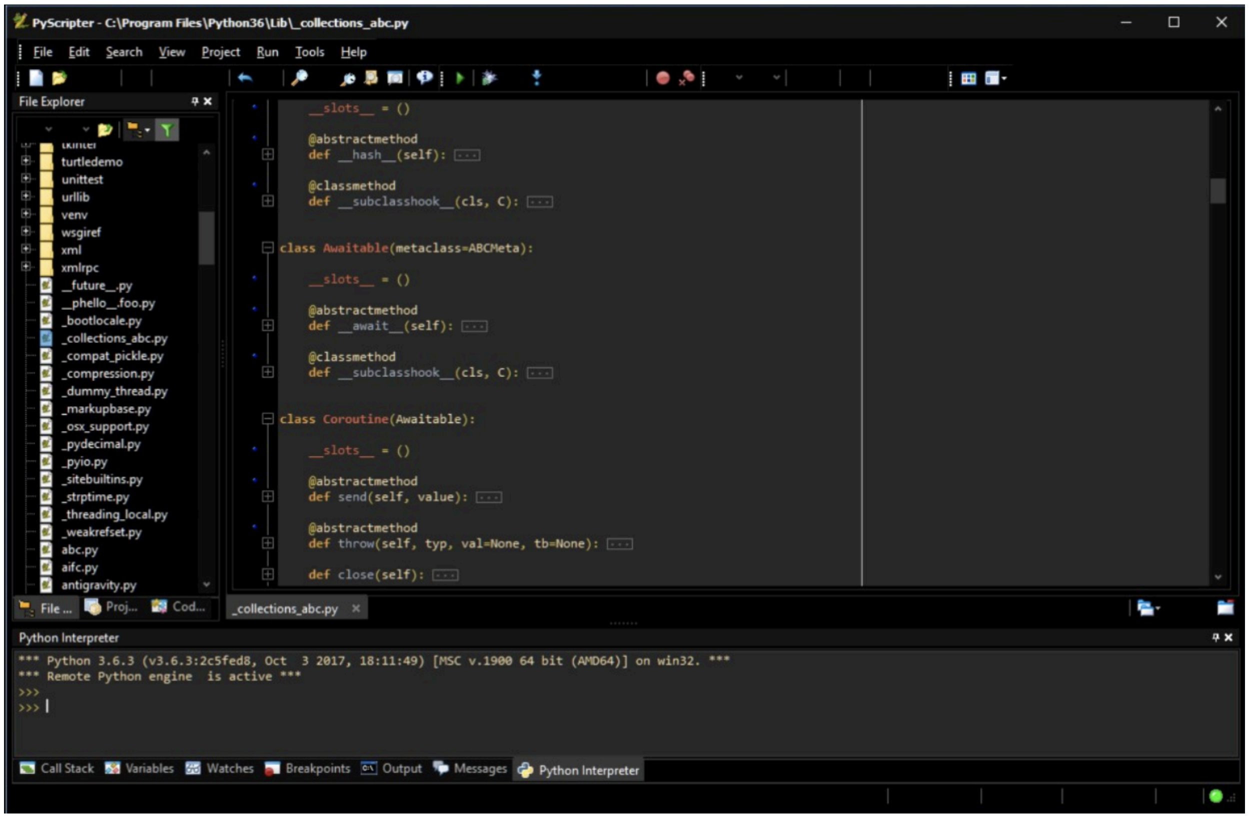
> *DEBUG_SERVER=1 DEBUGGER_SERVER=0.0.0.0:50001*

PyDebugger is a standalone debugger for Python that provides a simple and intuitive interface for step-by-step debugging, breakpoints, and variable inspection. This tool features a minimalistic design, making it easy to use and understand. PyDebugger is a good choice for developers who need a lightweight and straightforward tool for debugging Python code.

Key Debugging Features

- Standalone debugger for Python
- Simple and intuitive interface
- Step-by-step debugging
- Breakpoints
- Variable inspection

#7 PyScripter



```
    __slots__ = ()

    @abstractmethod
    def __hash__(self): ...

    @classmethod
    def __subclasshook__(cls, C): ...


class Awaitable(metaclass=ABCMeta):

    __slots__ = ()

    @abstractmethod
    def __await__(self): ...

    @classmethod
    def __subclasshook__(cls, C): ...


class Coroutine(Awaitable):

    __slots__ = ()

    @abstractmethod
    def send(self, value): ...

    @abstractmethod
    def throw(self, typ, val=None, tb=None): ...

    def close(self): ...
```

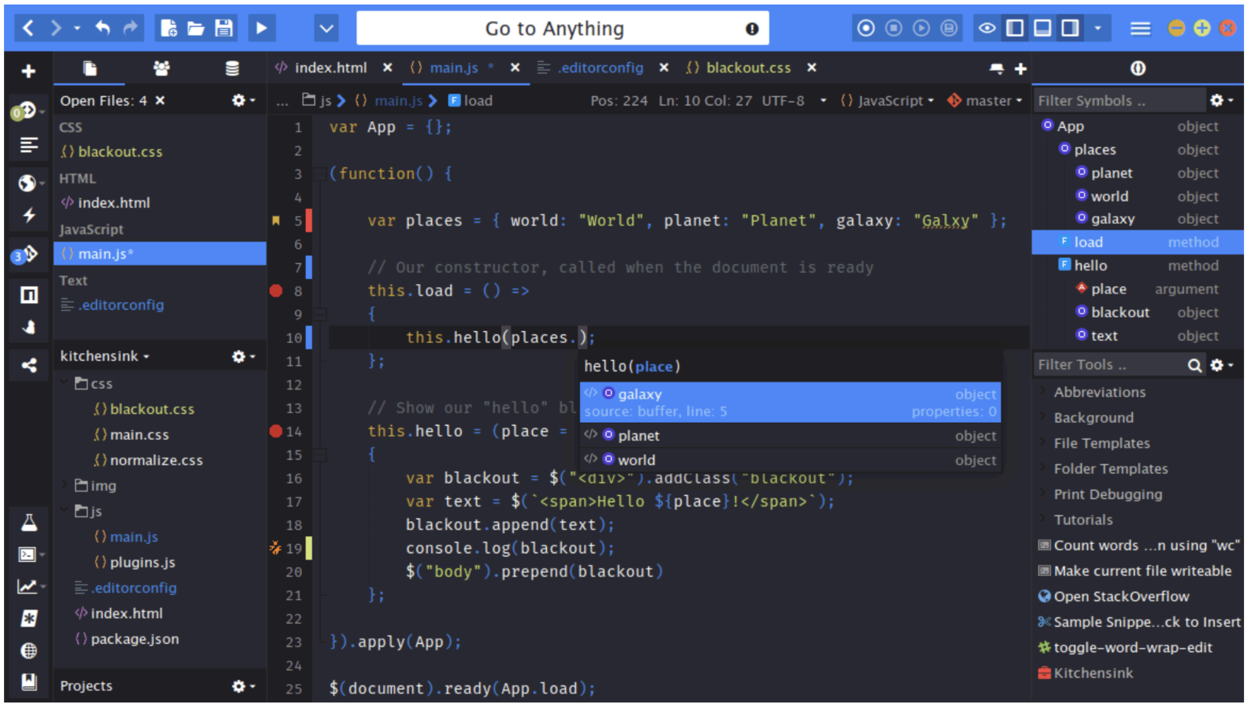
```
*** Python 3.6.3 (v3.6.3:2c5fed8, Oct  3 2017, 18:11:49) [MSC v.1900 64 bit (AMD64)] on win32. ***
*** Remote Python engine  is active ***
>>>
>>> |
```

PyScripter is a Python IDE with an advanced debugger. This tool supports multi-threaded and remote debugging, as well as embedded and threaded code. It features an intuitive interface and includes a code editor, integrated debugging, and project management capabilities. PyScripter is a good choice for developers who need a simple yet powerful tool for debugging Python code.

Key Debugging Features

- Advanced debugger
- Multi-threaded debugging
- Remote debugging
- Debugging of threaded and embedded code
- Code editor with advanced features

#8 Komodo IDE



```
var App = {};

(function() {

    var places = { world: "World", planet: "Planet", galaxy: "Galxy" };

    // Our constructor, called when the document is ready
    this.load = () =>
    {
        this.hello(places.);
    };

    // Show our "hello" bl
    this.hello = (place =
    {
        var blackout = $("<div>").addClass("blackout");
        var text = $(`<span>Hello ${place}!</span>`);
        blackout.append(text);
        console.log(blackout);
        $("body").prepend(blackout)
    };

}).apply(App);

$(document).ready(App.load);
```

Komodo IDE is a full-featured Python IDE that includes a comprehensive debugger for multi-threaded and remote debugging. This tool supports threaded and embedded code and consists of a code editor, integrated debugging, and project management capabilities. Komodo IDE is a good option for developers who need a complete development environment for their Python projects.

Key Debugging Features

- Comprehensive debugger
- Multi-threaded debugging
- Remote debugging
- Debugging of threaded and embedded code
- Full-featured Python IDE

Conclusion

In conclusion, debugging is an essential aspect of software development, and Python developers have many powerful tools available to help them identify and fix errors in their code. The best Python debugger tool will depend on your specific needs and requirements, but with a variety of options available, you're sure to find one that can help you achieve your debugging goals. So whether you are a beginner or an experienced developer, one of the top **Python debugging tools** is sure to have the features you need to effectively debug your code.

FAQ

Who Uses a Python Debugging Tool?

People in various industries and job roles, including software engineers, data scientists, web developers, educators, and game developers, just to name a few need an efficient tool to debug Python. If you are looking to hire top Python developers, then BairesDev can help. Reach out to us with your needs and lets see how we can help.

Which Tool Is Best for Integration With Visual Studio Code?

The Python extension for Visual Studio Code is the best Python debugger tool for integration with Visual Studio Code. This extension provides a comprehensive set of features for debugging Python code, including debugging interactive windows, local and remote debugging, multi-threaded debugging, and more. It integrates seamlessly with Visual Studio Code, providing a smooth and intuitive debugging experience for Python developers using the editor.