**Python debuggers**

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:

* Identify the error: This can be done by reading error messages, reviewing log files, or testing the code and observing any unexpected behavior.
* Reproduce the error: To debug the code, you need to be able to reproduce the error consistently.
* 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.
* Form a hypothesis: From the information gathered, form a hypothesis about what might be causing the error.
* Test the hypothesis: Use your hypothesis to guide your debugging process by making changes to the code and observing the results.
* Repeat the process: If the hypothesis is incorrect, repeat the process from Step 3 until the error is fixed.
* 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.
* 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.

Examples for python debuggers :

* PDB
* IPython (ipdb)
* PyCharm
* PyDev
* PyDebugger
* PyScripter
* PuDB