Project Development phase

Debugging & Traceability

Date	06 May 2023
Team ID	NM2023TMID18418
Project Name	CancerVision: Advanced Breast Cancer Prediction with Deep Learning

Debugging :

Debugging is the process of looking for bugs and their cause in applications. a bug can be an error or just some unexpected behaviour (e.g. a user complains that he/she receives an error when he/she uses an invalid date format). typically a debugger is used that can pause the execution of an application, examine variables and manipulate them.

Traceability :

Tracing "trace is a log of events within to the program"(Whitham). those events can be ordered chronologically. thats why they often contain a timestamp. Tracing is the process of generating and collecting those events. the use case is typically flow analysis.

Trace :

[2021-06-12T11:22:09.815479Z] [INFO] [Thread-1] Request started
[2021-06-12T11:22:09.935612Z] [INFO] [Thread-1] Request finished
[2021-06-12T11:22:59.344566Z] [INFO] [Thread-1] Request started
[2021-06-12T11:22:59.425697Z] [INFO] [Thread-1] Request finished

Profile:
2 "Request finished" Events
2 "Request started" Events

Debugging applications can sometimes be an unwelcome activity. You're busy working under a time crunch and you just want it to work. However, at other times, you might be learning a new language feature or experimenting with a new approach and want to understand more deeply how something is working.

Regardless of the situation, debugging code is a necessity, so it's a good idea to be comfortable working in a debugger. In this tutorial, I'll show you the basics of using pdb, Python's interactive source code debugger.

I'll walk you through a few common uses of pdb. You may want to bookmark this tutorial for quick reference later when you might really need it. pdb, and other debuggers, are indispensable tools. When you need a debugger, there's no substitute. You really need it.

By the end of this tutorial, you'll know how to use the debugger to see the state of any variable in your application. You'll also be able to stop and resume your application's flow of execution at any moment, so you can see exactly how each line of code affects its internal state.

This is great for tracking down hard-to-find bugs and allows you to fix faulty code more quickly and reliably. Sometimes, stepping through code in pdb and seeing how values change can be a real eye-opener and lead to “aha” moments, along with the occasional “face palm”.

pdb is part of Python’s standard library, so it’s always there and available for use. This can be a life saver if you need to debug code in an environment where you don’t have access to the GUI debugger you’re familiar with.