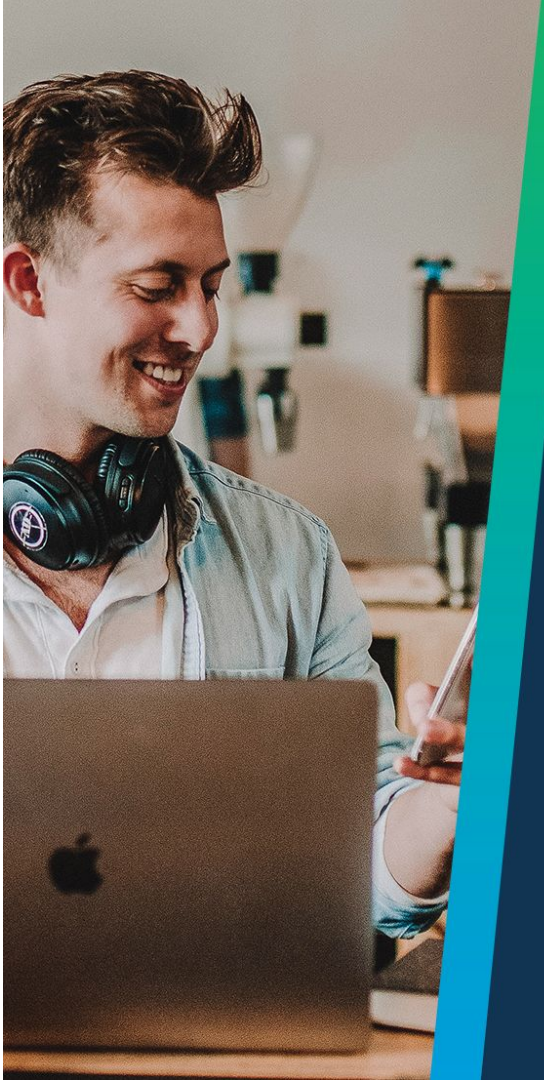


Cyber Security Bootcamp

Hyperiondev

Hypothesis-Driven Debugging using the Stack Trace

Welcome

Your Lecturer for this session

Joshua van Staden

Objectives

1. Learn about the stack trace
2. Gain skills for debugging failing code

What is the Stack Trace?

- When your code hits an error, you will see a large body of text with error messages.
- This is the stack trace (or traceback).
  - It shows each method that was called when the error occurred.
  - A “trail of breadcrumbs”.

An Example

```
def calling_function():
    another_function()

def another_function():
    last_function()

def last_function():
    return 5/0

result = calling_function()
```

```
Traceback (most recent call last):
  File "C:\Python310\debugging.py", line 10, in <module>
    result = calling_function()
  File "C:\Python310\debugging.py", line 2, in calling_function
    another_function()
  File "C:\Python310\debugging.py", line 5, in another_function
    last_function()
  File "C:\Python310\debugging.py", line 8, in last_function
    return 5/0
ZeroDivisionError: division by zero
```

Exceptions and Errors

- How to tell what caused the error?
  - It depends on what type of error was raised.
  - Multiple types of error.
- **ZeroDivisionError** - When you try to divide by zero.
- **NameError** - When you try to read the value of a variable that hasn't been created.
- Various others - can be found on official Python website.

How do we Fix these Bugs?

- Sometimes, these error messages can be vague and difficult to interpret.
- Without a good understanding of what is happening, debugging can be difficult.
- We need to follow a process.
    - Hypothesis-driven debugging.

Step 1: Make Observations

- Try to replicate the issue, and find out what causes the unexpected behaviour.
- Note the stack trace - inspect the code that is causing the error to occur.

Step 2: Question

- When did the problem start?
- Is this a recent problem?
- Could it have arisen from a recent change?
- Many other questions can be asked. This will help to narrow down the potential sources of the problem.

Step 3: Form a Hypothesis

- Based on your questioning, you can now make a guess of what is wrong.
- A hypothesis is like a guess, but something that can be tested.

Step 4: Make a Prediction

- “If I make this change, then we should see this change in the output”
- You can either guess that the change will completely fix the bug, or if it will cause some shift in the output.

Step 5: Test your Hypothesis

- Now it is time to test the hypothesis that you made.
- If your code changes as you have predicted, then your hypothesis is correct.
- If not, you will need to go back and change your hypothesis.

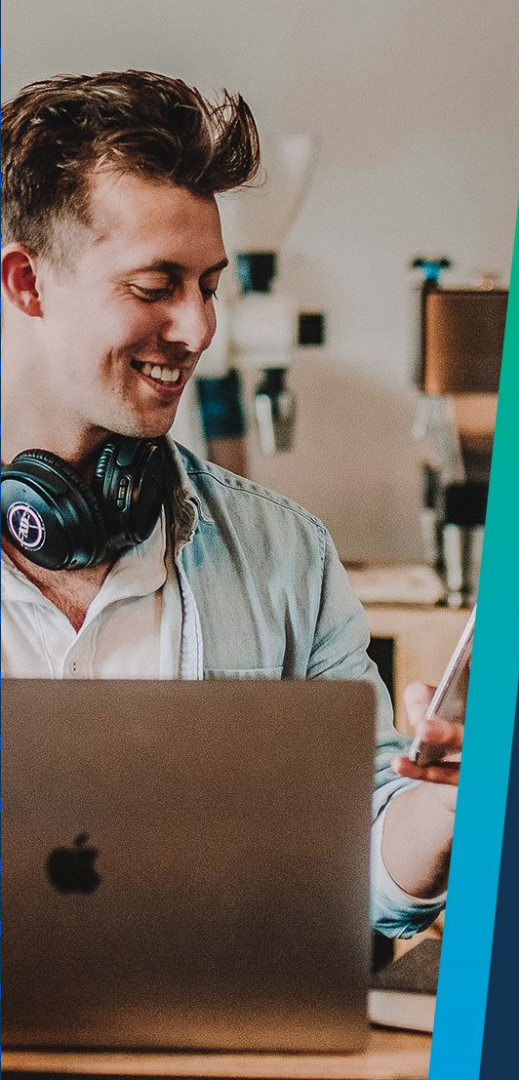
Gaining Visibility into your Code

- It is often useful to take a “look” at how your code is executing.
- Various debuggers are available for IDLE and VS Code.
- Print statements are used quite commonly.
- If you want to know the value of a variable at a certain point, just print it to console.

Hyperiondev

Q & A Section

Please use this time to ask any questions relating to the topic explained, should you have any



Hyperiondev

Thank you for joining us