PHANTASTIC BUGS

AND HOW TO FIND THEM

Magdalena Rother

lenarother

lenarother

Overview Repositories 70 Projec

Magdalena Rother

lenarother

I'm a django backend developer from Berlin.

Edit profile

42 followers · **43** following

Berlin

rother.magdalena@gmail.com

Search

Magdalena (Musielak) Rother
Software Engineer at thermondo

Software Engineer
thermondo · Full-time
Aug 2022 - Present · 2 yrs 1 mo
Berlin, Germany

Software Engineer
CompuGroup Medical SE & Co. KGaA · Part-time
Aug 2021 - Jun 2022 · 11 mos
Berlin, Germany

Senior Backend Developer
camping.info
Jan 2020 - Jun 2021 · 1 yr 6 mos

Django backend developer
Moccu - Digitalagentur
Aug 2016 - Jan 2020 · 3 yrs 6 mos
Berlin Area, Germany

Python developer
orderbird AG
Nov 2014 - Aug 2016 · 1 yr 10 mos
Berlin Area, Germany

Postdoctoral researcher
Humboldt University of Berlin
Aug 2012 - Aug 2014 · 2 yrs 1 mo

Agenda

- Hands-on debugging
- Types of bugs
- Debugging techniques

https://github.com/lenarother/debugging-workshop

Types of Bugs

- **Syntax Errors**

  When there is a bug in the Python Syntax or indentation, Python will refuse to execute any code. From the error message you know that there is a problem and roughly where it is. Most of the time, syntax issues are fairly easy to fix. Your editor should highlight them right away.

- **Runtime Exceptions**

  When Python executes a part of the code but then crashes, you have an Exception at runtime. The NameError, TypeError, ValueError and many others fall in this category. The error message will give you some hints what to look for, but the source of the error might be somewhere else. In any case you know there is a problem, and Python knows it too.

- **Semantic errors**

  If Python executes the code without error, but does not deliver the expected result, you can call this a Semantic Error. You still know that there is a problem, but Python doesn't. Semantic errors are harder to debug.

- **Complex issues**

  More complex bugs are: race conditions (timing issues with multiple threads), border cases (bugs that only occur with exotic input), and Heisenbugs (bugs that disappear when you start debugging them). These are tough.

- **Unknown Bugs**

  Finally, there might be bugs in the program that nobody knows about. This is of course bad, and we need to keep that possibility in the back of our heads.

Debugging Techniques

- read the code
- read the error message (message on bottom, line numbers, type of error on top)
- inspect variables with print(x)
- inspect the type of variables with print(type(x))
- reproduce the bug
- use minimal input data
- use minimal number of iterations
- isolate the bug by commenting parts of the program
- drop assertions in your code
- write more tests
- explain the problem to someone else
- step through the code in an interactive debugger
- clean up your code
- run a code cleanup tool (black)
- run a type checker (mypy)
- run a linter (pylint)
- take a break
- sleep over it
- ask for a code review
- write down what the problem is
- draw a formal description of your program logic (flowchart, state diagram)
- draw a formal description of your data structure (class diagram, ER-diagram)
- background reading on the library / algorithm you are implementing
- google the error message

Acknowledgements

Based on:

https://github.com/krother/advanced_python

Thank You!