Debugging without print()

• • •

An introduction to pdb

Matt Curcio & Emily Charles
March 9, 2023

Bugs or Moths?

1947 Harvard Mark II Team on a calculator program

What, when, where, why pdb?

- Python debugger (Pdb) comes 'Built-in'

- No GUI's? Use CLI

- More info. than: print(f'a = {a}')

- Step thru a program

- Check ANY Variables

- No need to remove 'print' statements later

Alternatives

- 'print' is very simple

- Use <u>Logging ala Glenn</u>

- ipdb, pudb, pdb++, PyCharm, IDE's

- Even more?

  https://wiki.python.org/moin/PythonDebuggingTools

Ready, Steady, Go!

Option 1: $ python3 -m pdb title.py   # -m(odule)
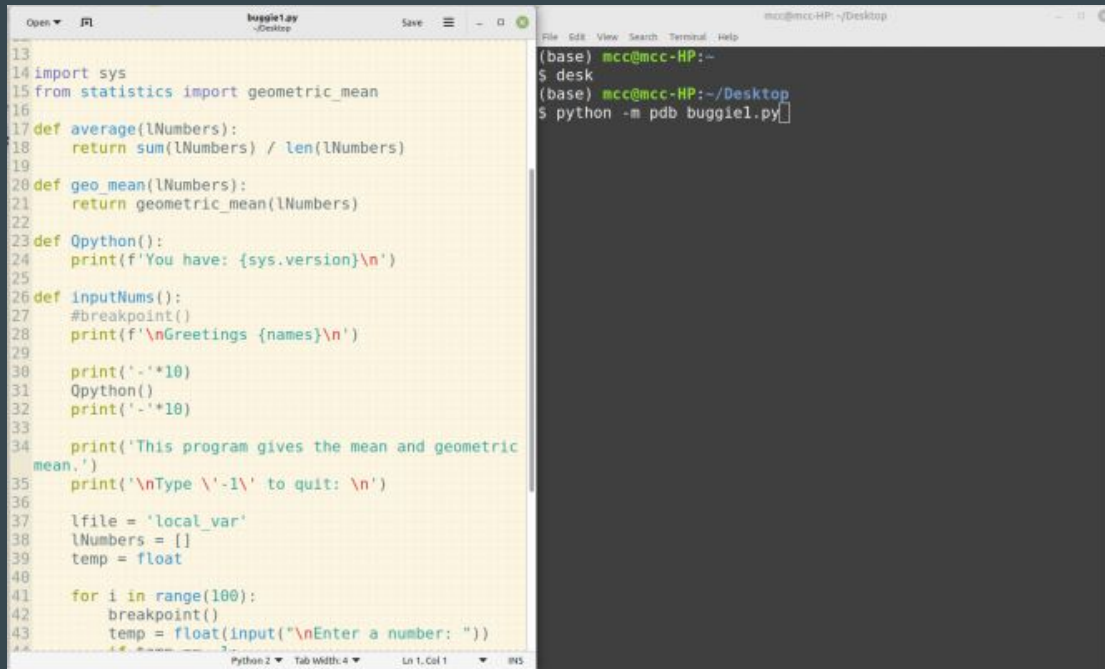
Option 2: Set breakpoints in code

>3.7 breakpoint()   <3.7 import pdb; pdb.set_trace()

Option 3: __import__('pdb').set_trace()

One Possible Workflow

<u>Step 1</u> Open terminal & your code in a text editor

Help

- @ Pdb: 'h' or 'help' <command>

- Pdb has NO man page!?!

- https://docs.python.org/3/library/pdb.html

- Matt's PDF

- Google: 'python pdb' NOT the Protein DataBase (PDB)

Pdb Interpreter Acts like Python

One Possible Workflow

Step 2 cli: python3 -m pdb title.py   # Start pdb

Step 3 Find your error(s)
          Via: Pdb: Use c*(ontinue)*, n*(ext), or* s*(tep)*

Step 4 Edit code in your text editor

Step 5 Pdb: r(estart) your code & find next error(s)

Step 6 Repeat

What You Cannot Do

- You **cannot find logical errors**

  For example:

  - Calculation errors, F=m/a, instead of F=ma
  - Deleting rows instead of columns
  - Counting from 1 instead of 0, etc.

What You Can Do

SYNTAX ERRORS

- Closing parenthesis: print("Hello World"
- Using invalid characters: a = 2 x 5, b = 10 ~ 4
- Missing Colons: if a == 10; or if a == 6
- Incorrect Indentation

What You Can Do

RUNTIME ERRORS

- ZeroDivisionError: c = a/0
- TypeError: a = '10', b = 5, c = a + b
- NameError: print(x), name 'x' is not defined
- IndexError: d = range(10), print(d[10])

Navigation within the Pdb interpreter

- l(ist)   list 11 lines surrounding
- n(ext)   execute the current line
- s(tep)   step thru function from current line
- c(ontinue)   execute until next breakpoint
- restart   restart your program from the top

Interacting with Code

- p <name>   print value of variable
- !<expr>   execute the expression
- run [args]   restart the debugger
- arguments [args]
- q(uit)   exit the debugger

  NOTE: this acts like a python interpreter

Now for a Live Demo

Don't Debug, De'moth' Python!

Other Resources

- pdb commands explained: https://www.codementor.io/@stevek/advanced-python-debugging-with-pdb-g56gvmpfa
- Ned's pdbrc: https://kylekizirian.github.io/ned-batchelders-updated-pdbrc.html