pdb Python debugger

CONCEPTS

Command names are written like s(tep): type the short part or the whole word. An empty line repeats the last command (except list, which continues).

Start breakpoint(), -m pdb
↓ stops
(Pdb) prompt frame, locals
↓ look
Inspect p, pp, w, ll
↓ move on
Resume n, s, r, c

Anything that is not a pdb command runs as Python in the current frame. Prefix ! when a name clashes with a command: !n = 1, !c.

START DEBUGGING

`breakpoint()`
Drop into pdb at this line

`python -m pdb` *`<script> <args>`*
Run a script under pdb from its first line

`python -m pdb -c continue` *`<script>`*
Run freely, stop only on a crash

`PYTHONBREAKPOINT=0`
Make every breakpoint() a no-op

AFTER A CRASH

`import pdb; pdb.pm()`
Post-mortem on the last uncaught exception

`pytest --pdb`
Open pdb on every failing test

`pytest -x --pdb`
Debug the first failure, then stop the run

STEP AND RUN

`n(ext)`
Run the line, stepping over calls

STEP AND RUN (cont.)

`s(tep)`
Run the line, stepping into calls

`r(eturn)`
Run until the current function returns

`c(ontinue)`
Run until the next breakpoint

`unt(il)`
Run to a higher line: leaves a loop

`q(uit)`
Leave pdb and abort the program

STACK

`w(here)`
Show the stack, newest frame at the bottom

`u(p) / d(own)`
Select the caller / callee frame

`a(rgs)`
Arguments of the current function

BREAKPOINTS

`b(reak)` *`<line>`*
Break at a line of the current file

`b` *`<file>`*`:`*`<line>`*
Break at a line of another file

`b` *`<line>`*`,` *`<cond>`*
Break only when the condition is true

`b`
List breakpoints with hit counts

INSPECT

`p` *`<expr>`*
Print an expression

`pp` *`<expr>`*
Pretty-print it (dicts, nested data)

`l(ist)`
11 lines around here, again continues

`ll`
Whole source of the current function