pdb Python debugger

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

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).

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

import pdb; pdb.set_trace()
Same, the pre-3.7 way

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

python -m pdb -m *<module>*
Debug a module, like python -m

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

START DEBUGGING (cont.)

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

AFTER A CRASH

python -i *<script>*
Keep the REPL open after an exception

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

s(tep)
Run the line, stepping into calls

r(eturn)
Run until the current function returns

STEP AND RUN (cont.)

c(ontinue)
Run until the next breakpoint

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

unt(il) *<line>*
Run to a line in this frame

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

u *<n>* **/ d** *<n>*
Move n frames at once

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

BREAKPOINTS (cont.)

b *<func>*
Break at the first line of a function

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

b
List breakpoints with hit counts

tbreak *<line>*
One-shot breakpoint, deleted on first hit

cl(ear) *<n>* **[***<n>* **...]**
Delete breakpoints by number

cl
Delete all breakpoints (asks first)

disable *<n>* **/ enable** *<n>*
Turn a breakpoint off / on

condition *<n>* **[***<cond>***]**
Change a condition; none removes it

INSPECT

p *<expr>*
Print an expression

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

INSPECT (cont.)

whatis *<expr>*
Show the type

display *<expr>*
Show it whenever it changes at a stop

retval
Value the function is returning (rv)

l(ist)
11 lines around here, again continues

l .
List around the current line again

ll
Whole source of the current function

CHANGE AND RUN CODE

!*<stmt>*
Run a Python statement in this frame

<var> **=** *<value>*
Change a variable, then go on

interact
Full Python REPL with the frame's names

h(elp) [*<cmd>***]**
List commands or explain one