Python Debugger Cheatsheet

Getting started

import pdb;pdb.set_trace()	start pdb from within a script
python -m pdb <file.py>	start pdb from the commandline

Basics

h(elp)	print available commands
h(elp) ***command***	print help about *command*
q(uit)	quit debugger

Examine

p(rint) ***expr***	print the value of *expr*
pp ***expr***	pretty-print the value of *expr*
w(here)	print current position (including stack trace)
l(ist)	list 11 lines of code around the current line
l(ist) ***first, last***	list from *first* to *last* line number
a(rgs)	print the args of the current function

Miscellaneous

!*stmt*	treat *stmt* as a Python statement instead of a pdb command
alias ***map stmt***	map Python statement as a map command
alias ***map <arg1 ...> stmt***	pass arguments to Python statement. *stmt* includes %1, %2, ... literals.

Save pdb commands to local <./.pdbrc> file for repetitive access.

Movement

<ENTER>	repeat the last command
n(ext)	execute the current statement (step over)
s(tep)	execute and step into function
r(eturn)	continue execution until the current function returns
c(ontinue)	continue execution until a breakpoint is encountered
u(p)	move one level up in the stack trace
d(own)	move one level down in the stack trace

Breakpoints

b(reak)	show all breakpoints with its *number*
b(reak) ***lineno***	set a breakpoint at *lineno*
b(reak) ***lineno, cond***	stop at breakpoint *lineno* if Python condition *cond* holds, e.g. i==42
b(reak) ***file:lineno***	set a breakpoint in *file* at *lineno*
b(reak) ***func***	set a breakpoint at the first line of a *func*
tbreak ***lineno***	set a temporary breakpoint at *lineno*, i.e. is removed when first hit
disable ***number***	disable breakpoint *number*
enable ***number***	enable breakpoint *number*
clear ***number***	delete breakpoint *number*

Author: Florian Preinstorfer (nblock@archlinux.us) — version 1.1 —
See `https://github.com/nblock/pdb-cheatsheet` for more information.