pdb Command Reference

These pdb commands and their syntax and descriptions are from the Python 3.6 documentation.

Command	Syntax / Description
a	a(rgs) Print the argument list of the current function.
alias	alias [name [command [parameter parameter ...]]] Create an alias called 'name' that executes 'command'. The command must *not* be enclosed in quotes. Replaceable parameters can be indicated by %1, %2, and so on, while %* is replaced by all the parameters. If no command is given, the current alias for name is shown. If no name is given, all aliases are listed. Aliases may be nested and can contain anything that can be legally typed at the pdb prompt. Note! You *can* override internal pdb commands with aliases! Those internal commands are then hidden until the alias is removed. Aliasing is recursively applied to the first word of the command line; all other words in the line are left alone. As an example, here are two useful aliases (especially when placed in the .pdbrc file): # Print instance variables (usage "pi classInst") alias pi for k in %1.__dict__.keys(): print("%1.",k,"=",%1.__dict__[k]) # Print instance variables in self alias ps pi self
args	a(rgs) Print the argument list of the current function.
b	b(reak) [([filename:]lineno | function) [, condition]] Without argument, list all breaks. With a line number argument, set a break at this line in the current file. With a function name, set a break at the first executable line of that function. If a second argument is present, it is a string specifying an expression which must evaluate to true before the breakpoint is honored. The line number may be prefixed with a filename and a colon, to specify a breakpoint in another file (probably one that hasn't been loaded yet). The file is searched for on sys.path; the .py suffix may be omitted.

Command	Syntax / Description
break	b(reak) [([filename:]lineno | function) [, condition]] Without argument, list all breaks. With a line number argument, set a break at this line in the current file. With a function name, set a break at the first executable line of that function. If a second argument is present, it is a string specifying an expression which must evaluate to true before the breakpoint is honored. The line number may be prefixed with a filename and a colon, to specify a breakpoint in another file (probably one that hasn't been loaded yet). The file is searched for on sys.path; the .py suffix may be omitted.
bt	w(here) Print a stack trace, with the most recent frame at the bottom. An arrow indicates the "current frame", which determines the context of most commands. 'bt' is an alias for this command.
c	c(ont(inue)) Continue execution, only stop when a breakpoint is encountered.
cl	cl(ear) filename:lineno cl(ear) [bpnumber [bpnumber...]] With a space separated list of breakpoint numbers, clear those breakpoints. Without argument, clear all breaks (but first ask confirmation). With a filename:lineno argument, clear all breaks at that line in that file.
clear	cl(ear) filename:lineno cl(ear) [bpnumber [bpnumber...]] With a space separated list of breakpoint numbers, clear those breakpoints. Without argument, clear all breaks (but first ask confirmation). With a filename:lineno argument, clear all breaks at that line in that file.

Command	Syntax / Description
commands	commands [bpnumber] (com) ... (com) end (Pdb) Specify a list of commands for breakpoint number bpnumber. The commands themselves are entered on the following lines. Type a line containing just 'end' to terminate the commands. The commands are executed when the breakpoint is hit. To remove all commands from a breakpoint, type commands and follow it immediately with end; that is, give no commands. With no bpnumber argument, commands refers to the last breakpoint set. You can use breakpoint commands to start your program up again. Simply use the continue command, or step, or any other command that resumes execution. Specifying any command resuming execution (currently continue, step, next, return, jump, quit and their abbreviations) terminates the command list (as if that command was immediately followed by end). This is because any time you resume execution (even with a simple next or step), you may encounter another breakpoint -- which could have its own command list, leading to ambiguities about which list to execute. If you use the 'silent' command in the command list, the usual message about stopping at a breakpoint is not printed. This may be desirable for breakpoints that are to print a specific message and then continue. If none of the other commands print anything, you will see no sign that the breakpoint was reached.
condition	condition bpnumber [condition] Set a new condition for the breakpoint, an expression which must evaluate to true before the breakpoint is honored. If condition is absent, any existing condition is removed; i.e., the breakpoint is made unconditional.
cont	c(ont(inue)) Continue execution, only stop when a breakpoint is encountered.
continue	c(ont(inue)) Continue execution, only stop when a breakpoint is encountered.
d	d(own) [count] Move the current frame count (default one) levels down in the stack trace (to a newer frame).
debug	debug code Enter a recursive debugger that steps through the code argument (which is an arbitrary expression or statement to be executed in the current environment).

Command	Syntax / Description
disable	disable bpnumber [bpnumber ...] Disables the breakpoints given as a space separated list of breakpoint numbers. Disabling a breakpoint means it cannot cause the program to stop execution, but unlike clearing a breakpoint, it remains in the list of breakpoints and can be (re-)enabled.
display	display [expression] Display the value of the expression if it changed, each time execution stops in the current frame. Without expression, list all display expressions for the current frame.
down	d(own) [count] Move the current frame count (default one) levels down in the stack trace (to a newer frame).
enable	enable bpnumber [bpnumber ...] Enables the breakpoints given as a space separated list of breakpoint numbers.
EOF	EOF Handles the receipt of EOF as a command.
exit	q(uit) exit Quit from the debugger. The program being executed is aborted.
h	h(elp) Without argument, print the list of available commands. With a command name as argument, print help about that command. "help pdb" shows the full pdb documentation. "help exec" gives help on the ! command.
help	h(elp) Without argument, print the list of available commands. With a command name as argument, print help about that command. "help pdb" shows the full pdb documentation. "help exec" gives help on the ! command.
ignore	ignore bpnumber [count] Set the ignore count for the given breakpoint number. If count is omitted, the ignore count is set to 0. A breakpoint becomes active when the ignore count is zero. When non-zero, the count is decremented each time the breakpoint is reached and the breakpoint is not disabled and any associated condition evaluates to true.

Command	Syntax / Description
interact	interact Start an interactive interpreter whose global namespace contains all the (global and local) names found in the current scope.
j	j(ump) lineno Set the next line that will be executed. Only available in the bottom-most frame. This lets you jump back and execute code again, or jump forward to skip code that you don't want to run. It should be noted that not all jumps are allowed -- for instance it is not possible to jump into the middle of a for loop or out of a finally clause.
jump	j(ump) lineno Set the next line that will be executed. Only available in the bottom-most frame. This lets you jump back and execute code again, or jump forward to skip code that you don't want to run. It should be noted that not all jumps are allowed -- for instance it is not possible to jump into the middle of a for loop or out of a finally clause.
l	l(ist) [first [,last] | .] List source code for the current file. Without arguments, list 11 lines around the current line or continue the previous listing. With . as argument, list 11 lines around the current line. With one argument, list 11 lines starting at that line. With two arguments, list the given range; if the second argument is less than the first, it is a count. The current line in the current frame is indicated by "->". If an exception is being debugged, the line where the exception was originally raised or propagated is indicated by ">>", if it differs from the current line.
list	l(ist) [first [,last] | .] List source code for the current file. Without arguments, list 11 lines around the current line or continue the previous listing. With . as argument, list 11 lines around the current line. With one argument, list 11 lines starting at that line. With two arguments, list the given range; if the second argument is less than the first, it is a count. The current line in the current frame is indicated by "->". If an exception is being debugged, the line where the exception was originally raised or propagated is indicated by ">>", if it differs from the current line.
ll	longlist | ll List the whole source code for the current function or frame.
longlist	longlist | ll List the whole source code for the current function or frame.

Command	Syntax / Description
n	n(ext) Continue execution until the next line in the current function is reached or it returns.
next	n(ext) Continue execution until the next line in the current function is reached or it returns.
p	p expression Print the value of the expression.
pp	pp expression Pretty-print the value of the expression.
q	q(uit) exit Quit from the debugger. The program being executed is aborted.
quit	q(uit) exit Quit from the debugger. The program being executed is aborted.
r	r(eturn) Continue execution until the current function returns.
restart	run [args...] Restart the debugged python program. If a string is supplied it is split with "shlex", and the result is used as the new sys.argv. History, breakpoints, actions and debugger options are preserved. "restart" is an alias for "run".
return	r(eturn) Continue execution until the current function returns.
retval	retval Print the return value for the last return of a function.
run	run [args...] Restart the debugged python program. If a string is supplied it is split with "shlex", and the result is used as the new sys.argv. History, breakpoints, actions and debugger options are preserved. "restart" is an alias for "run".

Command	Syntax / Description
rv	retval Print the return value for the last return of a function.
s	s(tep) Execute the current line, stop at the first possible occasion (either in a function that is called or in the current function).
source	source expression Try to get source code for the given object and display it.
step	s(tep) Execute the current line, stop at the first possible occasion (either in a function that is called or in the current function).
tbreak	tbreak [([filename:]lineno | function) [, condition]] Same arguments as break, but sets a temporary breakpoint: it is automatically deleted when first hit.
u	u(p) [count] Move the current frame count (default one) levels up in the stack trace (to an older frame).
unalias	unalias name Delete the specified alias.
undisplay	undisplay [expression] Do not display the expression any more in the current frame. Without expression, clear all display expressions for the current frame.
unt	unt(il) [lineno] Without argument, continue execution until the line with a number greater than the current one is reached. With a line number, continue execution until a line with a number greater or equal to that is reached. In both cases, also stop when the current frame returns.
until	unt(il) [lineno] Without argument, continue execution until the line with a number greater than the current one is reached. With a line number, continue execution until a line with a number greater or equal to that is reached. In both cases, also stop when the current frame returns.

Command	Syntax / Description
up	u(p) [count] Move the current frame count (default one) levels up in the stack trace (to an older frame).
w	w(here) Print a stack trace, with the most recent frame at the bottom. An arrow indicates the "current frame", which determines the context of most commands. 'bt' is an alias for this command.
whatis	whatis arg Print the type of the argument.
where	w(here) Print a stack trace, with the most recent frame at the bottom. An arrow indicates the "current frame", which determines the context of most commands. 'bt' is an alias for this command.