



## PDB INVOCATION

Invoking		Module pdb	
<code>python -m pdb file.py</code>	From command line	<code>run(stmt)</code>	Execute <i>stmt</i> in debugger
<code>import pdb;</code> <code>pdb.set_trace()</code>	Within a script (hard-coded breakpoint)	<code>runeval(expr)</code>	Execute <i>expr</i> in debugger and return value
<code>%pdb</code> or <code>%debug</code>	iPython	<code>runcall(fn)</code>	Call <i>fn</i> with given arguments in debugger and return value
		<code>set_trace()</code>	Enter debugger at calling stack frame (hard-code breakpoint)
		<code>post_mortem(), pm()</code>	Enter post-mortem debugging of preceding traceback

## PDB OPERATION

Navigating			
	Repeat last command	<code>d[own]</code>	Move down a level in stack trace
<code>n[ext]</code>	Continue until next line in current function is reached or it returns	<code>j[ump] lineno</code>	Set the next line that will be executed (in current frame)
<code>s[tep]</code>	Execute the current line, stop at the first possible occasion	<code>unt[il]</code>	Continue until reaching greater line number
<code>r[eturn]</code>	Continue until current function returns	<code>w[here]</code>	Print stack trace with arrow to current frame
<code>c[ontinue]</code>	Continue; stop only at breakpoints	<code>l[ist]</code>	List source code for the current file (11 lines around current line)
<code>u[p]</code>	Move up a level in stack trace	<code>l[ist] start, end</code>	List source code for the current file (specific lines)
Breakpoints		Manipulating	
<code>b[reak]</code>	List all breaks including number of times each breakpoint has been hit	<code>a[rgs]</code>	Print argument list of current function
<code>b[reak] lineno</code> <code>b[reak] fn</code>	Set a break at <i>lineno</i> in current file or at first executable line in <i>fn</i>	<code>p[rint] expr</code>	Evaluate <i>expr</i> in current context and print its value (  completion)
<code>cl[ear]</code>	Like <code>b[reak]</code> to clear breakpoints	<code>pp expr</code>	Like <code>p[rint]</code> to pretty-print expressions
<code>condition bpno</code> <code>[cond]</code>	Set or clear condition for breakpoint <i>bpno</i> ( <i>cond</i> must be true to stop)	<code>source expr</code>	Try to get and display source code for object <i>expr</i>
<code>commands bpno</code>	Specify PDB commands to execute at breakpoint <i>bpno</i> (end with <i>end</i> )	<code>display expr</code>	Display value of <i>expr</i> (if changed) when exctn stops in current frame
<code>ignore bpno</code> <code>[count]</code>	Ignore breakpoint <i>bpno</i> for <i>count</i> times or until reenabled	<code>run [args...]</code>	Restart debugged program (with arguments <i>args</i> )
Executing			
<code>[!] expr</code>	Evaluate <i>expr</i> in current stack frame (be careful in iPython)	<code>q[uit]</code>	Quit pdb; abort program