Python Debugging Cheat Sheet

Setting a Breakpoint

Add `breakpoint()` where you want to pause. Works in `.py` files and Jupyter notebooks.

```
def calculate_average(scores):
    total = sum(scores)
    breakpoint()  # program pauses here
    return total / len(scores)
```

Run your script normally (`python my_script.py`) or run the Jupyter cell. When Python hits the breakpoint, you get a `(Pdb)` prompt.

Jupyter tip: Type `%debug` in the next cell after an error to inspect the crash. Always type `q` to exit or the notebook will freeze.

Essential pdb Commands

Command	Short	What It Does
`next`	`n`	Run current line, stop at next line (stays in current function)
`step`	`s`	Run current line, go *into* function calls
`continue`	`c`	Resume until next breakpoint or end
`return`	`r`	Run until current function returns
`until`	`unt`	Skip to end of current loop
`print(expr)`	`p expr`	Print a variable or expression
`list`	`l`	Show code around current line
`longlist`	`ll`	Show entire current function
`args`	`a`	Show current function's arguments
`where`	`w`	Show the call stack
`break N`	`b N`	Set breakpoint at line N
`quit`	`q`	Exit the debugger
`!x = 5`		Change a variable's value

Typical Workflow

```
$ python my_script.py
> my_script.py(3)calculate_average()
-> return total / len(scores)
(Pdb) p total        # check variable values
265
(Pdb) p len(scores)  # find the problem
0
(Pdb) q              # quit when done
```

1. Place `breakpoint()` near the suspected bug
2. Use `p` to check variables -- the bug is usually a value you didn't expect
3. Use `n` to step through line by line
4. Remove `breakpoint()` when finished