Python pdb使用方法

如果你还主要靠print来调试代码，那值得花10分钟试试pdb这个Python自带的Debug工具。

pdb有2种用法：

* **非侵入式方法**（不用额外修改源代码，在命令行下直接运行就能调试）

python3 -m pdb filename.py

* **侵入式方法**（需要在被调试的代码中添加一行代码然后再正常运行代码）

**import** pdb;pdb**.**set\_trace()

当你在命令行看到下面这个提示符时，说明已经正确打开了pdb

(Pdb)

然后就可以开始输入pdb命令了，下面是pdb的常用命令

**1、查看源代码**

命令：

l

说明：

查看当前位置前后11行源代码（多次会翻页）  
当前位置在代码中会用-->这个符号标出来

命令：

ll

说明：

查看当前函数或框架的所有源代码

**2、添加断点**

命令：

b

b lineno

b filename:lineno

b functionname

参数：

filename文件名，断点添加到哪个文件，[如test.py](https://link.zhihu.com/?target=http%3A//xn--test-f96g.py/)  
lineno断点添加到哪一行  
function：函数名，在该函数执行的第一行设置断点

说明：

1.不带参数表示查看断点设置  
2.带参则在指定位置设置一个断点

**3、添加临时断点**

命令：

tbreak

tbreak lineno

tbreak filename:lineno

tbreak functionname

参数：

同b

说明：

执行一次后时自动删除（这就是它被称为临时断点的原因）

**4、清除断点**

命令：

cl

cl filename:lineno

cl bpnumber [bpnumber ...]

参数：

bpnumber 断点序号（多个以空格分隔）

说明：

1.不带参数用于清除所有断点，会提示确认（包括临时断点）  
2.带参数则清除指定文件行或当前文件指定序号的断点

**5、打印变量值**

命令：

p expression

参数：

expression Python表达式

**6、逐行调试命令**

包括 s ，n ， r 这3个相似的命令，区别在如何对待函数上

命令1：

s

说明：

执行下一行（能够进入函数体）

命令2：

n

说明：

执行下一行（不会进入函数体）

命令3：

r

说明：

执行下一行（在函数中时会直接执行到函数返回处）

**7、非逐行调试命令**

命令1：

c

说明：

持续执行下去，直到遇到一个断点

命令2

unt lineno

说明：

持续执行直到运行到指定行（或遇到断点）

命令3

j lineno

说明：

直接跳转到指定行（注意，被跳过的代码不执行）

**8、查看函数参数**

命令：

a

说明：

在函数中时打印函数的参数和参数的值

**9、打印变量类型**

命令：

whatis expression

说明：

打印表达式的类型，常用来打印变量值

**10、启动交互式解释器**

interact

说明：

启动一个python的交互式解释器，使用当前代码的全局命名空间（使用ctrl+d返回pdb）

**11、打印堆栈信息**

w

说明：

打印堆栈信息，最新的帧在最底部。箭头表示当前帧。

**12、退出pdb**

q

完成了。好吧，可能超过了10分钟，我承认这是一个善意的谎言，不过至此你已经掌握了，击个掌吧。