

ASIMOV

Python Debugger

Você provavelmente usou uma variedade de instruções de impressão para tentar encontrar erros em seu código. Uma maneira melhor de fazer isso é usando o módulo de depuração incorporado do Python (pdb). O módulo pdb implementa um ambiente de depuração interativo para programas Python. Ele inclui recursos para permitir que você pause seu programa, veja os valores das variáveis e assista a execução do programa passo a passo, para que você possa entender o que o seu programa realmente faz e encontrar erros na lógica.

Isso é um pouco difícil de mostrar, uma vez que requer criar um erro de propósito, mas espero que este exemplo simples ilustre o poder do módulo pdb. *Nota: tenha em mente que seria bastante incomum usar o pdb em uma configuração do iPython Notebook.*

Aqui vamos criar um erro de propósito, tentando adicionar uma lista a um número inteiro

In [1]:

```
x = [1,3,4]
y = 2
z = 3

result = y + z
print(result)
result2 = y+x
print(result2)
```

5

```
-----
TypeError                                Traceback (most recent call last)
<ipython-input-1-905e8cfe6928> in <module>()
      5 result = y + z
      6 print(result)
----> 7 result2 = y+x
      8 print(result2)
```

TypeError: unsupported operand type(s) for +: 'int' and 'list'

Hmmm, parece que temos um erro! Vamos implementar um `set_trace()` usando o módulo pdb. Isso nos permitirá basicamente pausar o código no ponto do rastreamento e verificar se algo está errado.

In [4]:

```
import pdb

x = [1,3,4]
y = 2
z = 3

result = y + z
print result
```

```
# Usa o método set_trace() para pausar o código neste ponto.
pdb.set_trace()

result2 = y+x
print result2
```

```
5
--Return--
> <ipython-input-4-0a2880872cf0>(11)<module>()->None
-> pdb.set_trace()
(Pdb) x
[1, 3, 4]
(Pdb) y
2
(Pdb) z
3
(Pdb) x+y
*** TypeError: can only concatenate list (not "int") to list
(Pdb) q

-----
BdbQuit                                Traceback (most recent call last)
<ipython-input-4-0a2880872cf0> in <module>()
      9
     10 # Set a trace using Python Debugger
----> 11 pdb.set_trace()
     12
     13 result2 = y+x

//anaconda/lib/python2.7/bdb.py in trace_dispatch(self, frame, event, arg)
     51         return self.dispatch_call(frame, arg)
     52         if event == 'return':
----> 53         return self.dispatch_return(frame, arg)
     54         if event == 'exception':
     55         return self.dispatch_exception(frame, arg)

//anaconda/lib/python2.7/bdb.py in dispatch_return(self, frame, arg)
     89         finally:
     90             self.frame_returning = None
----> 91         if self.quitting: raise BdbQuit
     92         return self.trace_dispatch
     93

BdbQuit:
```

Ótimo! Agora, poderíamos verificar quais eram as variáveis e verificar erros. Você pode usar 'q' para sair do depurador. Para obter mais informações sobre técnicas gerais de depuração e mais métodos, confira a documentação oficial: <https://docs.python.org/2/library/pdb.html>