Black Hat Python

Python Programming for Hackers and Pentesters

Justin Seitz

Foreword by Charlie Miller

no starch press

Capítulo 1 - Configurando seu ambiente Python

Essa parte é a menos divertida, mas mesmo assim crítica do livro, Onde nós passamos pela configuração de um ambiente para escrever e testar em Python. Vamos fazer um curso intensivo sobre como configurar uma máquina virtual (VM) com Kali Linux e instalar uma boa IDE para que você tenha tudo que precisa para desenvolver códigos. Ao final do capítulo, você deve estar pronto para fazer os exercícios e os exemplos de código pelo restante do livro.

Antes de começar, vamos baixar e Instalar o VMWare Player.[1] Também recomendo que você tenha algumas máquina virtual Windows prontas, incluindo Windows XP e Windows 7, de preferência de 32 bits em ambos os casos

1. Você pode baixar o VMWare Player de: http://www.vmware.com/.

Instalando o Kali Linux

Kali é o sucessor da distribuição do BackTrack Linux, designado para Defesa ofensiva desde o início como um sistema operacional para teste de penetração. Vem um com um número de ferramentas pré-instaladas e é baseado no Debian Linux, então será possível instalar uma grande variedade de ferramentas e bibliotecas adicionais além do que esta no sistema operacional.

Primeiro, pegue a imagem da máquina virtual Kali na seguinte URL: http://images.offensive-security.com/kali-linux-1.0.9-vm-i486.7z.[2] Baixe e descomprima a image, e então, dê dois cliques para o VMWare Player abrí-la. O Usuário padrão é *root* e a senha é *toor*. Isso pode colocar você dentro do ambiente Kali completo, como mostrado na Figura 1-1.

Figura 1-1: Área de trabalho Kali Linux

A primeira coisa que precisamos fazer é certificar que estamos com a correta versão de Python instalada. Nesse livro vamos usar o Python 2.7. No shell (**Aplicações > Acessórios > Terminal**), execute o seguinte comando:

```
root@kali:~# python --version
Python 2.7.3
root@kali:~#
```

2. Para uma lista "clicável" de links nesse capítulo, visite http://nostarch.com/blackhatpython/.

Se você baixou a exata imagem que recomendei acima, Python 2.7 já estará automaticamente instalado. Por favor note que usando uma versão diferente de Python deve quebrar alguns

exemplos de código desse livro. Você foi avisado.

Agora vamos adicionar algumas peças úteis de gerenciamento de pacotes Python chamados de `easy_install` e `pip`. Esses são muito parecidos com o gerenciador de pacotes `apt` pois eles permitem você Instalar bibliotecas Python diretamente, sem ter baixado manualmente, desempacotar, e instalar eles. Vamos instalar ambos os pacotes com os comandos a seguir:

```
root@kali:~#: apt-get install python-setuptools python-pip
```

Quando os pacotes estiverem instalados, podemos fazer um rápido teste e instalar o módulo que usaremos no Capítulo 7 para construir um trojan baseado em GitHub. Entre com este código no seu terminal:

```
root@kali:~#: pip install github3.py
```

Você deve ver que a saída do seu terminal indicou que a biblioteca começou a baixar e instalar.

E então, entre no terminal Python e valide se foi corretamente instalada:

```
root@kali:~#: python
Python 2.7.3 (default, Mar 14 2014, 11:57:14)
[GCC 4.7.2] on linux2 Type "help", "copyright", "credits" or "license" for
more information.
>>> import github3
>>> >>> exit()
```

Se os seus resultados não foram idênticos a esses, então houve um "erro de configuração" no seu ambiente Python e você deve trouxe uma grande vergonha para o nosso dojo de Python! Nesse caso, certifique-se que você seguiu todos os passos acima e que você está com a versão correta do Kali.

Tenha em mente que para a maioria dos exemplos desse livro, você pode desenvolver seu código em ambientes variados, incluindo Mac, Linux e Windows. Existem alguns capítulos que serão Específicos para Windows, e eu vou te informar ao começo desses capítulos.

Agora que nós temos uma máquina virtual para hacking feita, vamos instalar uma IDE de Python para desenvolvimento.

WingIDE

Embora eu normalmente não não defenda produtos de softwares comerciais, WingIDE é a melhor IDE que usei nos últimos sete anos na Immunity. WingIDE lhe dá todas as funcionalidades básicas de IDE, como *auto-completion* e explicação dos parâmetros de função, mas seus recursos de *debugging* são o seu diferencial além de outras IDEs. Eu vou dar a você um rápido resumo da versão comercial do WingIDE, mas é claro que você deve escolher a melhor versão para você.[3]

Você pode conseguir o WingIDE pelo link: http://www.wingware.com/, e recomendo que você instale a versão de teste para que possa experenciar de primeira mão algumas ferramentas disponíveis na versão comercial.

Você pode fazer o seu desenvolvimento em qualquer plataforma que deseja, mas pode ser melhor instalar o WingIDE na sua máquina virtual Kali, pelo menos de começo. Se você seguiu minhas instruções até agora, tenha certeza que baixou o pacote `.deb` de 32 bits do WingIDE, e salve dentro do seu diretório de usuário.
Então, entre no terminal e coloque:

```
root@kali:~# dpkg -i wingide5_5.0.9-1_i386.deb
```

Isso deve instalar o WingIDE como o planejado. Se você teve qualquer erro na instalação, provavelmente houve dependências não atendidas. Nesse caso, rode:

```
root@kali:~# apt-get -f install
```

O comando deve arrumar qualquer dependência em falta e instalar o WingIDE. Para verificar se você instalou ele corretamente, certifique-se que consegue acessá-lo como mostrado na Figura 1-2.

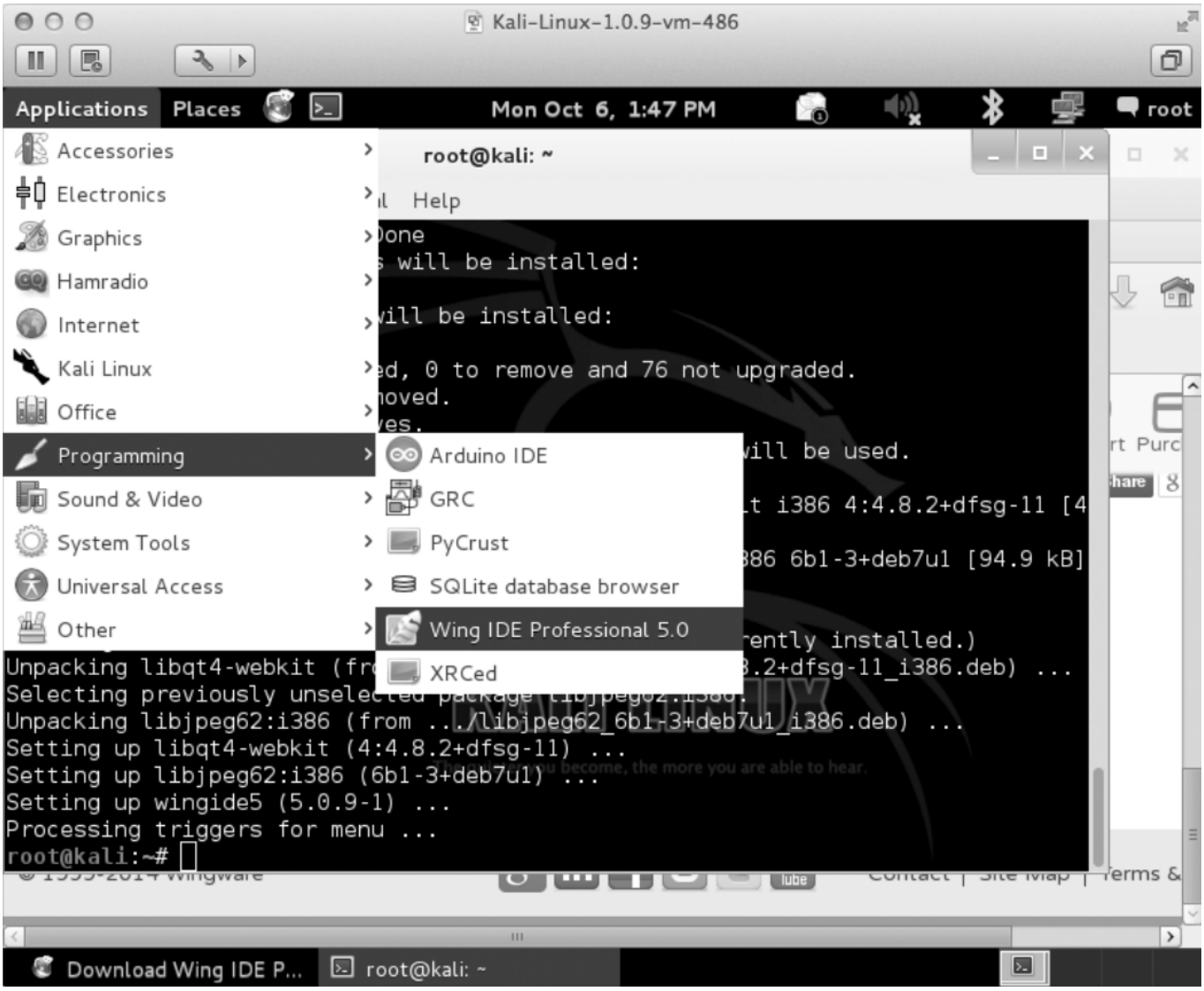

Figure 1-2: Acessando WingIDE via VM Kali

3. Para comparação de ferramentas entre versões, visite https://wingware.com/wingide/features/

Entre no WingIDE e abra um novo arquivo Python em branco. E então acompanhe enquanto apresento um rápido resumo de algumas ferramentas úteis. Para começar, sua tela irá se parecer com a Figura 1-3, com sua área principal de código no canto superior esquerdo e um conjunto de janelas na parte na parte inferior.

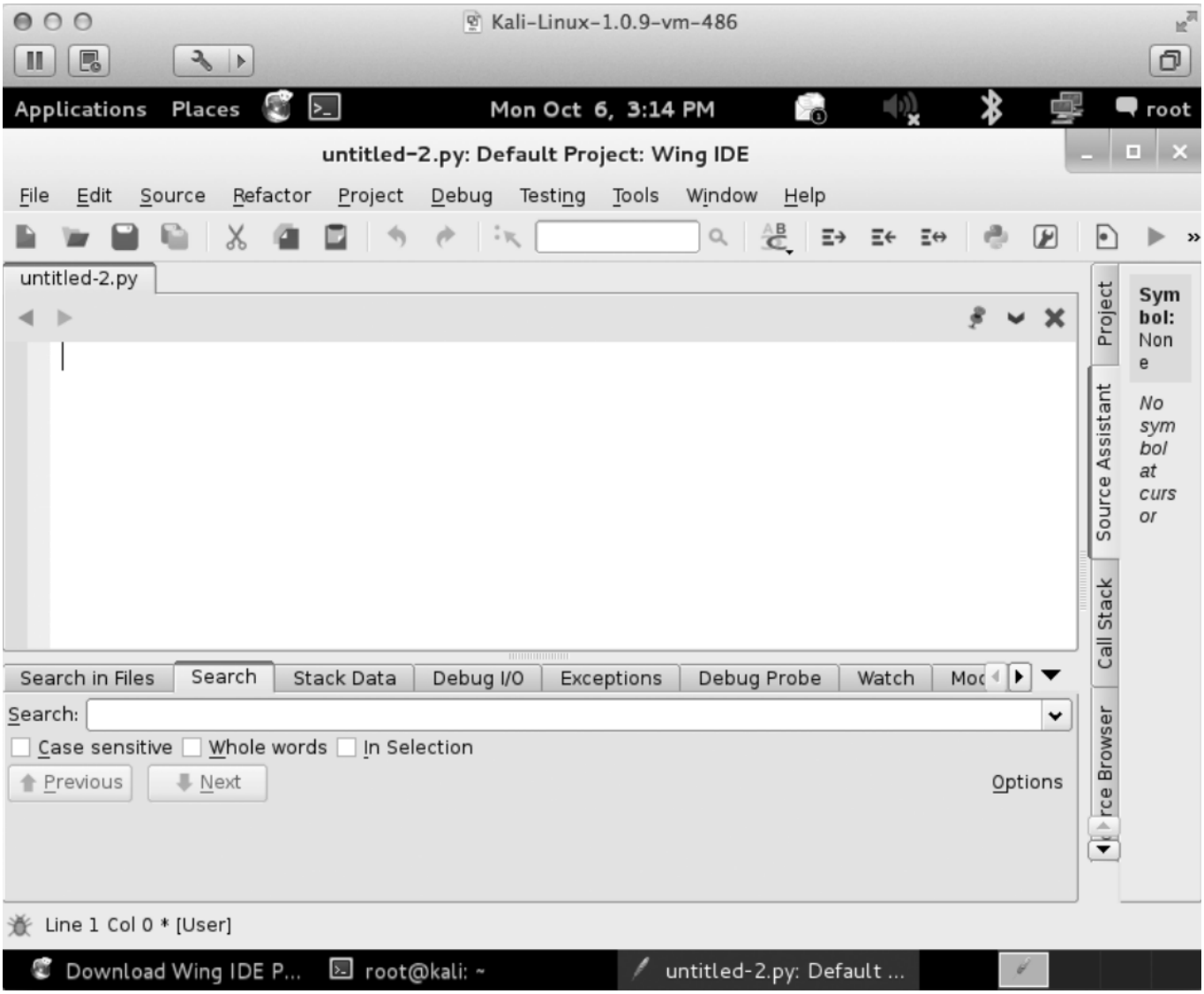

Figure 1-3: Layout da janela principal do WingIDE

Vamos escrever um código simples para ilustrar algumas funções úteis do WingIDE, incluindo as guias *Debug Probe* e *Stack Data*. Insira o código a seguir no editor:

```
def sum(number_one,number_two):
        number_one_int = convert_integer(number_one)
        number_two_int = convert_integer(number_two)

        result = number_one_int + number_two_int

        return result

def convert_integer(number_string):

        converted_integer = int(number_string)
        return converted_integer
```

```
answer = sum("1","2")
```

Este é um exemplo bem inventado, mas é uma excelente demonstração de como WingIDE facilita sua vida. Salve o arquivo com qualquer nome que queira, clique no menu ***Debug***, e selecione a opção ***Select Current as Main Debug File***, como mostrado na Figura 1-4.

Figure 1-4: Configurando o script atual para debugging

Agora, coloque um ponto de parada (*breakpoint*) na linha que está escrito:

```
        return converted_integer
```

Você pode fazer isso clicando na margem a esquerda ou pressionando a tecla F9. Você deve ver que um pequeno ponto vermelho aparece na margem. Agora rode o script pressionando

F5, e a execução deve pausar no ponto de parada. Clique no menu ***Stack Data*** e você deve ver uma tela parecida com a Figura 1-5.

O menu de pilha de dados está mostrando para nós algumas informações importantes, como o estado de qualquer variável local e global no momento que o ponto de parada foi alcançado. Isso permite você depurar os códigos mais avançados onde você precisa inspecionar variáveis durante a execução para resolver *bugs*. Se você clicar na barra suspensa, você também pode ver a chamada atual da pilha, qua mostra qual função chama a função atual que você está. Dê uma olhada na Figura 1-6 para ver o caminho da pilha.

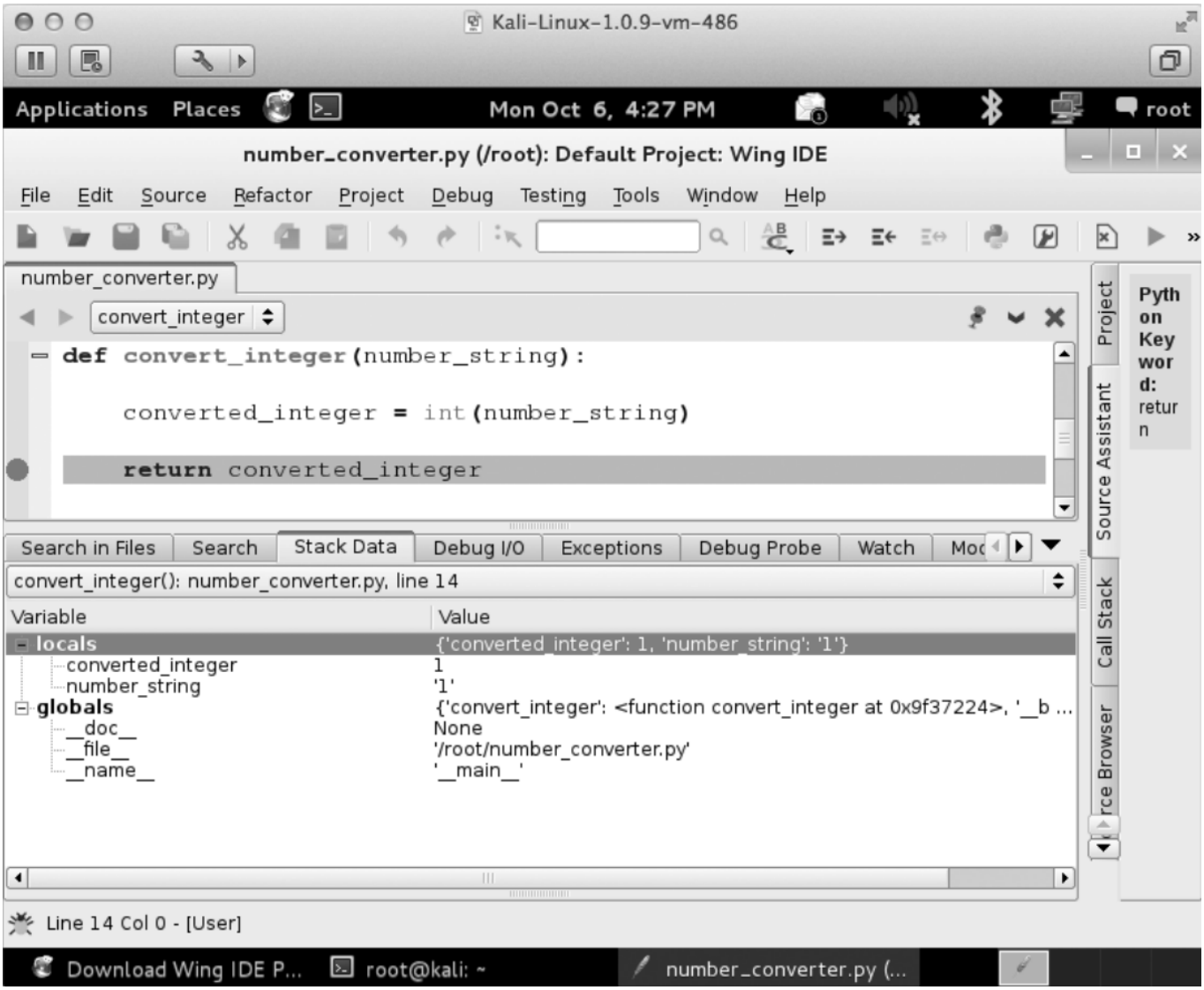

Figura 1-5: Vendo a pilha de dados depois de alcançar o ponto de parada

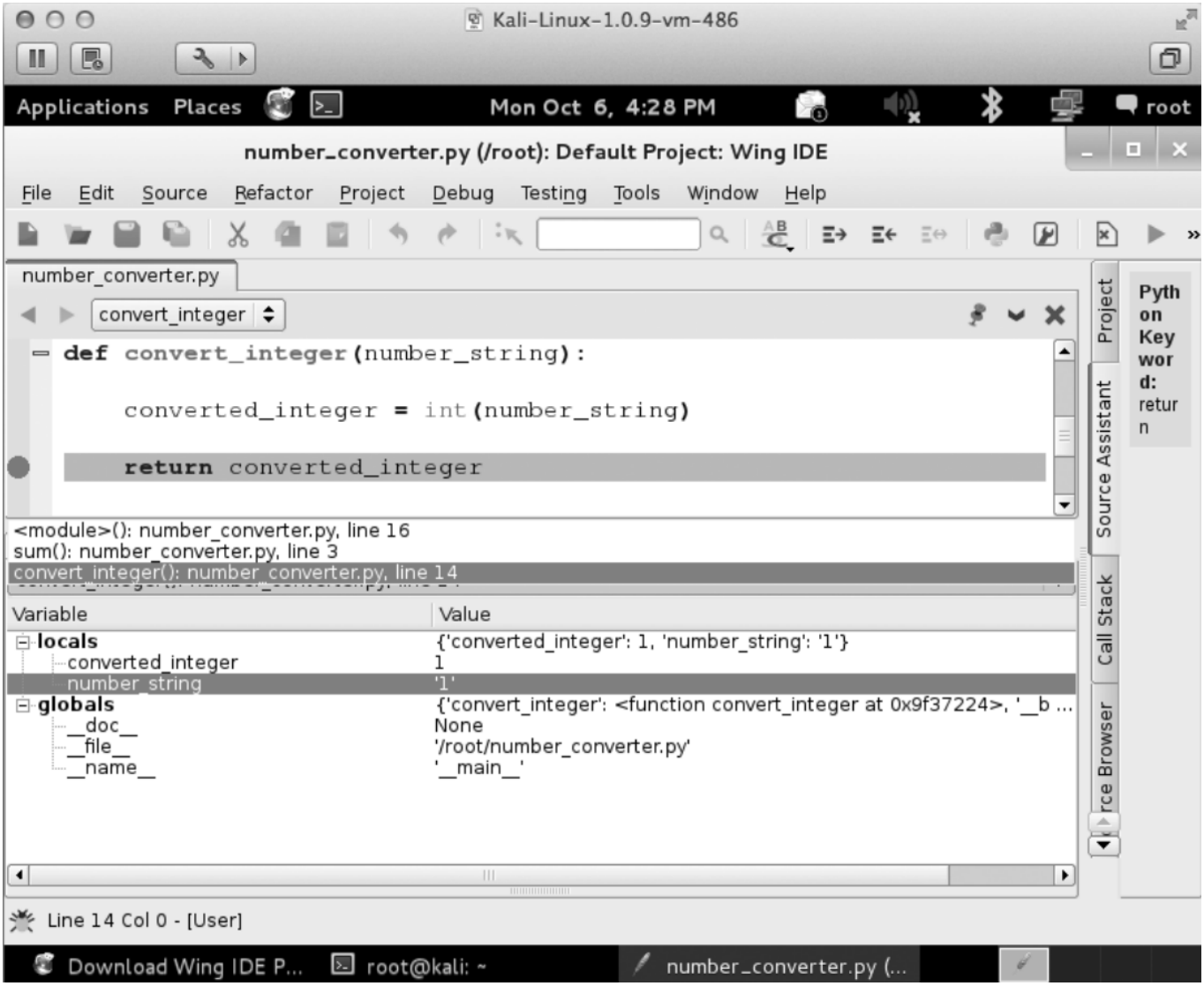

Figura 1-6: Vendo o caminho da pilha

Nós podemos perceber que `convert_integer` foi chamado pela função `sum` na linha 3 do nosso script em Python. Isso se torna muito útil se você tem recursivas chamadas de função ou uma função que é chamada de vários lugares. Usar a pilha de dados será muito útil na sua carreira de desenvolvimento em Python!

A próxima ferramenta é o menu *Debug Probe*. Esse menu permite que você entre em um terminal Python que está executando dentro o contexto atual do exato momento que seu ponto de parada foi alcançado. Isso permite você inspecionar e modificar variáveis, bem como escrever pequenos trechos de teste de código para novas ideias ou para resolver problemas. A Figura 1-7 demonstra como inspecionar a variável `converted_integer` e mudar seu valor.

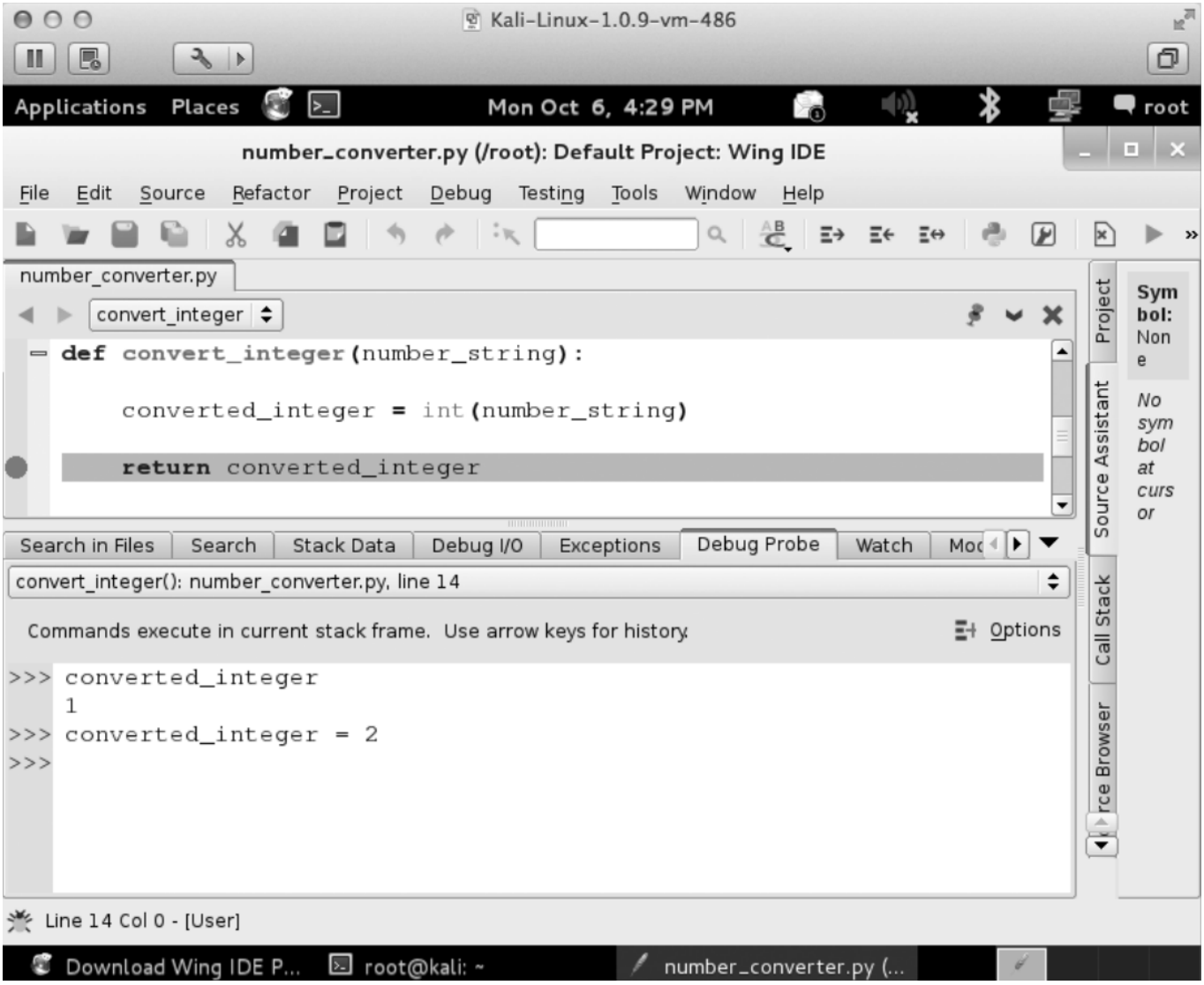

Figura 1-7: Usando o Debug Probe para inspecionar e modificar variáveis locais

Depois de fazer algumas modificações, você pode retomar a execução do script pressionando F5.

Mesmo isso sendo um exemplo muito simples, pode demonstrar algumas das ferramentas mais úteis do WingIDE para desenvolvimento e depuração de scripts em Python.[4]

Isso é tudo que precisamos para começar desenvolver nosso código pelo resto do livro. Não se esqueça de fazer máquinas virtuais prontas como alvo para os capítulos específicos de Windows, mas com certeza dispositivos nativos não apresentaram nenhum problema.

Agora, vamos entrar na verdadeira diversão!

4. Se você já usa uma IDE e deseja comparar as ferramentas dela com WingIDE, me mande um email ou tweet porque eu adoraria escutar sobre isso!