

# Beginning Ethical Hacking with Python

---

Sanjib Sinha

Sanjib Sinha

# Comenzando el Hacking Ético con Python

Apress®

---

Sanjib Sinha  
Howrah, Bengala Occidental, India

Cualquier código fuente u otro material suplementario referenciado por el autor en este texto está disponible para los lectores en [www.apress.com](http://www.apress.com). Para obtener información detallada sobre cómo localizar el código fuente de tu libro, ve a [www.apress.com/source-code/](http://www.apress.com/source-code/). Los lectores también pueden acceder al código fuente en SpringerLink en la sección Material suplementario de cada capítulo.

ISBN 978-1-4842-2540-0e-ISBN 978-1-4842-2541-7  
DOI 10.1007/978-1-4842-2541-7

Número de control de la Biblioteca del Congreso: 2016963222

Sanjib Sinha 2017

Este trabajo está sujeto a derechos de autor. El Editor se reserva todos los derechos, tanto si se trata de la totalidad como de una parte del material, en particular los derechos de traducción, reimpresión, reutilización de ilustraciones, recitación, radiodifusión, reproducción en microfilm o de cualquier otro modo físico, así como de transmisión o almacenamiento y recuperación de información, adaptación electrónica, software informático, o mediante una metodología similar o diferente ahora conocida o desarrollada en el futuro.

Los nombres, logotipos e imágenes de marcas registradas pueden aparecer en este libro. En lugar de utilizar un símbolo de marca comercial con cada aparición de un nombre, logotipo o imagen de marca comercial, utilizamos los nombres, logotipos e imágenes sólo de forma editorial y en beneficio del propietario de la marca comercial, sin intención de infringir la marca comercial. El uso en esta publicación de nombres comerciales, marcas registradas, marcas de servicio y términos similares, incluso si no están identificados como tales, no debe tomarse como una expresión de opinión sobre si están o no sujetos a derechos de propiedad.

Aunque el consejo y la información de este libro se consideran verdaderos y exactos en la fecha de publicación, ni los autores, ni los editores ni el editor pueden aceptar responsabilidad legal alguna por cualquier error u omisión que

se puede hacer. El editor no ofrece ninguna garantía, expresa o implícita, con respecto al material contenido aquí.

Impreso en papel sin ácido

Distribuido a todo el mundo por Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Llame al 1-800-SPRINGER, envíe un fax al (201) 348-4505, envíe un correo electrónico a [orders-ny@springer-sbm.com](mailto:orders-ny@springer-sbm.com) o visite [www.springeronline.com](http://www.springeronline.com) Apress Media, LLC es una LLC de California y el único miembro (propietario) es Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc es una corporación de Delaware.

---

*DR. AVIJIT SEN, DRISTIPRADIP, KOLKATA.*

*(Por traer la luz a las tinieblas)*

---

# **Prólogo - Hacker's Goal**

Este libro está dirigido a principiantes de programación o personas en general que no saben nada sobre ningún lenguaje de programación pero que quieren aprender hacking ético.

Despejémoslo primero: El Hacking Ético no está asociado a ningún tipo de actividad electrónica ilegal. Siempre están dentro de las leyes. Este libro está dirigido a aquellas personas -jóvenes y mayores- que son creativas y curiosas y que quieren desarrollar una afición creativa o dedicarse a la seguridad en Internet actuando como hackers éticos. Teniendo esto en cuenta, también aprenderemos el lenguaje de programación Python 3 para mejorar nuestra habilidad como hackers éticos.

Este libro no está destinado a ningún tipo de usuario malintencionado. Si alguien intenta usar este libro o cualquier tipo de ejemplos de código de este libro con fines ilegales, este libro no asumirá ninguna responsabilidad moral por esos comportamientos maliciosos.

Si cree que puede utilizar este libro para cualquier propósito malicioso, le aconsejamos que lea el primer capítulo "Legal Side of Ethical Hacking". Espero que no te guste la idea de terminar en la cárcel dañando otros sistemas.

Me gustaría comenzar esta breve introducción con una imagen. Esta imagen muestra muchas cosas que más adelante discutiré en detalle. Dice, "El autor está usando la distribución "Ubuntu" de Linux como su sistema operativo por defecto. Ha instalado Virtual Box - una especie de máquina virtual - que también funciona en Windows. Y en esa Caja Virtual ha instalado tres sistemas operativos más. Uno es "Windows XP" y los otros dos son "Kali Linux" y "Windows 7 Ultimate". La imagen también dice, y eso es muy importante, "Actualmente hay tres sistemas operativos que se están ejecutando virtualmente en el escritorio".



**( La caja virtual está ejecutando tres sistemas operativos. Puede probar cualquier tipo de experimento en este sistema operativo virtual. Eso no dañará su sistema principal. )**

Como hacker ético, aprenderás a defenderte. Para defenderte en algún momento necesitas atacar a tu enemigo. Pero es parte de su sistema de defensa. Es parte de su estrategia de defensa. Cuanto más sepas sobre la estrategia de tu enemigo, más podrás defenderte. Usted necesita aprender que esas herramientas son usadas frecuentemente por los hackers o crackers maliciosos. Usan la misma herramienta que tú usas para defenderte.

Si usted es un hacker ético o un cracker malicioso, usted hace lo mismo. Se utilizan las mismas herramientas de software para atacar el sistema de seguridad. Sólo su propósito o intención difiere.

Probablemente usted sabe que una gran empresa de automóviles antes de lanzar un nuevo modelo de coche generalmente prueba el sistema de cierre. Tienen sus propios ingenieros de seguridad y además llaman a los expertos en cerraduras para probar la vulnerabilidad. Ellos pagan una buena cantidad de dinero si usted puede romper el sistema de cierre del coche. Básicamente es un trabajo de "PENTESTING". Los expertos en cierre PENTESTRAN el sistema y ven si hay alguna debilidad en el sistema.

Es un buen ejemplo de hacking ético. Se invita a los expertos en cerraduras a hacer el trabajo y se les paga bien. Por el contrario, los ladrones de coches hacen el mismo trabajo sin invitación. Simplemente rompen el sistema de cierre de un coche desatendido aparcado en la carretera y se lo llevan. Espero que a estas alturas ya tengas

entendió la diferencia entre el pirateo ético y el cracking.

Su intención principal se centra en la seguridad del sistema. La seguridad consta de cuatro componentes clave. A medida que el libro vaya avanzando, irá encontrando palabras como "PENTESTING", "EXPLOIT", "PENETRATION", "BREAK IN THE SYSTEM", "COMPROMISE THE ROUTER", etcétera. El cuatro componentes clave que se mencionan a continuación se refieren principalmente a estos términos. Los componentes clave son:

## **1. Disponibilidad**

## **2. Integridad**

## **3. Autenticidad**

## **4. Confidencialidad**

Veremos cómo los crackers quieren atacar estos componentes para acceder al sistema. Dado que el objetivo principal de un hacker es explotar las vulnerabilidades del sistema, quiere ver si hay alguna debilidad en estos componentes básicos.

Supongamos que el hacker quiere bloquear la disponibilidad de los datos. En ese caso, utilizará el método "Denegación de ataque" o "DoS". Para realizar este ataque, los hackers suelen utilizar los recursos o el ancho de banda del sistema. Pero DoS tiene muchas otras formas. Cuando el recurso o el ancho de banda de su sistema se consume por completo, el servidor suele fallar. El objetivo final es un sistema, pero el número de víctimas es suficiente. Es algo así como millones de personas se reúnen frente a la puerta principal de tu casa y la atascan con una especie de cadena humana para que tú y los miembros de tu familia no puedan entrar en ella.

El segundo componente clave, la integridad, no debe comprometerse a toda costa. ¿Qué significa este término "integridad"? Básicamente se centra en la naturaleza de los datos. Cuando esta naturaleza de datos se altera con algún tipo de ataque de 'BIT- FLIPPING', la integridad del sistema también se ve comprometida. Se puede hacer simplemente cambiando el mensaje. Los datos pueden estar en movimiento o en reposo, pero pueden ser cambiados. Imagínese lo que sucede cuando una transacción de dinero es manipulada con la adición de unos cuantos ceros más al final! Supongamos que un banco está transfiriendo dinero. En su instrucción está escrito: "Transferencia \$10, 000". Ahora el atacante cambia el texto críptico de tal manera que la cantidad cambia a \$10, 000000. Así que el ataque está destinado al propio mensaje o a una serie de mensajes.

El problema de la autenticación es normalmente manejado por el filtro de Control de Acceso a los Medios (MAC). Si está colocado correctamente, la red no permite el uso de dispositivos no autorizados. ¿Qué sucede si alguien falsifica la dirección MAC de una estación de red legítima y la quita? Puede tomar la identidad de la estación y controlarla. Esto se denomina ataque de autenticación o spoofing de direcciones MAC.

Por último, la cuestión de la confidencialidad se plantea por encima de todo. Los datos viajan en texto claro a través de la red de confianza. Aquí los datos significan información. El robo de información como descifrar la contraseña de alguien es un ataque a la confidencialidad. Los datos o la información están destinados a alguien, pero en lugar del destinatario, el hacker obtiene el acceso. En realidad, el cracker lo roba cuando los datos se mueven a través de la red de confianza en forma de texto claro.

---

## Agradecimientos

KARTICK PAUL, SYSTEM MANAGER, AAJKAAL, KOLKATA, sin su ayuda persistente e inspiradora, no podría escribir este libro.

---

# **Contenido**

## **Parte I**

**Capítulo 1: El lado legal de la  
piratería informática Capítulo 2:  
Entorno de piratería informática**

**Hacking y Networking Ético**

**¿Qué significa Network?**

**Resumen**

**Capítulo 3: Instalación de Virtual Box**

**Capítulo 4: Instalación de Kali Linux y otros sistemas operativos en VB**

**Capítulo 5: Terminal Linux, comandos básicos**

**Resumen**

## **Parte II**

**Capítulo 6: Pitón 3 y el pirateo ético**

**Capítulo 7: El entorno de Pitón**

**Capítulo 8: Sintaxis General**

**Cree la función main()**

**Indentación y Espacio en**

**Blanco Comentando**

**Asignación de valores**

**Capítulo 9: Variables, objetos y valores**

**Uso del string**

**de números**

**Qué es Tipo e ID**

**Valores lógicos**

**Tuples y Listas**

**Diccionario**

**Objeto**

**Capítulo 10: Condiciones**

**Capítulo 11: Loops**

**mientras que**

**loops para**

**loops**

**Capítulo 12: Expresiones**

**Regulares Usando el Módulo**

**"re**

**Reutilizar con Expresiones**

**Regulares Búsqueda con**

**Expresiones Regulares**

**Capítulo 13: Excepciones, errores de**

**captura Capítulo 14: Funciones**

**Los valores de**

**retorno generan**

**funciones Listas de**

**argumentos**

**Argumentos**

**nombrados**

## **Capítulo 15: Clases**

**Metodología orientada a objetos**

**La base de la orientación a objetos**

**Comprender las clases y los objetos**

**Escribir tu propio juego, "Bueno contra**

**Malo" Clase primaria y objeto**

**Acceso a datos de objetos**

**Polimorfismo**

**Uso de**

**Generadores**

**Decorador de**

**Herencia**

## **Capítulo 16: Métodos de cadena**

## **Capítulo 17: Entrada y salida de**

## **archivos Capítulo 18: Contenedores**

**Funcionamiento en Tuple y Listar**

**Objeto Funcionamiento en Objeto**

**Diccionario**

## **Capítulo 19: Base de datos**

**Comencemos con**

**SQLite3 MySQL para**

**Big Project**

## **Capítulo 20: Módulo**

## **Capítulo 21: Depuración, Módulo Unittest**

**Capítulo 22: Socket and Networking**

**Capítulo 23: Importando un Módulo**

**Nmap**

**Capítulo 24: Creación de un escáner de red**

**Nmap Parte III**

**Capítulo 25: Proteger el anonimato en Internet**

**Capítulo 26: Dark Web y Tor**

**Wikipedia Oculta**

**Capítulo 27: Cadenas de**

**proxy**

**Capítulo 28: Red privada virtual o VPN**

**Capítulo 29: Dirección MAC**

**Epílogo-¿Qué es lo**

**siguiente?**

---

# **Contenidos de un vistazo**

Sobre el autor

Acerca del revisor técnico

Agradecimientos

Prólogo - Hacker's Goal

## **Parte I**

Capítulo 1: El lado legal de la piratería informática

Capítulo 2: Entorno de piratería informática

Capítulo 3: Instalación de Virtual Box

Capítulo 4: Instalación de Kali Linux y otros sistemas operativos en VB

Capítulo 5: Terminal Linux, Comandos básicos

## **Parte II**

Capítulo 6: Python 3 y el Hacking Ético

**Capítulo 7: Medio Ambiente de Python**

**Capítulo 8: Sintaxis general**

**Capítulo 9: Variables, objetos y valores**

**Capítulo 10: Condiciones**

**Capítulo 11: Bucles**

**Capítulo 12: Expresiones Regulares**

**Capítulo 13: Excepciones, errores de detección**

**Capítulo 14: Funciones**

**Capítulo 15: Clases**

**Capítulo 16: Métodos de encadenamiento**

**Capítulo 17: Entrada y salida de archivos**

**Capítulo 18: Contenedores**

Capítulo 19: Base de datos

Capítulo 20: Módulo

Capítulo 21: Depuración, Módulo Unitest

Capítulo 22: Conexiones y redes

Capítulo 23: Importación del módulo Nmap

Capítulo 24: Creación de un escáner de red Nmap

### Parte III

Capítulo 25: Proteger el anonimato en Internet

Capítulo 26: Dark Web y Tor

Capítulo 27: Cadenas de proxy

Capítulo 28: Red privada virtual o VPN

Capítulo 29: Dirección MAC

Epílogo-¿Qué sigue?

## Indice

---

## Acerca del autor y del revisor técnico

### Sobre el autor

#### **Sanjib Sinha**

escribe historias y códigos, no siempre en el mismo orden.

Comenzó con C# y .NET framework y ganó el premio Microsoft Community Contributor Award en 2011. Más tarde, el movimiento de Software Libre lo atrajo y se convirtió en un entusiasta de Linux, PHP y Python, especializándose y trabajando en White Hat Ethical Hacking.

Como principiante, tuvo que luchar mucho -siempre- para encontrar una forma fácil de aprender a codificar. Nadie le dijo que codificar es como escribir: imaginar una imagen y llevarla a la Tierra con la ayuda de palabras y símbolos.

A lo largo de sus libros ha intentado ayudar a los principiantes desde su perspectiva  
-...como principiante.

### Acerca del revisor técnico

#### **Abir Ranjan Atarthy**

es un Hacker Ético Certificado del Ec-Council, Auditor ISO27001 e implementador de PCIDSS.

Tiene más de 12 años de amplia experiencia en el manejo de los programas de Seguridad de la Información y Cibernética en todos los aspectos clave, es decir, Política, Estándares, Procedimientos, Concienciación, Seguridad de Redes, Seguridad Web, Seguridad de Aplicaciones Android, Respuesta a Incidentes, Análisis de Seguridad, Monitoreo de Seguridad, Protección contra Malware, Configuración de Seguridad, Criptografía, Protección de Datos, Conocimientos de las herramientas más avanzadas de la industria de la seguridad, con el complemento de conocimientos sobre lenguajes de scripts para explotar manualmente las vulnerabilidades.

Es autor de varios artículos técnicos que han sido publicados en revistas de seguridad informática y es frecuentemente invitado a hablar en muchas conferencias de seguridad cibernética y foros de Python.

Ha diseñado cursos de ciberseguridad para Corporaciones en red y web

pruebas de penetración, análisis forense y criptografía.

Abir realiza regularmente talleres de trabajo, sesiones de formación y programas de certificación para empresas, organizaciones gubernamentales, establecimientos de defensa, agencias de seguridad, escuelas de ingeniería y universidades sobre programación Python, pruebas de penetración y ciberforestación forense.

Ha creado varias herramientas de seguridad informática y criptográficas utilizando Python.

Ha realizado programas de corto plazo en programación orientada a objetos y temas seleccionados en ingeniería de software del Instituto Indio de Tecnología -Kharagpur.

Abir es considerado un experto en seguridad cibernética y a menudo es citado por los principales periódicos y canales de televisión.

Actualmente dirige el departamento de inteligencia de amenazas cibernéticas en TCG Digital Solutions Pvt. Ltd.

---

# Parte I

# 1. El lado legal de la piratería informática

Sanjib Sinha 

(1) Howrah, Bengala Occidental, India

## Material suplementario electrónico

La versión en línea de este capítulo (doi: [10.1007/978-1-4842-2541-7\\_1](https://doi.org/10.1007/978-1-4842-2541-7_1)) contiene material suplementario, que está a disposición de los usuarios autorizados.

---

Con el paso del tiempo y el progreso, nuestro viejo entorno también está cambiando muy rápidamente. No ha sido como antes cuando guardamos registros introduciendo los datos en un gran cuaderno de bitácora y los apilamos uno por uno con respecto a una fecha. Ahora guardamos los datos en una computadora. Ya no vamos al mercado a comprar nada. Lo pedimos a través de Internet y el pago se realiza mediante tarjeta de crédito o débito. La naturaleza del delito también ha cambiado en consecuencia.

Los criminales solían robarle sus datos físicamente antes. Ahora se la arrebatan por Internet usando computadoras. Ahora las computadoras se han convertido en una nueva herramienta tanto para los negocios como para los delitos tradicionales. Sobre la base de lo cual, un término -El "ciberderecho" pasa a primer plano. Como hacker ético, lo primero y más básico que debes recordar es "no intentes penetrar o manipular ningún otro sistema sin pedir permiso".

Pueden preguntarse cómo experimentaría con mis conocimientos. La respuesta es Caja Virtual. En su máquina virtual puede instalar tantos sistemas operativos como desee y experimentar con ellos (la imagen de arriba muestra Virtual Box y dos sistemas operativos ejecutándose en ella). Inténtalo todo con ellos. La prueba de cualquier virus en su máquina virtual no afectará a su sistema principal. Al mismo tiempo seguirás aprendiendo sobre malware, virus y todo tipo de posibles ataques.

Algunos ejemplos pueden darle una idea de qué tipo de delitos informáticos son punibles en nuestro sistema legal.

Si usted utiliza cualquier herramienta de software para generar un número de tarjeta de crédito o de débito, entonces es un delito muy punible. Se le impondrá una multa de cincuenta mil dólares y quince años de prisión. Establecer un sitio web falso para tomar números de tarjetas de crédito con una falsa promesa de vender productos inexistentes es una ofensa altamente punible. A esto le sigue un encarcelamiento riguroso y una multa considerable. Puedo darle varios otros ejemplos que pueden causarle problemas si usted no se mantiene dentro de la ley.

Recuerde, usted es un hacker ético y está aprendiendo herramientas de hacking para proteger su sistema o el de su cliente. En aras de la protección y la defensa, es necesario conocer los métodos de ataque, explotación o penetración.

Pruebe cada uno de los experimentos en su máquina virtual. Esa es la regla número uno del hacking ético.

## 2. Entorno de piratería informática

Sanjib Sinha 

(1) Howrah, Bengala Occidental, India

---

Lo primero que necesita es una máquina virtual... Como dije antes, tengo Ubuntu como sistema operativo por defecto y dentro de mi máquina virtual he instalado dos sistemas operativos: uno es Windows XP y el otro es Kali Linux.

Técnicamente, de ahora en adelante mencionaré Windows XP y Kali Linux como mis máquinas virtuales. Kali Linux es una distribución de Linux que ofrece muchas herramientas útiles de hacking. Así que sugiero encarecidamente que lo uses como tu máquina virtual. También puedes leer la página de documentación de Kali Linux, que también será de gran ayuda.

Al mismo tiempo, no sugeriría el uso de Windows de ningún tipo con el propósito de piratear éticamente. Algunos pueden argumentar que pocas herramientas de hacking pueden ser usadas en Windows, así que ¿por qué está sugiriendo lo contrario? El punto es: en el mundo del hacking ético, hay que ser anónimo todo el tiempo. No querrás mantener tu rastro, de todos modos, para que puedas ser rastreado. Permanecer en el anonimato es un gran desafío. En Linux es bastante fácil y puede permanecer anónimo por el momento.

Teniendo eso en cuenta, explico esa técnica de ser anónimo en gran detalle para que antes de saltar a la gran tarea, hagas tu defensa mucho más fuerte. Ser anónimo es lo más importante en el mundo del hacking ético. No es posible mantener el anonimato en Windows. Así que es mejor adaptarse primero al entorno Linux. Otra cosa importante es que la mayoría de las grandes herramientas de hacking no están disponibles en el entorno Windows.

Si nunca has oído hablar de ninguna distribución de Linux, no te preocunes. Puede instalar Ubuntu dentro de su sistema Windows o puede particionar fácilmente su disco en dos partes e instalar Ubuntu y Windows.

por separado como sus dos sistemas operativos predeterminados. Es preferible hacer lo segundo. La instalación y desinstalación de sistemas operativos paralelos siempre le enseña algo nuevo. Si usted está familiarizado con Windows, no le diré que simplemente lo tire por el bien de aprender el hacking ético. Puedes quedártelo y usarlo para tu trabajo diario. No hay ningún problema en hacer esto.

En el mundo de Internet, Linux se usa más. Así que necesitas aprender algunos comandos de Linux. La instalación de software en Linux es ligeramente diferente de los entornos Windows. Hay distribuciones de Linux como Fedora o Debian, y muchas más. Le puse el nombre de Ubuntu sólo porque es muy popular y los usuarios de Windows se sienten cómodos dentro de él. Las operaciones son más o menos las mismas, incluyendo las instalaciones de software. Para los principiantes, no es una buena idea instalar Kali Linux como su sistema operativo predeterminado. Debe leer la documentación de Kali, donde se indica claramente que Kali es más para desarrolladores. Lo vas a instalar dentro de tu Caja Virtual. Kali Linux es un tipo de distribución de Linux que viene con muchas herramientas de hacking. Necesitas conocerlos y usarlos en el curso de un hacking ético.

La instalación de una máquina virtual es un paso muy importante como primer paso en la construcción de su entorno. En el siguiente capítulo le mostraré cómo puede hacerlo para diferentes sistemas operativos. Otra cosa importante es aprender un lenguaje de programación que realmente te ayudará a aprender mejor el hacking ético.

La opción obvia es Python . En el momento de escribir este libro, Python 3.x ya ha llegado y es considerado el futuro de este lenguaje. Se está poniendo al día rápidamente con la antigua versión 2.x de Python, que ha estado en el mercado durante un tiempo. La página oficial de descarga de Python proporciona el repositorio de instaladores de Python para sistemas operativos Windows, Mac OS X y Linux. Si descarga un instalador, es de gran ayuda porque viene con el intérprete de Python, la biblioteca estándar y los módulos estándar. La biblioteca estándar y los módulos incorporados son específicamente muy importantes porque le ofrecen varias capacidades útiles que le ayudarán a lograr su objetivo como hacker ético. Entre los módulos útiles, obtendrá servicios criptográficos, manejo de datos de Internet, interacción con protocolos IP, interoperabilidad con el sistema operativo, y muchos más. Así que adelante, coge cualquier buen libro de principiantes sobre Python, lee la documentación oficial y sabe que es parte de tu horario de aprendizaje. Python es un idioma extremadamente fácil de aprender.

Para crear un ambiente ideal de hacker ético, unos pocos pasos son extremadamente importantes. Los pasos incluyen: instalar una Máquina Virtual o Caja Virtual (VB), tener un conocimiento básico sobre redes, y aprender un lenguaje de programación útil como Python. Veamos primero el

conocimiento básico de las redes.

---

## Hacking ético y trabajo en red

Un conocimiento básico sobre el trabajo en red es extremadamente importante si quieras aprender el hacking ético. A medida que progresas y deseas profundizar, es aconsejable aprender más sobre el trabajo en red. El hacking ético y el trabajo en red están estrechamente relacionados. A medida que avanza en este libro encontrará palabras como "paquete", "switch", "router", "módem", "TCP/IP", "OSI" y muchas más.

Lo primero que necesita saber es: los datos viajan a través de muchas capas. Los hackers éticos tratan de entender estas capas. Una vez que han entendido el movimiento, quieren rastrear y bloquear los datos o recuperarlos.

En este capítulo veremos muy brevemente cómo funcionan los modelos de redes. Examinaremos los diferentes tipos de modelos de redes. También aprenderemos sobre los dispositivos que componen una red.

---

## ¿Qué significa red?

Una red es un conjunto de dispositivos que se conectan a través de los medios de comunicación. Una de las principales características de una red es que los dispositivos contienen servicios y recursos. Los dispositivos contienen ordenadores personales, conmutadores, routers y servidores, entre otros. ¿Qué es lo que hacen básicamente? Envían datos y los obtienen ya sea por conmutación o por enrutamiento. En realidad, conectan a los usuarios para que éstos obtengan datos completos en lugar de obtenerlos por partes. Por lo tanto, los servicios básicos que estos dispositivos proporcionan incluyen conmutación, enrutamiento, direccionamiento y acceso a datos.

Podemos concluir que una red conecta principalmente a los usuarios para aprovechar estos servicios. Ese es su primer trabajo. El segundo trabajo también es muy importante. Una red siempre mantiene un sistema para que los dispositivos permitan a los usuarios compartir los recursos de forma más eficiente.

Ahora surge un problema, no un problema trivial. Los fabricantes de hardware y software no se conocen entre sí. Pertenecen a diferentes países y comparten diversas culturas. Cuando la concepción de las redes pasó a primer plano, se descubrió que el hardware y el software no coincidían. Como dije antes, una red es una colección de dispositivos. Estos dispositivos están construidos principalmente de hardware y software que hablan en diferentes idiomas.

Para resolver este problema, se necesita un modelo de red común con funciones de comunicación para que dispositivos diferentes puedan interoperar.

La importancia de los modelos de trabajo en red consiste en algunos

conceptos principales. En primer lugar, fomentan la interoperabilidad. En segundo lugar, proporcionan una

referencia a través de la cual se comunicarán los datos. En tercer lugar, facilitan la ingeniería modular.

Existen dos tipos de modelos de redes.

Se trata del modelo de referencia de interconexión de sistemas abiertos (OSI) y del modelo de protocolo de control de transmisión/protocolo de Internet (TCP/IP). Ambos modelos son ampliamente utilizados hoy en día.

El modelo de referencia de Interconexión de Sistemas Abiertos (OSI) fue desarrollado por la Organización de Estándares de Internet (ISO) y tiene siete capas en total. Las capas son las siguientes: aplicación (capa 7), presentación (capa 6), sesión (capa 5), transporte (capa 4), red (capa 3), enlace de datos (capa 2) y físico (capa 1).

Tratemos muy brevemente de comprender cómo funciona este modelo. Suponga que un usuario intenta abrir una página web. Lo primero que hace es enviar una petición al servidor que se encuentra a varios miles de kilómetros de distancia. Aquí, el disco duro o hardware del servidor es la última capa (capa 1) que se denomina "física". Así, la petición del usuario golpea primero la capa de "aplicación" (7) que es la más cercana y luego procede. Cada proceso en cada capa implica un funcionamiento complicado de "bits y bytes". Un ordenador sólo entiende 0 y 1. Pero al usuario no le gusta ver un vídeo en 0 y 1.

Vamos a dividir el proceso en más detalles.

En la capa de aplicación (7), el usuario interactúa con el dispositivo, que puede ser un ordenador personal o un teléfono inteligente o cualquier cosa que se pueda adivinar. Así que la capa de aplicación básicamente maneja la interacción del usuario. El nombre del datagrama es "data". El usuario solicita los datos y finalmente los recupera. ¿Qué sucede cuando el usuario envía peticiones desde la capa 7? Entra en la siguiente capa: (6) presentación . Se inicia el proceso de encapsulación. Los datos están formateados y encriptados. A continuación, la capa 5 o sesión entra en la escena. Esta capa gestiona la comunicación de extremo a extremo. Suponga que escribe una contraseña e inicia sesión en su cuenta de medios sociales. Esta capa mantiene la comunicación de extremo a extremo (usuario a servidor) para que usted pueda permanecer conectado a su página. Dígale a esta capa que el nombre del datagrama es "datos".

Para ayudarle a mantener su sesión, las siguientes tres capas trabajan muy duro. Son: transporte (capa 4), red (capa 3), enlace de datos (capa 2), respectivamente. El nombre del datagrama de la capa de transporte es "segmento". ¿Por qué se llama "segmento"? Se llama "segmento" porque divide su petición en varias fracciones. En primer lugar, añade los números de puerto de origen y de destino. A continuación, intenta que sea fiable, añadiendo números de secuencia. Por lo tanto, en pocas palabras, proporciona control de flujo, secuenciación y confiabilidad.

¿Qué pasa después?

Su petición entra en la capa 3 que se llama red . El nombre del datagrama es "paquete". Añade direcciones IP de origen y destino. También se asegura de que su solicitud encuentre el mejor camino para llegar al destino.

Ahora su solicitud de datos casi llega a la fase final. Entra en la capa 2 que es el enlace de datos. Se está acercando al punto final que es el hardware del servidor. Por lo tanto, esta capa añade direcciones de control de acceso a medios (MAC) de origen y de destino. A continuación, pasa por los procesos del sistema Frame Check System (FCS). Comprueba cuadro por cuadro si las solicitudes de origen llegan al destino correcto. Por eso el datagrama se conoce como "frame".

Ahora ha entrado en el destino final que es la capa 1 o física. Sólo hay fragmentos en el medio físico. El nombre del datagrama es "bits y bytes".

Ahora podemos imaginarnos una pequeña oficina con un router, dos conmutadores y unos cuantos ordenadores de sobremesa, portátiles, impresoras y servidores. El enrutador está conectado a los conmutadores y los conmutadores están conectados a los dispositivos como ordenadores de sobremesa, portátiles, impresoras y servidores. Aquí, los ordenadores de sobremesa, portátiles, impresoras y servidores pertenecen a la capa 1 que es física. Los conmutadores pertenecen a la capa 2, que es el enlace de datos, y el enrutador encaja en la capa 3, que es la red.

Los enrutadores son dispositivos de capa 3 y realizan algunas tareas definidas. Son: conmutación de paquetes, filtrado de paquetes, selección de ruta y, finalmente, comunicación. La tarea de conmutación de paquetes implica el proceso de llevar un paquete al siguiente dispositivo. Aquí, el siguiente dispositivo son los interruptores. El filtrado de paquetes sugiere en su nombre lo que realmente hace. Permite o bloquea los paquetes en función de determinados criterios. La selección del camino es determinar el mejor camino a través de la red hasta el destino. La comunicación es otra parte importante de esta capa. Los enrutadores se comunican con otras redes como Internet.

Entre los enrutadores, los dispositivos de capa 3 y los dispositivos de aplicación final, físicos, de capa 1, hay conmutadores que son dispositivos de capa 2. En algunos casos, los conmutadores realizan la tarea de los dispositivos de capa 3. Los interruptores se ocupan básicamente del filtrado y reenvío de tramas. También mantiene la conexión entre la capa 3 y la capa 1.

---

## Resumen

Recapitulemos rápidamente lo que acabamos de aprender sobre las relaciones entre el pirateo ético y el trabajo en red.

1. Los modelos de trabajo en red fomentan la interoperabilidad entre

diferentes

que proporcionan una referencia para describir la comunicación de datos. Al mismo tiempo, facilita la ingeniería modular.

2. Existen dos tipos de modelos de redes. Son el Modelo de Referencia OSI y el Modelo TCP/IP.
3. El modelo OSI tiene siete capas. Son: aplicación (capa 7), presentación (capa 6), sesión (capa 5), transporte (capa 4), red (capa 3), enlace de datos (capa 2) y físico (capa 1).
4. El modelo TCP/IP tiene cuatro capas. Son: aplicación (capa 4), transporte (capa 3), red (capa 2) y red (capa 1).
5. Un hacker ético intenta comprender este proceso de comunicación de datos y penetra según la vulnerabilidad.

### 3. Instalación de Virtual Box

Sanjib Sinha 

(1) Howrah, Bengala Occidental, India

---

La primera pregunta que nos viene a la mente es: ¿por qué necesitamos una caja virtual cuando tenemos un sistema operativo por defecto? Hay varias razones. La razón más importante es: en una caja virtual podemos jugar con cualquier sistema operativo sin miedo a estropearlo, incluso romperlo. Existe la posibilidad de que mientras probamos una herramienta de hacking podamos romper un sistema. Le animo a que lo haga. Es una máquina virtual. Así que, adelante. Pruebe todo lo que le venga a la mente. Otra gran razón para usar la caja virtual es la seguridad. Cuando usted visita un sitio web, puede considerar que es seguro, pero en realidad no puede serlo. Pero nada importa en el caso de una caja virtual. No es su máquina original con datos confidenciales. Visitar un sitio web inseguro ya no es molesto.

Sólo tienes que recordar una cosa. Manténgase dentro de la ley. Al probar sus herramientas de hacking o códigos de ejecución, no puede poner en peligro ningún otro sistema.

El sitio web oficial de Oracle Virtual Box ofrece muchas opciones de descarga.

Puedes elegir cualquiera de ellos. Según su sistema operativo, vaya a la sección "download" y vea lo que está disponible para usted. A partir de la siguiente imagen tendrá una idea de cómo puede seguir adelante.

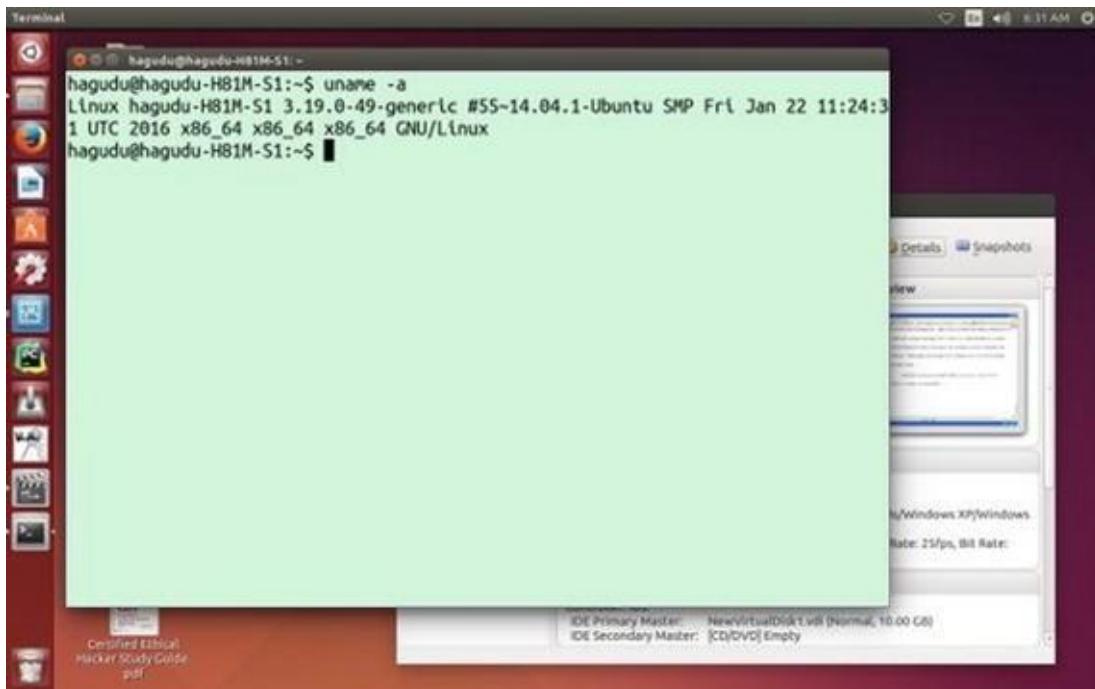


Figura 3-1. Sección de descarga de Virtual Box para hosts Linux

La línea seleccionada de la imagen de arriba muestra el sistema operativo por defecto que estoy ejecutando actualmente. Esto es Ubuntu 14.04 (Trusty) y la arquitectura es AMD64.

Virtual Box es muy fácil de instalar. Cualquiera que sea su sistema operativo (Mac OS X, Windows o Linux), puede instalarlo. En primer lugar, usted necesita saber acerca de su propio sistema operativo. Puede ser de 32 o 64 bits. En cualquier distribución de Linux, es extremadamente fácil de aprender. Sólo tienes que abrir el terminal y escribir: "uname -a".

El terminal proporcionará información vital que incluye todos los datos relativos a mi sistema actual por defecto. El Linux es de la versión 3.19.0 y el nombre del superusuario es "hagudu". También indica qué tipo de arquitectura de sistema es ésta. Se parece a esto:



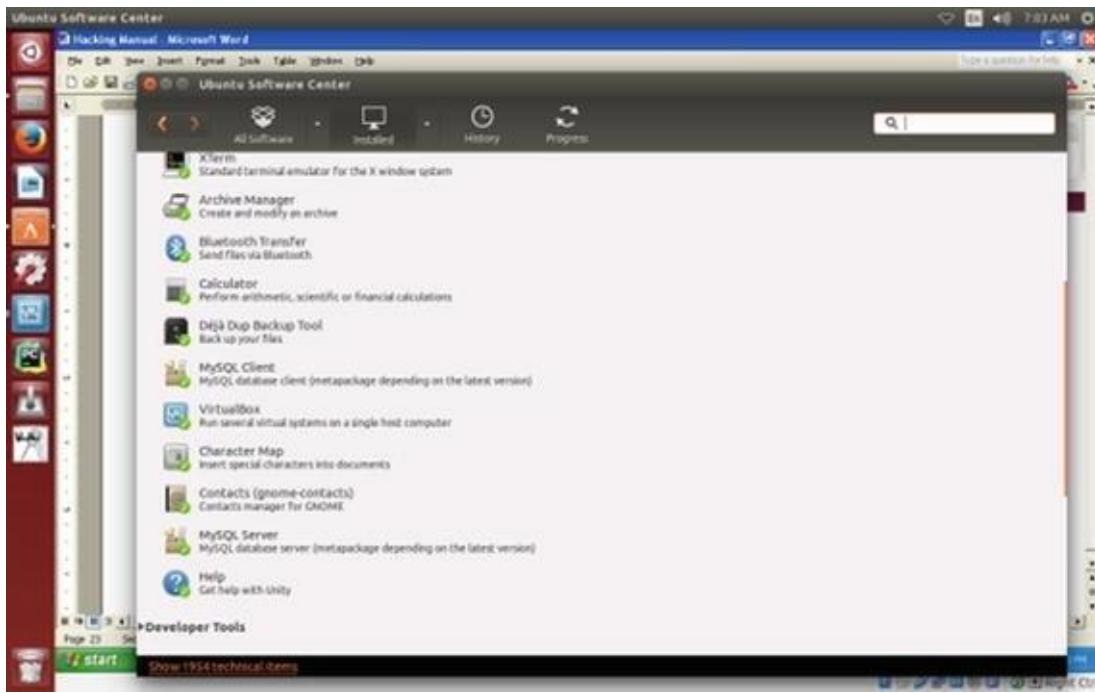
**Figura 3-2.** Una imagen del terminal que habla sobre la arquitectura del sistema

Como en mi caso, se ve claramente que "x86\_64" significa 64 bits. En la página oficial de descarga de Virtual Box para todas las distribuciones de Linux, primero descargue los paquetes requeridos y luego instálelos de acuerdo a la naturaleza de su sistema operativo. Para Red Hat, Fedora o cualquier distribución de Linux que pertenezca a esa categoría, notará que la última extensión es ".rpm". En ese caso, puede moverse a la carpeta Virtual Box y emitir comandos como "rpm -i" o "yum install" en caso de que ejecute Red Hat o Fedora .

Pero hay métodos más sencillos para instalar Virtual Box.

Para los principiantes absolutos es muy útil ejecutar la distribución Ubuntu Linux como su sistema operativo predeterminado. Puede instalar Virtual Box directamente desde el centro de software sin tener que abrir el terminal ni dar ninguna orden.

El centro de software de Ubuntu tiene muchas categorías. Una de ellas muestra el software "instalado".



**Figura 3-3.** El centro de software de Ubuntu muestra Virtual Box funcionando

No está allí por defecto. En ese caso, es extremadamente fácil de instalar. Sólo tiene que escribir "Virtual Box" en el cuadro de texto de búsqueda y aparecerá. Avance y presione el botón de instalación.

## 4. Instalación de Kali Linux y otros sistemas operativos en VB

Sanjib Sinha 

(1) Howrah, Bengala Occidental, India

---

Una vez que la Caja Virtual ha sido instalada en su máquina, no tiene que preocuparse de instalar varios sistemas operativos en ella. Desde el principio, estamos interesados en instalar Kali Linux en nuestra Caja Virtual. Vaya al sitio web oficial de Kali Linux y descargue la imagen ISO de la última versión estable. Kali Linux es una distribución de Linux mucho más grande que otras distribuciones de Linux. Debe tener unos 3 GB. Ubuntu y otros están alrededor de 1 GB o un poco más.

Ahora, una vez finalizado el proceso de instalación, puede almacenarlo en su disco duro local o grabarlo en un DVD. Ahora abra su Caja Virtual y haga clic en "Nuevo". Se abrirá automáticamente una nueva ventana que le preguntará qué tipo de sistema operativo va a instalar. La siguiente imagen es bastante auto-explicativa.



**Figura 4-1.** Cómo instalar un sistema operativo en una máquina virtual

Usted ve en la Caja Virtual que ya he instalado dos sistemas operativos. Uno es Kali Linux y el otro es Windows XP . En su caso, cuando vaya a instalar fresh, el panel izquierdo de su caja virtual estará vacío.

Todo el procedimiento es muy explícito en sí mismo. Le guiará a hacer lo que tiene que hacer a continuación. Básicamente, en Internet hay muchas guías ilustrativas que te ayudarán a hacer lo mismo. Ahora es el momento de escribir el nombre del sistema operativo que va a instalar. A continuación, seleccione el tipo (si es Linux o Windows, etc.) y la versión. En la larga lista de versiones de la sección no encontrarás el nombre de Kali. Pero básicamente es "Debian..." Así que siga adelante y seleccione Debian de 32 o 64 bits de acuerdo con la arquitectura de su sistema. Haga clic en "Siguiente" y le pedirá el uso de la memoria tal y como se muestra en la siguiente imagen.



Figura 4-2. El proceso de instalación de Kali Linux en Virtual Box solicita el tamaño de la memoria

Puede asignar el tamaño de la memoria según la capacidad de su máquina. Un mínimo de 1 GB es suficiente. Es mejor si puedes asignar más. En el próximo paso se pedirá capacidad de almacenamiento y algunas otras cosas muy importantes.

Puedo asegurarle que, como principiante, no tendrá ninguna dificultad para instalar Kali Linux en su Caja Virtual. La parte más importante de este proceso de instalación es que necesita mantener su conexión a Internet en funcionamiento para que Kali Linux pueda ajustar sus requisitos en línea.

Por lo general, cuando se instala un sistema operativo en una máquina virtual, aparece en un tamaño pequeño y se mantiene así. La siguiente imagen le mostrará el tamaño original.

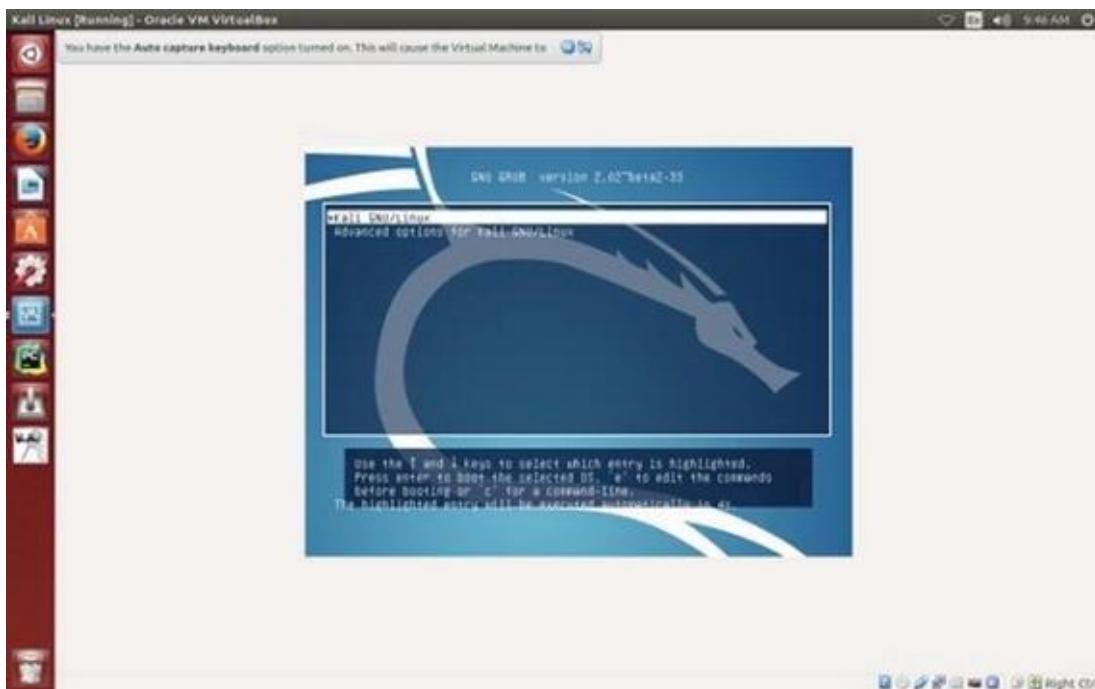


Figura 4-3. Kali Linux corriendo en Oracle VM Virtual Box

Pero trabajar en este tamaño es realmente engorroso. Para resolver este problema, normalmente se utiliza la adición de invitados a la caja virtual. Pero antes de eso, es posible que desee actualizar y actualizar su recién instalado Kali Linux. Esa es una buena práctica que te ayuda a estar actualizado todo el tiempo. Después de haber ingresado escribiendo el nombre de usuario y la contraseña, encontrará el terminal en el panel de la izquierda. Ábrelo y escribe:

```
actualización de apt-get
```

Debe estar en línea para que se actualice por sí solo. Podría llevar algún tiempo. Después de que termine, usted emite el segundo comando:

```
apt-get upgrade
```

Normalmente, la actualización lleva más tiempo que la actualización. Si usted es un usuario root entonces no debería haber ningún problema. Pero si has creado otro usuario y te registras como tal, entonces debes escribir "su" antes. "su" significa superusuario o usuario root que es el administrador. Le pedirá su contraseña de superusuario al instante. Dámelo y funcionará bien.

Volvamos a un viejo problema. La nueva instalación de Kali Linux parece pequeña en tamaño y obviamente estás perdido y no sabes qué hacer. ¿Cómo obtendrá la vista de pantalla completa?

Aquí hay un comando que te rescatará de este problema y lo resolverá. Necesita instalar un paquete más y actualizar su máquina virtual de nuevo para que obtenga la vista de pantalla completa.



**Figura 4-4.** Kali Linux ejecutando Oracle VM Virtual Box con herramienta de ataques de contraseña

Abra el terminal y escriba :

```
apt-get update && apt-get install - dkms linux-headers - $(uname -r)
```

Esto instalará el paquete necesario que ejecutará el Virtual Box Guest Addition. Es algo que puede imaginar como una herramienta que controla el tamaño de la pantalla de su sistema operativo.

¿Cómo lo ejecutará una vez instalado el paquete? La siguiente imagen te guiará para encontrar el lugar donde lo conseguirás.

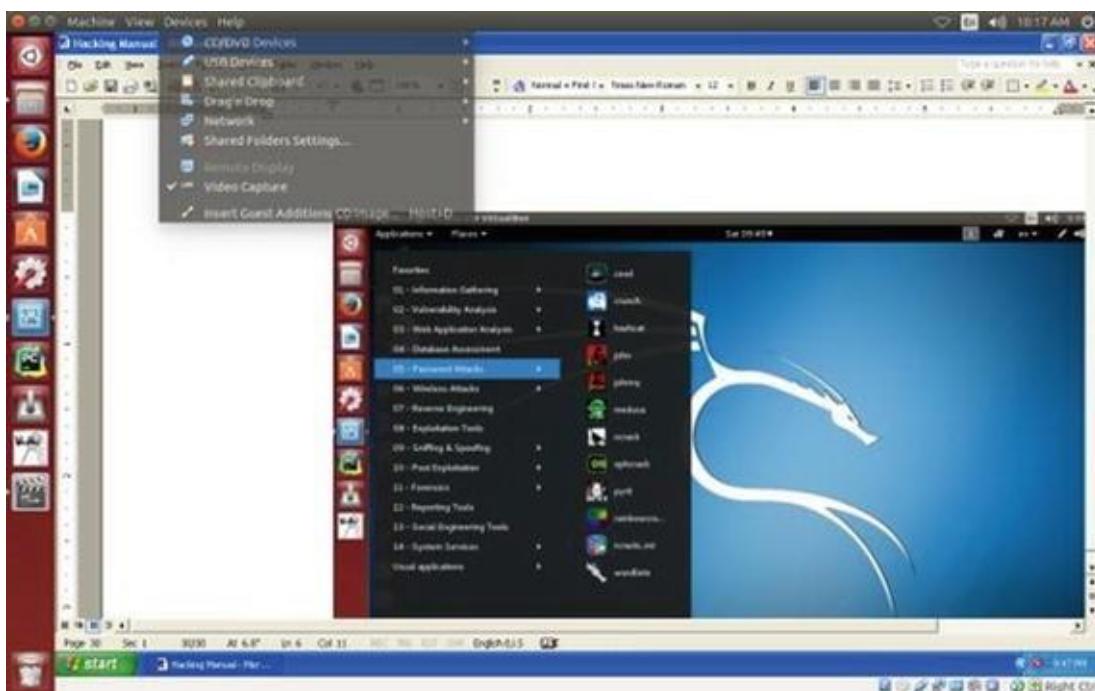


Figura 4-5. Obtener el tamaño de pantalla completa de Kali Linux en Virtual Box

Lleve el puntero del ratón a la parte superior central donde encontrará el menú "Dispositivos". El último dice así: "Insertar imagen de CD de la edición para invitados". Haga clic en él y automáticamente se encargará de todo.

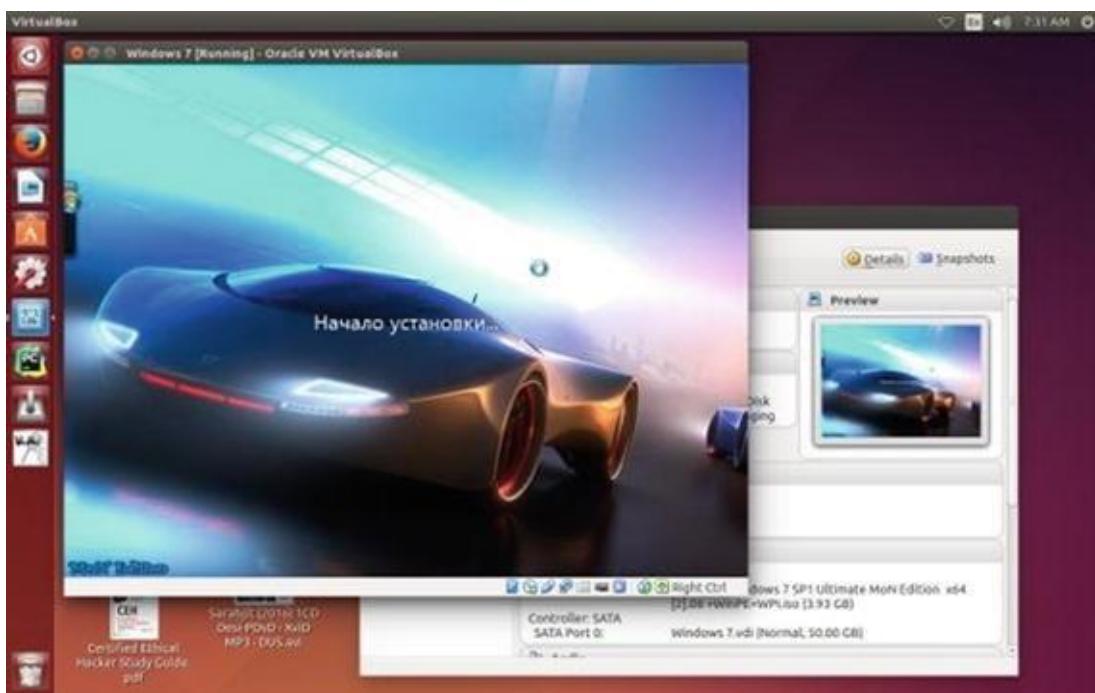
Normalmente debería funcionar bien. Si no, tómelo como un desafío. Busca en Internet. Hay muchas manos que te ayudan esperando para que ayudes a conseguir lo que quieras.

Ahora vamos a instalar Windows 7 Ultimate . El proceso de arranque es el mismo. Abre la caja virtual. Vaya a "nuevo" y haga clic. Se abrirá una ventana que le pedirá que escriba el nombre del sistema operativo que va a instalar. A continuación le pedirá el tamaño de la memoria. Para Windows 7 Ultimate es necesario asignar al menos 2 GB. Cuanto más grande, mejor. Para la capacidad de almacenamiento del disco duro, 50 GB son suficientes.

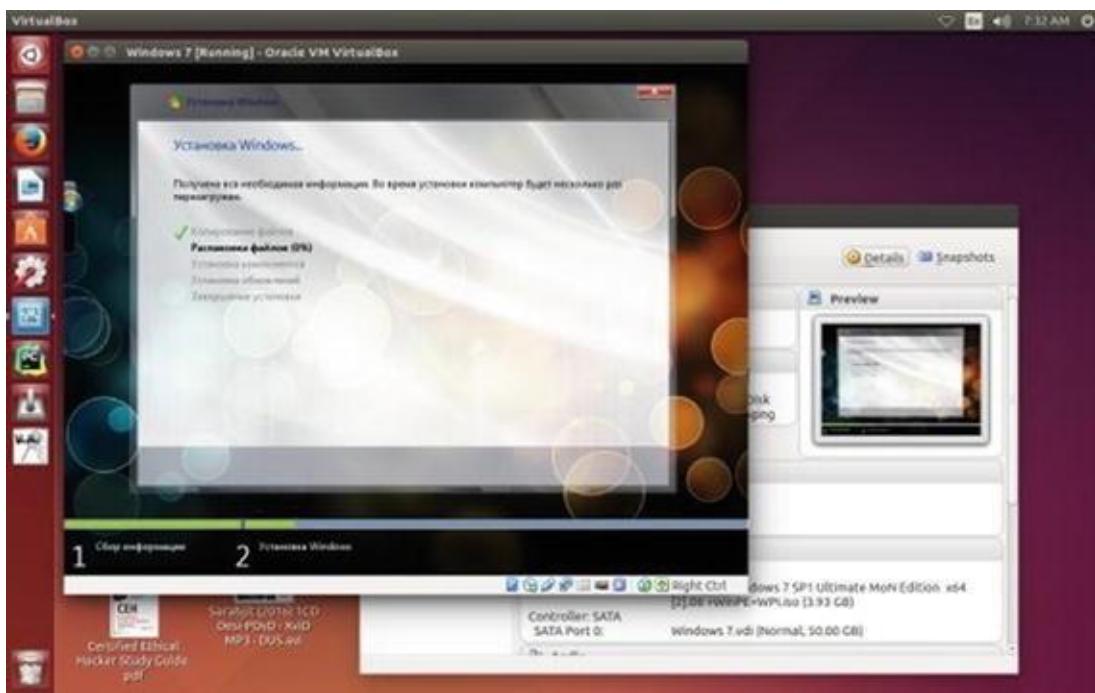
Ahora está listo para conectarse a la imagen ISO del sistema operativo.

Esta parte es un poco complicada, pero cualquier guía en línea le mostrará cómo puede conectarlos.

Cuando haga clic en la sección "almacenamiento" de su Caja Virtual, se abrirá una ventana que le indicará que se conecte con la imagen ISO. No es nada difícil. La ventaja de Virtual Box es que si no hace algún trabajo no afectará a su máquina original.



**Figura 4-6.** La instalación de Windows 7 Ultimate tiene lugar



**Figura 4-7.** Windows 7 Ultimate se está instalando

Cuando se instala un nuevo sistema operativo en su máquina virtual, suele ser de tamaño pequeño. Pero hay una técnica que te ayudará a conseguir la pantalla completa original

efecto.

Para Windows 7 Ultimate, hay una carpeta Virtual Box Guest Guest Addition disponible en la sección de almacenamiento. La caja de color azul viene con una etiqueta. Lee las adiciones de Virtual Box Guest. Sólo tienes que hacer clic en él. Se abrirá. Contendrá varios archivos. Notará dos archivos ".exe". Uno es para el 32-bit y el otro es para la arquitectura de sistema de 64-bit. Mi máquina es de 64 bits, así que hago clic y la ejecuto. Los pasos son muy sencillos. Se le pedirá que lo instale. Haga clic en Aceptar y continúe. Esto hará que su máquina virtual Windows 7 Ultimate esté en pantalla completa.

Hemos instalado con éxito Virtual Box en nuestra máquina virtual y hemos instalado Kali Linux y Windows 7 Ultimate en ella. Ahora es el momento de seguir adelante.

## 5. Terminal Linux, Comandos Básicos

Sanjib Sinha 

(1) Howrah, Bengala Occidental, India

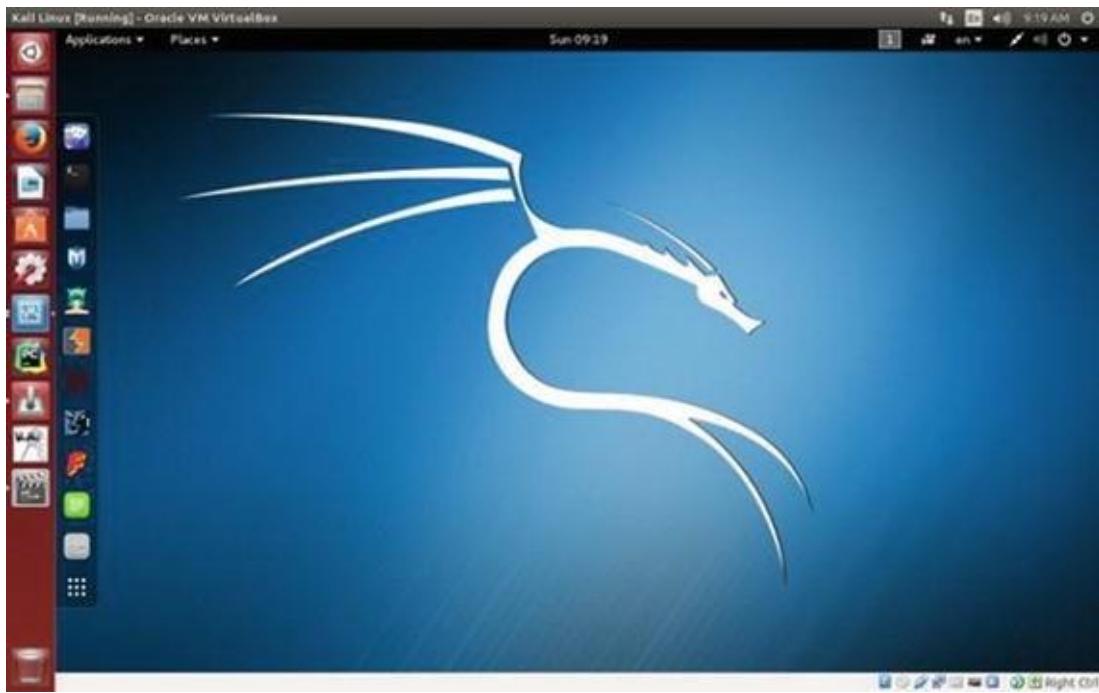
---

Es muy importante conocer el terminal y los comandos de Linux. No en gran detalle, pero este conocimiento primario le ayudará inmensamente en el futuro. Cuanto más se adentre en el mundo del hacking ético, más sentirá que necesita saber más sobre el sistema Linux. Este libro no te llevará tan lejos. Pero un conocimiento muy básico es necesario para que puedas entender lo que está sucediendo a tu alrededor.

Puede parecer repetitivo, pero me gustaría que se cimentara en tu mente que sin conocer bien Linux no puedes adentrarte en el misterioso mundo del hacking ético. Así que primero debes conocer los comandos básicos. Estos comandos le dirán acerca de la computadora en sí. Le dirá la ubicación del sistema de archivos

-donde estás en tu ordenador. Con estos comandos puede cambiar el permiso de un sistema de archivos, copiar o eliminar permanentemente un archivo. Puede añadir un nuevo usuario a su sistema. Puede tener una lista de los archivos que se encuentran actualmente en el directorio en el que se encuentra. Este listado incluye los archivos ocultos. En pocas palabras, al menos puede realizar las operaciones básicas a través del teclado sin utilizar el puntero del ratón. Eso es genial desde la perspectiva de un principiante, supongo.

Para empezar, comenzemos primero con Kali Linux. En la siguiente imagen verás una representación a pantalla completa de Kali. Voy a explicar algunas cosas primero, para que como principiante aprendas lo que necesitas saber primero sobre Kali.



**Figura 5-1.** Vista de pantalla completa de Kali Linux con su panel izquierdo

La imagen de arriba muestra la vista de pantalla completa de Kali Linux. En el panel de la izquierda, en la parte superior, se encuentra el navegador "Iceweasel .". A continuación, sigue la herramienta de línea de comandos. Necesitamos esa herramienta muy a menudo en las próximas lecciones. La herramienta o terminal de línea de comandos trata básicamente con todo tipo de entradas de teclado. Los buenos programadores apenas utilizan un puntero del ratón. Se sienten más cómodos con este terminal y con la manipulación. El sistema de archivos lo sigue. Si hace clic en él, se abrirá una ventana como cualquier versión de Windows NT. Verá varios directorios y carpetas como "Home", "Downloads", "Pictures", etcétera.

Comencemos con la herramienta de comandos abriéndola. Puedes hacer que parezca más grande. Sólo tiene que utilizar las teclas "control" y "shift" con el signo "+".

En la siguiente imagen verás unos cuantos comandos de inicio que normalmente escribimos para saber qué tipo de archivos tenemos en algunos directorios o carpetas.

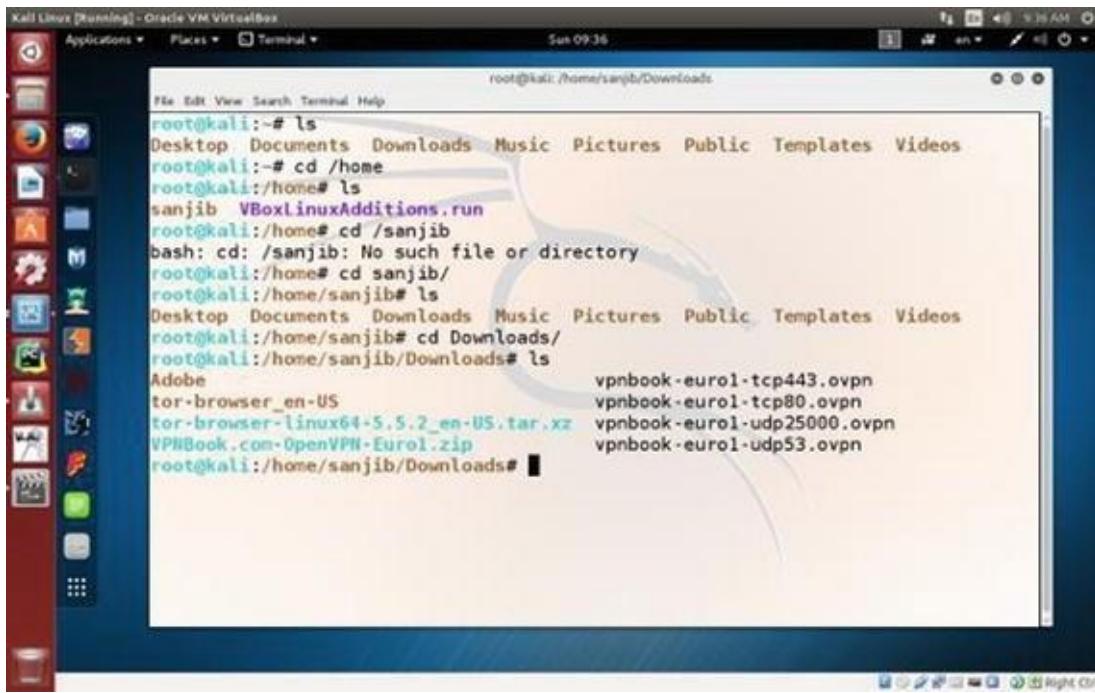


Figura 5-2. Kali Linux con la herramienta de línea de comandos

¿Qué muestra la imagen?

Esto demuestra que he escrito "ls" primero. ¿Qué significa ese comando "ls"? Significa listado. Le digo a Kali que muestre la lista de archivos y carpetas que tienes y, en una fracción de segundo, me muestra todo lo que tiene.

A continuación he utilizado el comando "cd". ¿Qué significa eso?

Este comando "cd" significa "cambiar de directorio". En la imagen se ve que he cambiado el directorio a "home" y emito el comando "ls" de nuevo para ver lo que tiene. Tiene una carpeta llamada "sanjib" y un archivo. La carpeta "sanjib" significa que la "raíz" o el propio sistema tiene un usuario llamado "sanjib". Ahora como root o administrador, he creado ese usuario para que al principio pueda entrar como "sanjib". Puede crear varios usuarios en un sistema Linux para que desde varias máquinas puedan acceder a sus archivos y carpetas. Pero los usuarios nunca tendrán el privilegio de root. No pueden penetrar en el espacio del administrador, pero el root o administrador siempre puede ver lo que están haciendo los usuarios. Como root, un administrador puede crear o eliminar cualquier usuario.

Desde este lugar se puede adivinar lo que está pasando. Cambiamos el directorio y miramos lo que "sanjib" tiene en su directorio "Descargas".

A continuación nos enteramos del comando "pwd", que indica tu posición. Como root, si está en el directorio "Home" y emite un comando "pwd", tiene una salida como esta:

```
root@kali:/home# pwd  
/home  
root@kali:/home#
```

Dice que está en el directorio "/home". Este comando "pwd" es importante cuando hay que controlar un sistema grande y complicado. A menudo se le puede olvidar dónde está trabajando. Por lo general, si desea volver al directorio anterior, debe escribir esto:

```
root@kali:# cd  
/home/sanjib/  
root@kali:/home/sanjib# cd  
.... root@kali:/home#
```

Significa que primero vas al directorio "sanjib" y luego vuelves con un comando "cd" con dos puntos.

A continuación aprenderemos sobre el comando "cp" . Este comando significa copia.

Puede copiar un archivo de un destino a otro. Hemos visto que en nuestro directorio "home" tenemos un archivo, "VBoxLinuxAdditions.run..." Copiemos este archivo al directorio "Documents" del usuario "sanjib".

```
root@kali:/home# cp -v VBoxLinuxAdditions.run  
/home/sanjib/Documents/  
'VBoxLinuxAdditions.run' -> VBoxLinuxAdditions.run  
DIFUNDE LA PALABRA-
```

Ahora nos gustaría ir a la carpeta de documentos "sanjib" y ver si el archivo ha sido copiado correctamente o no.

```
root@kali:/home# cd sanjib/Documents/  
root@kali:/home/sanjib/Documents# ls  
VBoxLinuxAdditions.run  
root@kali:/home/sanjib/Documents#
```

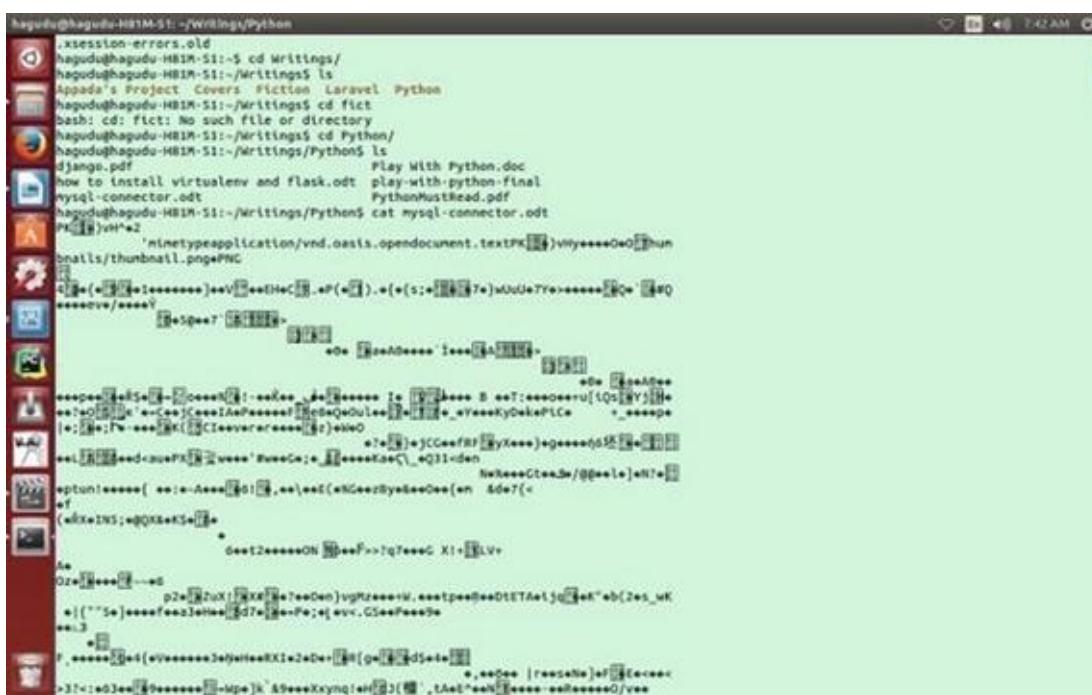
He cambiado el directorio a "sanjib/Documents" y emito el comando "ls" para ver la lista. Muestra el archivo. Así que está funcionando correctamente.

Puedes aprender sobre cualquier comando muy fácilmente. Sólo tiene que añadir un comando "- help" como este: "cp -help ." Escupe todo sobre ese comando y es muy verboso. Le informa sobre cualquier comando con todo detalle.

Otro comando muy importante es "mv". Con este comando, puede

mover cualquier archivo de una carpeta a otra. Este comando es más o menos como un comando "cp". Pero hay una gran diferencia. Este comando mueve completamente el archivo de un lugar a otro. Otro comando importante es "cat..." Puede leer cualquier archivo de texto con la ayuda de este comando.

Tengo una carpeta llamada "Escritura" y tengo algunos documentos allí. Ahora con la ayuda de este comando podemos leer cualquier archivo de texto. Recuerde que sólo es cierto para un archivo de texto. Para un experimento, quería leer un archivo con la extensión ".odt" y la siguiente imagen muestra cómo se veía en el terminal.



```
hagudu@hagudu-HBIM-S1:~/Writings$ cat mysql-connector.odt
bash: cat: No such file or directory
```

Figura 5-3. Intentar leer un archivo sin texto con el comando "cat"

En esta parte quiero mostrar otro truco que a menudo se usa en Linux. Suponga que desea escribir un archivo de texto muy rápidamente. Puedes usar "nano". Viene con todas las distribuciones de Linux. Simplemente escriba "nano" en su terminal y se abrirá un editor de texto en el propio terminal. La siguiente imagen muestra cómo sucede.

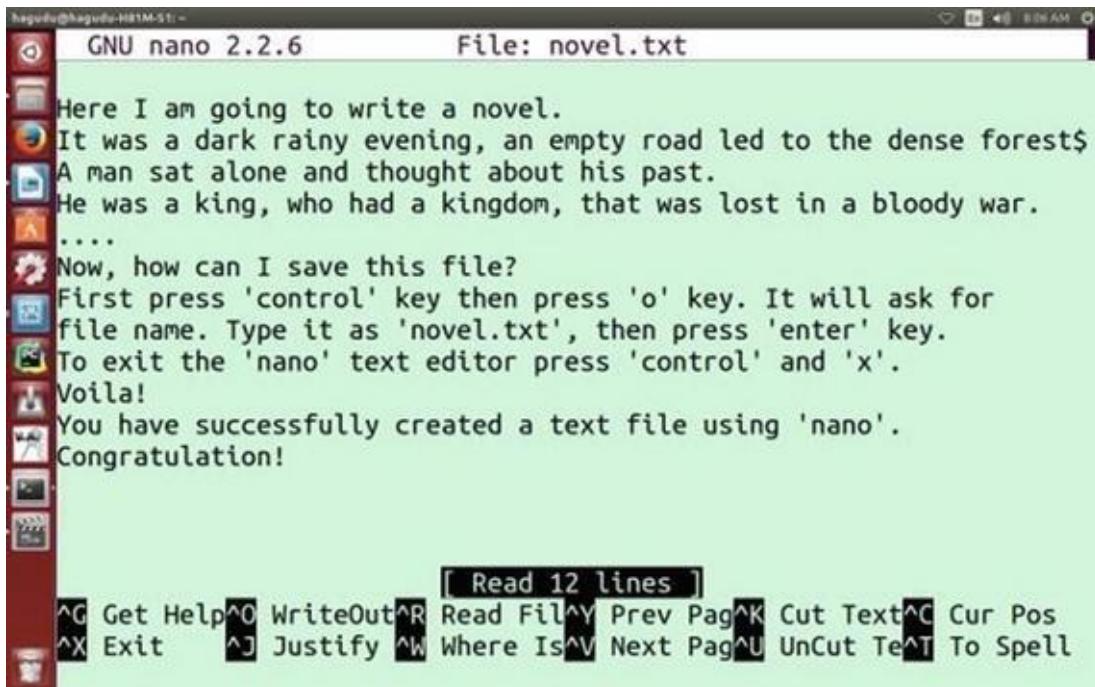


Figura 5-4. Nano editor de texto. Cómo guardar un archivo y salir del editor está escrito en él.

Ahora puede leer con seguridad este nuevo archivo, "novel.txt", con su comando "cat". Todo lo que tiene que hacer es emitir un comando en su terminal de esta manera:

```
gato novel.txt
```

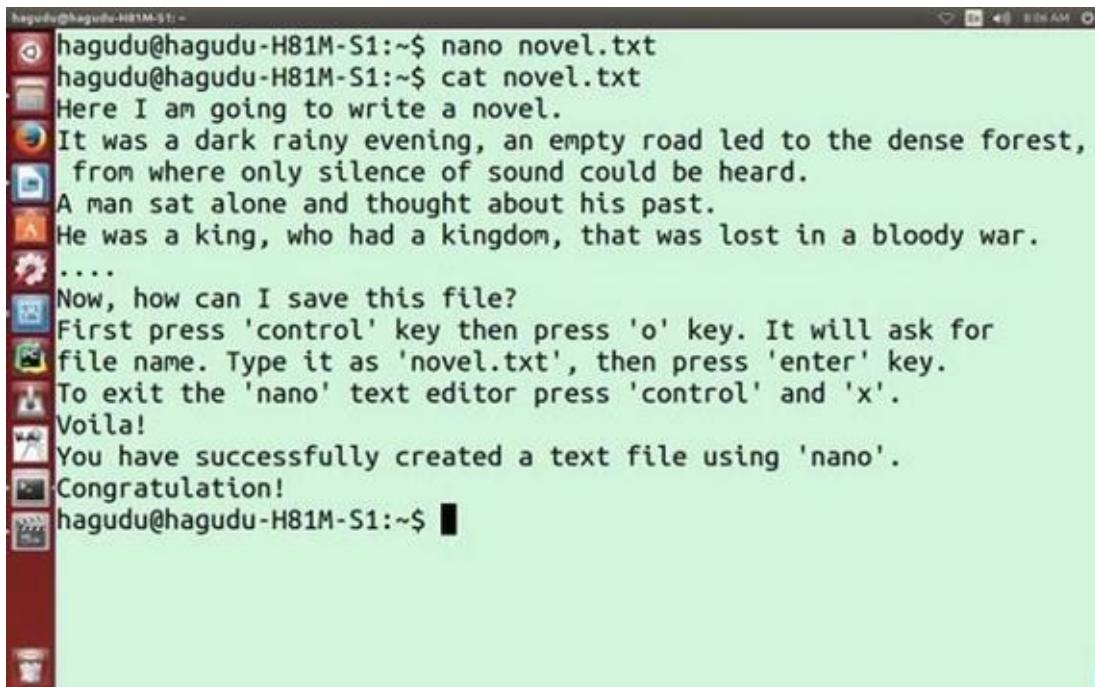
Leerá su archivo en el propio terminal.

Ahora puede ser una buena idea editar este archivo. Puede editarlo en el terminal usando "nano". En ese caso, debe escribir en su terminal este comando:

```
nano novela.txt
```

Esto le dirá a "nano" que abra el archivo. El resto es el mismo. Puede editar cualquier parte y, con las teclas "control" y "o", puede volver a guardarla. A continuación, puede salir del archivo con "control" y "x".

En la siguiente imagen veremos cómo se ve cuando intentamos leer un archivo usando el comando "cat".



```
hagudu@hagudu-H81M-S1:~$ nano novel.txt
hagudu@hagudu-H81M-S1:~$ cat novel.txt
Here I am going to write a novel.
It was a dark rainy evening, an empty road led to the dense forest,
from where only silence of sound could be heard.
A man sat alone and thought about his past.
He was a king, who had a kingdom, that was lost in a bloody war.

.....
Now, how can I save this file?
First press 'control' key then press 'o' key. It will ask for
file name. Type it as 'novel.txt', then press 'enter' key.
To exit the 'nano' text editor press 'control' and 'x'.
Voila!
You have successfully created a text file using 'nano'.
Congratulation!
hagudu@hagudu-H81M-S1:~$
```

Figura 5-5. Lectura de un archivo de texto con el comando "cat"

Normalmente, los programadores experimentados solían trabajar en el terminal y los editores de texto como "VI", "VIM" o "NANO" son muy populares.

Ahora vamos a aprender un comando muy importante de Linux llamado "grep". Este comando realiza algún tipo de búsqueda dentro de un archivo y lo hace de una manera muy interesante. Veamos primero lo que tenemos en nuestro directorio raíz.

Emitimos un comando como este en nuestro terminal y vemos la salida.

```
hagudu@hagudu-H81M-S1:~ $ cd
/etc/apt hagudu@hagudu-H81M-
S1:/etc/apt$ ls
apt.conf.d      sources.list
                  fuentes.lista.savetrusted.gpg
trusted.gpg.d
preferences.d   sources.list.d  trustdb.gpgt
rust.gpg ~
hagudu@hagudu-H81M-S1:/etc/apt$
```

Como puede ver, hemos cambiado el directorio a "/etc/apt" y vemos la lista. Encontramos muchos archivos allí y actualmente estamos interesados en el archivo "sources.list". Podemos usar el comando "cat" para leer el archivo pero tenemos algo diferente en mente.

Nos gustaría buscar alguna palabra en particular y queremos separarlas y

verlas en segregación. El comando "grep" junto con otro

"|" (pipa), nos ayudará a hacerlo.

En realidad, le decimos al terminal que primero muestre el contenido de "sources.list" y luego canalizar ese término a nuestro proceso de búsqueda. Veamos cómo funciona.

Si simplemente escribimos un comando como "cat sources.list", mostrará una larga lista de las fuentes de este sistema Linux. Puedes escribirlos y verlos. Pero estamos interesados en buscar la palabra "src" y queremos ver cuántas veces se ha utilizado esa palabra en el "sources.list".

Así que el comando final y la salida son así:

```
hagudu@hagudu-H81M-S1:/etc/apt$ cat sources.list |  
grep src  
deb-src http://in.archive.ubuntu.com/ubuntu/  
trusty principal restringido  
deb-src http://in.archive.ubuntu.com/ubuntu/  
trusty-updates main restricted  
deb-src http://in.archive.ubuntu.com/ubuntu/  
universo de confianza  
deb-src http://in.archive.ubuntu.com/ubuntu/  
universo trusty-updates  
deb-src http://in.archive.ubuntu.com/ubuntu/  
confiable multiverso  
deb-src http://in.archive.ubuntu.com/ubuntu/  
trusty-updates multiverso  
deb-src http://in.archive.ubuntu.com/ubuntu/  
trusty-backports principal universo restringido  
multiverso  
deb-src http://security.ubuntu.com/ubuntu trusty-  
security main restricted  
deb-src http://security.ubuntu.com/ubuntu trusty-  
universo de seguridad  
deb-src http://security.ubuntu.com/ubuntu trusty-  
security multiverse  
# deb-src http://archive.canonical.com/ubuntu  
socio de confianza  
deb-src http://extras.ubuntu.com/ubuntu trusty  
main # deb-src http://archive.ubuntu.com/ubuntu  
trusty  
universo  
hagudu@hagudu-H81M-S1:/etc/apt$
```

Es interesante notar que primero dimos una orden como esta: gato

```
fuentes.list | grep src
```

Y la salida larga que sigue a ese comando tiene todas las sentencias que tienen "src" en él.

Incluso podemos filtrar el archivo fuente más claramente. Podemos reducir más nuestras búsquedas y decirle a la terminal que busque la palabra "src" sólo con letras minúsculas escribiendo este comando:

```
cat sources.list | grep -i src
```

En el futuro, usaremos este comando "grep" extensivamente para escanear una red con una palabra en particular.

Otro comando importante es "echo". Este comando literalmente hace eco de todo lo que usted escribe en su terminal. También puede hacer algo más con este comando. Incluso puede cambiar un archivo de texto con este comando.

Anteriormente hemos escrito un archivo de texto "novel.txt" y lo hemos guardado en nuestro directorio principal. Ahora vamos a sobreescibir ese archivo con este comando "echo".

```
hagudu@hagudu-H81M-S1:~ $ echo "NO ME GUSTA ESTA NOVELA PARA QUE LO CAMBIE" > novel.txt  
hagudu@hagudu-H81M-S1:~ $ cat novel.txt
```

YA NO ME GUSTA ESTA NOVELA, ASÍ QUE LA CAMBIO.

```
hagudu@hagudu-H81M-S1:~ $
```

Primero hemos hecho eco de algún texto en nuestro terminal, luego usamos ">" (más que signo) para poner ese texto en el archivo "novel.txt". En el siguiente comando, hemos vuelto a utilizar el comando "cat" para leer el archivo "novel.txt" y hemos encontrado que el archivo ha sido cambiado.

Ahora aprenderemos a crear directorios en Linux. Hay un comando muy útil: "mkdir". Significa claramente "hacer directorio". Hagamos un directorio con el nombre de este proyecto: "Hacking ético". Puedes adivinar que el comando es extremadamente simple: mkdir Ethical Hacking

No, no lo es. En este caso, si escribe de esa manera, el terminal Linux entiende otra cosa. Comprende que desea crear dos directorios separados. Uno es "Ético" y el otro es "Hacking". Ya ha creado dos directorios de esa manera. Así que vamos a eliminarlos primero y después vamos a crear un directorio con un significado más significativo.

Para eliminar un directorio, debe tener privilegios de "root". Significa que eres un

administrador o superusuario del sistema. En Ubuntu, si queremos ser un "root" o "superusuario", primero emitimos el comando "sudo". En Kali Linux es diferente: "su". Pero en ambos casos, una vez que escriba ese comando, el sistema le pedirá la contraseña a través del terminal. Veamos cómo funciona.

Primero emitimos el comando y en el siguiente paso comprobamos con el comando "ls" para ver si esos directorios ya existen.

```
hagudu@hagudu-H81M-S1:~ $ sudo rm -rf Ético/  
Hacking/  
sudo] contraseña para  
hagudu: hagudu@hagudu~  
H81M-S1: $ ls
```

Funcionó: se han eliminado dos directorios con éxito. Tratemos de entenderlo mejor. Ya sabemos que el comando "rm" significa la palabra "remove". ¿Pero qué pasa con el comando "-rf" que le sigue? El comando "-rf" significa "hacerlo recursivamente con fuerza". Generalmente este comando "-rf" se utiliza para eliminar directorios. Hay que tener mucho cuidado al usar este comando porque en Linux, una vez que se ha usado este comando, el archivo o directorio se borra permanentemente. Es casi imposible recuperarlos. Es prudente tener mucho cuidado al usarlo.

Esperamos que también hayas notado que hemos comenzado nuestra línea de comandos con "sudo". Y escribes "sudo", te pide la contraseña. En este caso, siempre se debe dar la contraseña que normalmente se escribe para iniciar sesión en el sistema.

Hagamos de nuevo el directorio correctamente y esta vez lo llamaremos "Ethical- Hacking", de modo que el sistema ya no lo interpretará como dos directorios separados.

```
hagudu@hagudu-H81M-S1:~ $ mkdir Ethical-  
Hacking hagudu@hagudu-H81M-S1: $ cd Ethical-  
Hacking/ hagudu@hagudu-H81M-S1: /Ethical-  
Hacking$ ls hagudu@hagudu-H81M-S1: /Ethical-  
Hacking$ touch  
archivo1 archivo2  
hagudu@hagudu-H81M-S1:~/Ethical-Hacking$ ls  
file1 fichero2  
hagudu@hagudu-H81M-S1:~/Ethical-Hacking$
```

Primero hemos hecho el directorio "Ethical-Hacking". Luego usamos "cd" para entrar en él y, con la ayuda de "ls", comprobamos que el directorio está vacío.

Después emitimos el comando "touch" para crear dos archivos: "file1" y

"Archivo 2". De nuevo emitimos el comando "ls" para comprobar que dos archivos han sido creados con éxito.

En Ethical Hacking, el anonimato es un gran problema. En el siguiente capítulo lo aprenderemos en gran detalle. Antes de eso necesitas entender que, en el proceso de ser anónimo, es bueno ser cualquier usuario en lugar de ser el usuario root. Como usuario root o superusuario, primero añada un usuario en su Linux virtual de Kali. Establezca una contraseña. Apaga a Kali. Reinicie e inicie sesión como el nuevo usuario. Es una buena práctica.

Ahora, ¿cómo se puede añadir un usuario? Abramos nuestro Kali virtual y como usuario root usaremos el comando "adduser" en el terminal. Supongamos que nuestro nuevo usuario tiene un nombre como "xman". En ese caso, el comando será muy simple: adduser xman.

Una vez que haya emitido este comando, Kali le pedirá la contraseña y otros detalles. Proporcione una contraseña segura de al menos ocho caracteres con caracteres alfanuméricos. Ahora apague su máquina e inicie sesión como "xman". Para otros detalles, no es obligatorio que usted tenga que dar su verdadera identidad. Puedes llenarlos con cualquier dato.

Como usuario root o superusuario puedes añadir tantos usuarios como deseas. Puede borrarlos en cualquier momento. Puede restringir sus actividades desde cualquier ángulo. Como administrador, puede añadir un usuario que no podrá iniciar sesión después de seis meses. Puede crear grupos y establecer una regla para que la entrada esté restringida. Algunos usuarios pueden entrar en ese grupo. Algunos no pueden.

En primer lugar, es necesario añadir un usuario, "xman", e iniciar sesión en el sistema como si fuera el nuevo. Un usuario no está autorizado a acceder o manipular ningún archivo del usuario root o superusuario. Pero como superusuario siempre puedes cambiar el permiso del archivo. Es un concepto muy importante desde todos los ángulos. En Internet, el concepto de permiso de archivo es extremadamente importante.

Cualquier archivo tiene tres tipos de permisos relacionados con él. Sólo puede ser de "sólo lectura". El significado es claro. No se puede escribir ni ejecutar. Puede ser "sólo para escribir". Otro estado del archivo es "modo ejecutable..." Si es ejecutable, puede realizar cualquier acción ejecutándola. Puedes escribir un simple programa Python.

Este programa tomará las entradas de los usuarios y dará salidas. Después de escribir un archivo Python puedes hacerlo ejecutable.

Veamos cómo sucede. Abramos nuestro terminal Kali Linux y, con la ayuda del comando "ls", veamos lo que tenemos allí actualmente.

```
sanjib@kali:~$ cd Documents/  
sanjib@kali:~/Documents$ ls  
VBoxLinuxAdditions.run
```

```
sanjib@kali:~/Documentos$ ls -la
total 7048
drwxr-xr-x  2 sanjibsanjib      4096 29 de mayo
10:30 .
drwxr-xr-x 18 sanjibsanjib      4096Jun   3 09:59
...
-rwxr-xr-x 1root    root      7208397 29 de mayo
10:30 VBoxLinuxAdditions.run
sanjib@kali:~/Documentos$
```

Primero vamos a la carpeta "Documents" y emitimos el comando "ls". Eso muestra sólo un archivo: "VBoxLinuxAdditions.run". Nuestro siguiente comando es "ls - la". Significa: queremos un listado de todos los archivos con todos los detalles. Puedes ver la diferencia arriba. La salida está en rojo. Muestra dos archivos ocultos con el archivo visto anteriormente. Y también muestra los propietarios de los archivos y también muestra los permisos. Consideremos esta línea minuciosamente.

```
-rwxr-xr-x 1root    root      7208397 29 de mayo
10:30 VBoxLinuxAdditions.run
```

Nos dice que el propietario de este archivo es "root". Y la línea de salida también es muy importante. Maneja los permisos de los archivos.

rwxr-xr-x

¿Qué significa esto? Tiene tres partes distintas. La primera parte es "r-x". La segunda y tercera parte son también la misma: "r-x". La primera parte es para el propietario del archivo o usuario actual. La segunda parte es para "grupo..." Y la parte final o tercera es para el superusuario que está viendo este archivo. Ya he creado otro usuario, "sanjib", y me he registrado como "sanjib". Es por eso que usted ve esto

tipo de salida: sanjib@kali:~/Documents\$ ls -la

Ahora para hacer más claro este concepto vamos a crear un usuario llamado "xman".

Y nos registraremos como "xman" y veremos lo que tenemos en nuestra carpeta Documentos.

Para crear un nuevo usuario, debe iniciar sesión como usuario root o superusuario. Supongamos que nos hemos registrado como "root". Los comandos y la salida se dan a continuación.

```
root@kali:~# adduser xman
Añadir usuario `xman' ....
```

Añadido nuevo grupo `xman' (1002) ....

Añadiendo un nuevo usuario `xman' (1001) con el  
grupo `xman'...

```
Creación del directorio raíz `/home/xman' ....
Copiando archivos desde `/etc/skel' ....
Introduzca una nueva contraseña de UNIX:
Vuelva a escribir la nueva contraseña de UNIX:
passwd: contraseña actualizada con éxito
Cambiar la información del usuario para
xman
Introduzca el nuevo valor, o pulse ENTER para el
    Nombre completo por defecto []: xman
    anonymous
    Número de
    habitación []: 123
    Teléfono del
    trabajo []: 321
    Teléfono
    particular []: 213
    Otro []: anon
```

¿Es correcta la información? [S/N] y  
root@kali:~#

Felicitaciones! Acaba de crear con éxito un nuevo usuario llamado 'xman'. Notarás que te ha pedido la contraseña y te ha dicho que vuelvas a escribir la contraseña de UNIX.

Salgamos como "root" y entremos como "xman". Vayamos también a la carpeta "Documentos" de "xman" y veamos lo que tenemos.

```
~
xman@kali:~$ cd Documentos/
xman@kali:~/Documentos$ ls
xman@kali:~/Documentos$ ls -la
total 8
drwxr-xr-x  2 xman xman xman      3 10:33 .
                4096 Jun
drwxr-xr-x 14 xman xman xman      3 10:33 ..
                4096 Jun
xman@kali:~/Documentos$
```

Todo va como se esperaba. Sólo falta una cosa. Este nuevo usuario no tiene esta línea: -r-xr-xr-x 1 root root 7208397 29 de mayo 10:30 VBoxLinuxAdditions.run.

Tal vez habíamos movido ese archivo ejecutable de cualquier carpeta raíz

a la carpeta "Documents" del usuario "sanjib" anteriormente.

Ahora ya sabemos cómo crear un archivo usando el editor de texto "nano". Así que podemos seguir adelante y tener un archivo Python muy pequeño... Presumiblemente no conoces a Python, así que lo mantengo muy sencillo sólo para mostrar cómo podemos cambiar los permisos de los archivos.

```
#!/usr/bin/python3
print("TType your name.")
inputs =
input(">>>>>")
outputs = inputs
def main():
    print(outputs)
si nombre _ == ...y que no se puede hacer nada más:
    main()
```

Dentro del editor "nano" escribimos un programa sencillo que toma las entradas y las salidas. Guardar el archivo como "pyfile.py" y salir de "nano", y vamos a emitir "ls - la" para ver lo que muestra.

```
xman@kali:~/Documentos$ ls -la
total 12
drwxr-xr-x 2 xman xman xman 4096 3 10:50 .
                Jun
drwxr-xr-x 15 xman xman xman      4096 3 10:42 ..
                Jun
-rw-r--r-- 1 xmanxman        86 Jun 3 10:44
r--
pyfile.py
xman@kali:~/Documentos$
```

Como ves, el archivo lo dice todo. Dice que ahora la carpeta "Documents" tiene un nuevo archivo, "pyfile.py", y que ha sido creado a las 10:44. El propietario es "xman" y tiene permisos de archivo como los siguientes: **rw-r--r--**

Ahora sabes lo que esto significa. Esto significa: el usuario "xman" puede leer y escribir este archivo pero no puede "ejecutar" este archivo.

```
xman@kali:~/Documento$ chmod +x pyfile.py
xman@kali:~/Documentos$ ls -la
total 12
drwxr-xr-x 2 xman xman 4096Jun 3 10:50 .
drwxr-xr-x 15 xman xman 4096Jun 3 10:42 ...
-rwxr-xr-x 1 xmanxman     86Jun 3 10:44
pyfile.py xman@kali: /Documentos$
```

Mira cómo hemos usado el comando "chmod" para cambiar el permiso del archivo a ejecutable. Una vez que haya cambiado el permiso de archivo a ejecutable, cambiará el color a verde. Y también mira el permiso del archivo: **rwxr-xr-x**

La primera parte la marqué como roja para que puedas entender la diferencia

entre ellos. La primera parte del permiso dice "x" ha sido añadida desde que usamos el comando "xman@kali:~/Documents\$ chmod +x pyfile.py".

Ejecutemos el archivo y veamos cómo toma la entrada y da la salida.

```
xman@kali:~/Documents$ ./pyfile.py
```

Escriba su nombre.

```
>>>>>DIFUNDE  
LA PALABRA-
```

Cuando ejecuta el archivo, le pide que escriba su nombre y devuelve suavemente la salida.

---

## Resumen

Usted ha aprendido algunos comandos básicos de Linux en este capítulo. Ahora al menos tienes una idea de cómo funciona un sistema Linux y cómo puedes usar tu terminal o línea de comandos para operar tu sistema.

En el proceso de aprendizaje del hacking ético, lo encuentras extremadamente útil. En el futuro, necesitará aprender algunos comandos más de Linux. Su conocimiento de Linux o de cualquier otro sistema operativo debe ser encomiable. A medida que progresas, espero, tu "apetito viene con la comida".

Hemos discutido suficientes rudimentos como para pisarnos los pies sobre la base del pirateo ético. Ahora es el momento de seguir adelante. Estamos listos para dar el primer paso importante en el mundo del hacking ético aprendiendo un lenguaje de programación muy útil: Python 3.

Hemos hablado de Python 3 de una manera que no necesita conocimientos de programación. Se ha discutido ampliamente para que puedas llegar a la etapa intermedia y escribir tu propio programa en Python 3. A medida que progresas en el vasto universo del hacking ético, encontrarás la importancia de aprender Python.

## Parte II

## 6. Python 3 y el Hacking Ético

Sanjib Sinha 

(1) Howrah, Bengala Occidental, India

---

Python puede hacer muchas cosas, especialmente en el campo de los enchufes y las redes. Además, en la monitorización del sistema tiene una gran importancia. En el nivel avanzado de hacking ético puede lanzar un hechizo mágico. Usted puede escribir su propio programa Python para cualquier tipo de propósito de seguridad.

Recuerde, cualquier programa escrito en Python o en cualquier idioma emite algunas instrucciones. Y son más o menos iguales. Lo son:

**ENTRADA:** Obtenga datos del teclado o de cualquier archivo o fuente.

**SALIDA:** Visualizar los datos en pantalla o enviarlos a cualquier archivo, dispositivo o cualquier otra fuente.

**MATEMÁTICA:** Haga algunas operaciones matemáticas básicas como sumar, restar, multiplicar o dividir. También puede ser complejo. Depende de su aplicación.

**EJECUCIÓN CONDICIONAL:** Compruebe que las condiciones se cumplen correctamente. Como "si eso es verdad, entonces haz algo más/haz algo más".

**REPETICIÓN:** Realice alguna acción repetidamente.

La mayoría de la gente solía tener una plataforma Windows o Macintosh en su casa. Antes de empezar, le pido que pruebe Linux como un sistema operativo dual. Hay muchas distribuciones de Linux gratuitas y fáciles de usar. Puede probar Ubuntu, o cualquier paquete de Debian. Simplemente descargue la imagen ISO estable y grábela en un DVD e instálela junto con su sistema operativo en funcionamiento. Ayudará. Python viene con todas las distribuciones de Linux.

El Linux disponible se ejecutará dentro de Windows, así que cuando quiera probar cualquier lenguaje de programación de código abierto como Python o PHP, puede aprovecharlo. Puede probar el terminal Linux siempre que sea necesario.

Básicamente, Python viene con cualquier distribución de Linux, por lo que

no necesita

preocuparse por la instalación en Linux. Eso también es una ventaja.

Si quieres mantenerte en Windows, visita la sección de descargas del sitio oficial de Python. Según la configuración de su sistema, descargue el archivo "python-3.4.4.4.tar.xz" para Windows. Cuando extraiga ese archivo, obtendrá el "python-3.4.4.4 Windows Installer Package". Sólo tienes que ejecutarlo y seguir los sencillos pasos. Le sugiero que descargue la documentación junto con el paquete de instalación. Esta documentación es extremadamente útil, no sólo para principiantes sino también para programadores experimentados. Después de la descarga, abra la documentación.

Esta documentación está diseñada exclusivamente para programadores, no para principiantes. Pero como principiante, necesita acostumbrarse a este manual para que después de un cierto período, se convierta en parte de su vida de programación.

Casi todos los problemas de programación posibles se discuten en esta documentación y, además, se puede desarrollar el código y crear una aplicación impresionante con la ayuda de esta documentación.

Se parece a esto:



**Figura 6-1.** Página de documentación de Python 3

## 7. Medio ambiente de Python

Sanjib Sinha 

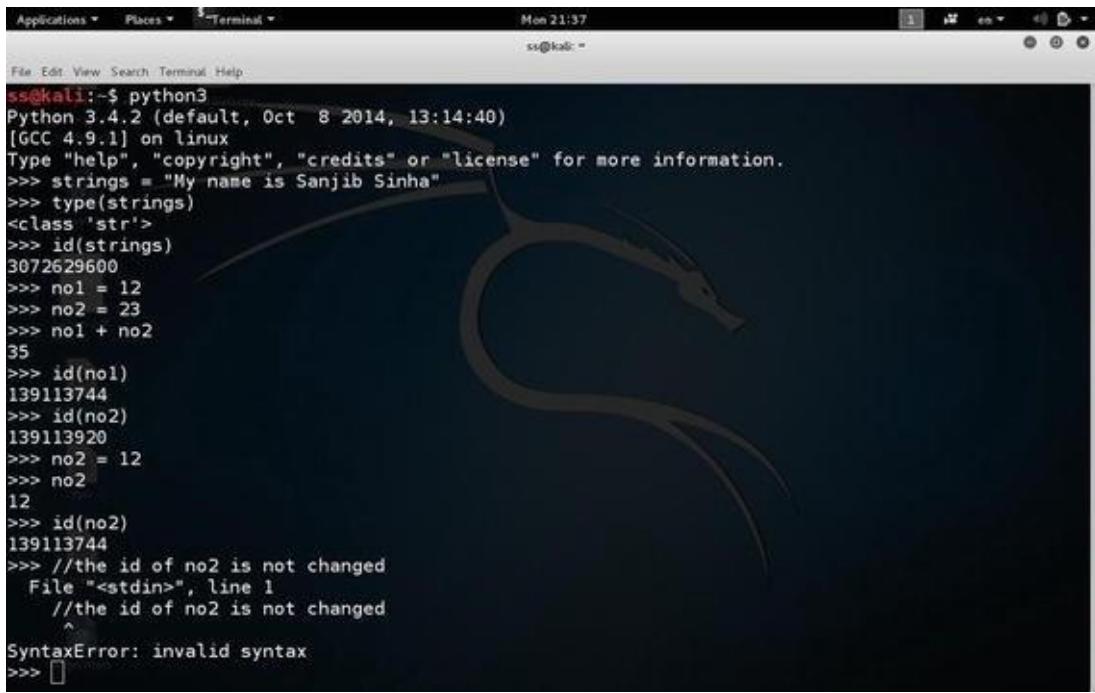
(1) Howrah, Bengala Occidental, India

---

Vas a aprender Python 3. Python 2 existe desde hace mucho tiempo y tiene una gran biblioteca y soporte de módulos, pero Python 3 es el lenguaje del futuro. También puede instalar fácilmente Python 3. Consulte la sección de descargas de la página web oficial. En cualquier distribución Linux moderna, abra su terminal y escriba "python3". Te dará el intérprete de Python o shell donde puedes escribir tu código.

Recuerde, Python viene con todas las distribuciones modernas de Linux. Por lo tanto, ya no es necesario que lo instale. Sin embargo, es posible que necesite instalar algunos paquetes. Hay toneladas de tutoriales y mucha ayuda de la comunidad que puede obtener a través de Internet.

El intérprete de Python en una distribución típica de Linux tiene este aspecto:



```
File Edit View Search Terminal Help
ss@kali:~$ python3
Python 3.4.2 (default, Oct  8 2014, 13:14:40)
[GCC 4.9.1] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> strings = "My name is Sanjib Sinha"
>>> type(strings)
<class 'str'>
>>> id(strings)
3072629600
>>> no1 = 12
>>> no2 = 23
>>> no1 + no2
35
>>> id(no1)
139113744
>>> id(no2)
139113920
>>> no2 = 12
>>> no2
12
>>> id(no2)
139113744
>>> //the id of no2 is not changed
  File "<stdin>", line 1
    //the id of no2 is not changed
      ^
SyntaxError: invalid syntax
>>> 
```

Figura 7-1. Imagen del intérprete de Python

En cualquier distribución moderna de Linux, no necesita hacer nada. Abra el terminal y escriba "python3", y tendrá una salida como esta:

```
hagudu@hagudu-H81M-S1:~ $ pitón3
Python 3.4.3 (por defecto, 14 de octubre de
2015, 20:28:29) [GCC 4.8.4] en linux
Para más información, escriba "ayuda", "copyright",
"créditos" o "licencia".
>>>
```

Dice que mi ordenador tiene Python 3.4.3. Ahora puedes escribir algo de código directamente sobre él para obtener una salida como esta:

```
>>> nombre = "Sanjib"
>>>
imprimir(nombr
e) Sanjib
>>>
```

En Linux, primero se guarda un archivo Python. Escribe este código:

```
<código>
```

```

#!/usr/bin/python3
def main():
    print ("Hola Python!")
si nombre __ == " principal ":
    main()
</código>

```

Si es nuevo en Linux, primero guarde este archivo Python como "hello.py" y luego cámbielo a ejecutable con este comando:

```
sudo chmod +x hello.py
```

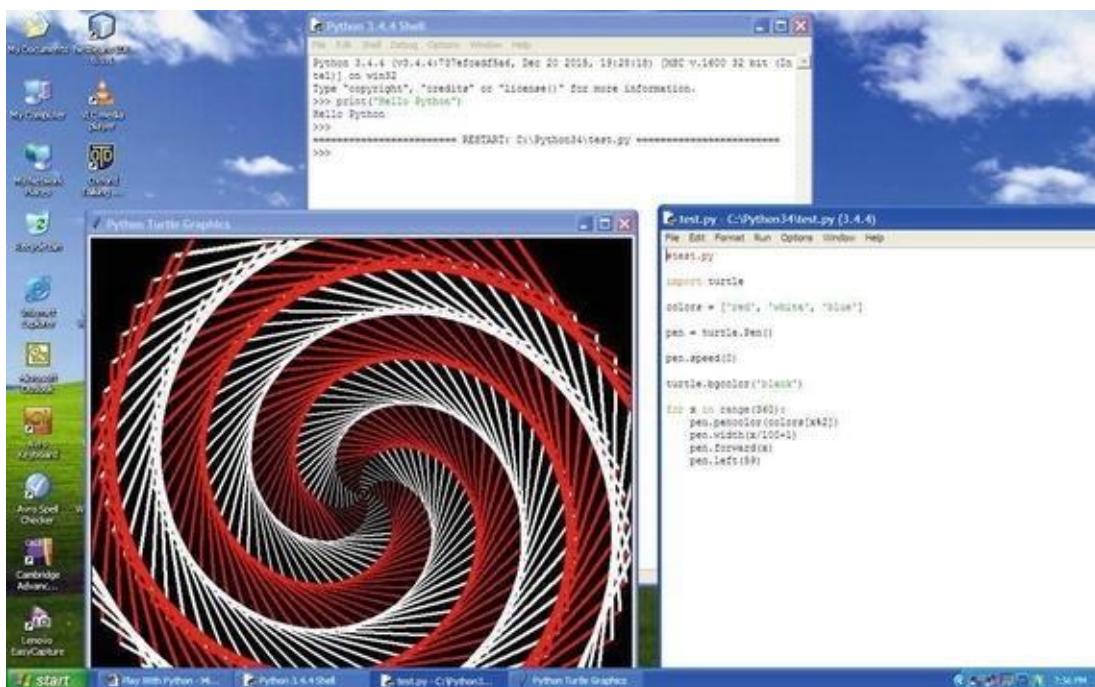
En el terminal, ejecute este archivo con este comando :

```
. /hola.py
```

Dará la salida: **¡Hola Python!**

Este es tu primer código Python.

Para Windows, descargue el instalador y el documento de Python. El documento viene en un archivo ".chm". Ayudará más tarde. Para instalar Python, simplemente ejecute el instalador. Se instalará en su unidad "C" en un minuto. Ahora puedes ir a "todos los programas" y ejecutar Python desde allí. Normalmente, un pequeño IDE llamado IDLE viene con Python. Puedes escribir código y simplemente ejecutarlo. Veamos cómo se ve:



*Figura 7-2.* Python IDE en Windows

En la imagen de arriba, se ve en la parte superior es IDLE, que es el Python Shell. Se puede obtener salida directamente de él. También puede ir a la sección de archivos de IDLE y crear un nuevo archivo. Ya lo he hecho. Creé un archivo, "test.py", y escribí algo de código en él. Luego, desde IDLE puede ejecutar este módulo o simplemente presionar F5 y seguirá funcionando. Como puede ver en el dibujo, nuestro código Python dibujó una hermosa forma. En Windows 7 o posterior, puede abrir Power Shell y escribir lo mismo y obtendrá el mismo resultado. Pero prefiero que primero instales un buen editor de texto Python o IDE.

Para Linux, la edición comunitaria "Pycharm" es una buena elección. Es gratis. Para Windows o Mac, hay varios buenos editores de texto libre. Busque en Internet e instale. La principal ventaja es que no es necesario sangrar cada línea de código. Está automatizado. En segundo lugar, el soporte de una gran librería Python está disponible en cada IDE.

Sanjib Sinha 2017

Sanjib Sinha, Comenzando el Hacking Ético con Python, DOI 10.1007/978-1-4842-2541-7\_8

---

## 8. Sintaxis General

Sanjib Sinha 

(1) Howrah, Bengala Occidental, India

---

En este capítulo aprenderemos algo sólo para probar algunos códigos. Aprenderemos las mismas cosas en detalle más adelante. Todo lo que tenemos que hacer ahora es intentar escribir algo de código en Python y ver cómo funciona. Al mismo tiempo, aprenderemos sobre las sintaxis generales que se usan a menudo en Python.

---

### Crear la función main()

Como dije, los guiones Python son casi como el lenguaje humano. No es necesario utilizar muchos caracteres o símbolos especiales. Todo lo que necesitas recordar es que la "sangría" juega un papel muy importante en Python. Cuando usted aplica algunas condiciones especiales dentro de su código, esta sangría es importante.

Pocas cosas se repiten en cada código. Así que puedes escribirlo en un archivo separado y utilizarlo en cada nuevo archivo de Python. El código de estructura sintáctica general tiene el siguiente aspecto:

```
/usr/bin/python3
def main():
    print("Soy un archivo Python de sintaxis
          general")
    si nombre __ == " principal__":
        main()
</código>
```

Guarde este archivo como "general-syntax.py". Cuando ejecute este fichero, dirá o imprimirá: "Soy un archivo Python de sintaxis general."

La primera línea, "`#!/usr/bin/python3`", denota el camino del intérprete de Python. La grandeza de Python es que permanece igual en todos los sistemas operativos. En la segunda parte hemos definido una función `main()` y, bajo esa función `main()`, podemos llamar a cualquier función. Sin una función `main()`, no se puede llamar a una función antes de que sea definida. Considere este ejemplo:

```
/usr/bin/python3
def main():
    print("Soy un archivo Python de sintaxis
          general") LetUsDoSomething()

def LetUsDoSomething():
    print ("Estoy haciendo algo")

si nombre __ == " principal__":
    main()
</código>
```

Ahora dará un buen resultado como este:

```
Soy un archivo Python de sintaxis
general Estoy haciendo algo
```

Suponga que no tiene ninguna función `main()`. Ahora bien, si desea llamar a la función `LetUsDoSomething()` antes de que se defina esa función, se producirá un error.

Pruebe este código y vea el error:

```
<code>
#!/usr/bin/python3
LetUsDoSomething()
def LetUsDoSomething():
    print ("Estoy haciendo algo")
</código>
```

Dice: `NameError LetUsDoSomething() no está definido.` Siempre se puede llamar después de definir la función. En ese caso, no necesita definir la función `main()`. Pero en una larga línea de código donde muchas funciones son

no siempre es posible mantenerlo. Para resolver este problema, es una buena práctica definir primero la función main(). Después de eso puede escribir cualquier función después de la función main() y llamarla dentro de la función main().

---

## Sangría y espacio en blanco

Desempeñan un papel muy importante cuando trabajas con Python.

La hendidura o espacio en blanco es muy, muy importante. Antes de que empieces a aprender Python, tienes que entender esto correctamente. Considera este código:

```
< código
# coding=utf-8
def main():
    print('Una línea dentro de la función
principal.') print("Una línea fuera de la
función principal.")
si nombre __ == main():main()
</ código>
```

Mira este código. La función print() dentro de la función main() ha sido sangrada. Tiene unos cuatro espacios. Y la segunda función print() está fuera de la función main(). Y mira el código; cae en la misma línea que la función main(). Así que cuando ejecutamos este programa, la función externa print() se ejecuta primero. Y la salida es así:

```
//output
Una línea fuera de la
función principal. Una línea
dentro de la función
principal.
Salida //output finalizada
```

Si intentamos empujar un poco hacia dentro la función print() exterior, se producirá un error, porque el intérprete de Python pensará que está dentro de la función main(). En realidad esto no es cierto. Si queremos empujar esa función "outside print() function" dentro de la función main(), tenemos que colocarla en la misma línea de la función inside print() así:

```
< código
# coding=utf-8
```

```
def main():
    print('Una línea dentro de la función
principal.')
```

```

    print("Una línea fuera de la función
          principal.")
si nombre __ == main():main()
</código>

```

Ahora la salida cambia. Se parece a esto:

```

//output
Una línea dentro de la
función principal. Una línea
fuera de la función
principal.
Salida //output finalizada

```

Aprendemos una lección muy importante que debemos aprender de memoria. La lección es: el espacio en blanco o indentación en Python juega un papel importante. Cuando escribimos una función y ponemos otras funciones dentro de ella, deben caer en la misma línea. En cualquier editor de texto o IDE, se hace automáticamente. Al pulsar la tecla "enter" o "return", las siguientes líneas siguen cayendo en la misma línea. Si desea salir de esa función, sólo tiene que seguir el primer ejemplo. Para entender cómo funciona la indentación en Python, escribimos un poco de código largo y vemos cómo se ve.

```

<código
# coding=utf-8
def main():
    # print('Una línea dentro de la función
    # principal.') #
    # print("Una línea fuera de la función
    # principal.")
    OutsideMainFunction()

def
OutsideMainFunction():
    x = 0
    mientras que x < 5:
        print(x)
        x = x + 1
si nombre __ == main():main()
</código>

```

Mira el código. Tenemos una función main(). Además, tenemos una

función llamada "OutsideMainFunction() '". Está realmente fuera de la función main(). Así que son funciones diferentes y tienen sus propias hendiduras. Dentro de la "OutsideMainFunction()" vemos un "while loop". Que "mientras

"loop" también tiene su propia sangría. En realidad, será mejor que lo llamemos "bloque". Así que cada bloque de código tiene su propio "espacio en blanco" o el código dentro de ese bloque está indentado en consecuencia. Si no usas ningún IDE y tratas de escribirlo en tu terminal, tienes que usar la barra espaciadora. Dentro de una función, si usas "cuatro espacios", entonces lo que escribas dentro de esa función debe caer en la misma línea. Es decir, siempre que escriba una nueva línea, debe tener "cuatro espacios". No se puede dar dos o tres espacios de repente. Pero si escribe otra función, puede cambiar esa regla. En ese caso, la nueva función tiene su propio bloque de código y tiene su propia regla. Ahora puede usar dos espacios.

---

## Comentando

En cualquier tipo de programación, los comentarios son muy importantes. Otro programador leerá su programa. Cada uno de sus pasos debe ser legible. Si hay algún tipo de giro o intentas algo especial, debes explicarlo dentro de tu código. Considere este código:

```
< código
# esta es la función main()
def main():
    Función principal externa()
# esta función está fuera de la función main()
def
    OutsideMainFunction()
    : x = 0
    mientras que x < 5:
        print(x)
        x = x + 1
si nombre __ == main():main()
</ código>
```

Normalmente cualquier comentario se escribe con una marca # (hash). Cuando el intérprete de Python ve #, sabe que es un comentario y lo ignora. En nuestro código, definimos claramente cuál es la función main() y también decimos en nuestros comentarios que hay otra función que está fuera de la función main().

Normalmente un programador experimentado nunca comenta cosas tan simples. Pero para empezar, puedes añadir comentarios cuando lo consideres necesario. Porque después de un tiempo, cuando vuelvas a visitar tus viejos códigos, puedes recordar por qué lo hiciste. Comentar es útil en ese sentido. Al mismo tiempo, no se puede confiar en todos los comentarios. Los

programadores a menudo se olvidan de cambiar los comentarios cuando

cambiar sus códigos.

---

## Asignación de valores

En Python, el operador de asignación es un signo igual (=). Cuando escribes "a = 10", significa que "a" es una variable o un contenedor. Esta variable "a" se asigna a un valor entero. ¿Cuál es ese valor? Son 10. Este valor podría haber sido una cadena. ¿Qué es una cadena? Una cadena es una suma de caracteres. Suponga que escribe "b = Dos". Significa que la variable "b" se asigna a un valor de cadena, y esa cadena es "Dos", que no es más que tres caracteres: "T"+ "w" + "o".

De acuerdo a su asignación, Python interpreta el valor y mantiene un lugar de almacenamiento definido para ellos. Sabe cuántos bits y bytes serán necesarios para ellos.

En Python, todo es objeto. Python es un lenguaje de programación orientado a objetos. Como principiante, es posible que no entienda este concepto. No te preocunes.

Lo discutiremos en detalle a medida que avancemos. Lo aprenderás. En este momento sólo recuerda que un objeto significa una instancia de clase.

Imagínate a ti mismo como un objeto. En ese caso, usted es un ejemplo de clase "humana". Usted tiene algunas propiedades como altura, anchura, etc. Tú también puedes hacer algo. La clase "humana" es un modelo de ti y de los demás humanos, y en la clase "humana" todo está bien definido. Hay muchas propiedades y muchos verbos de acción definidos. Y según esa definición, tú, yo y otros humanos seguimos haciendo cosas.

Cuando decimos en Python que todo es un objeto, significa que todo tiene una clase o plano detrás. Escribimos así:

```
/usr/bin/python3 #
coding=utf-8
a = 1
imprisión(a)
imprisión(tipo
(a))
imprisión(id(
a)) a = "One"
imprisión(a)
imprisión(tip
o(a))
imprisión(id(a
))
</código>
```

Y la salida es así:

```
//output
1
<clase
'int'>
139113568
Uno
<Clase
"str'>
3073583584
Salida //output finalizada
```

En el siguiente capítulo aprenderemos más detalladamente.

## 9. Variables, objetos y valores

Sanjib Sinha 

(1) Howrah, Bengala Occidental, India

---

En Python todo es un objeto. Para empezar, necesitas recordar algunas cosas:

- 1. Las variables, funciones e incluso el código son objetos.**
- 2. Cada objeto tiene un ID, un tipo y un valor.**  
ID significa identificación de una instancia concreta de un objeto.  
Este ID no puede cambiar durante la vida útil de ese objeto.
- 3. El tipo identifica una clase de un objeto. No puede cambiar durante la vida del objeto.**
- 4. El valor es el contenido del objeto y los objetos mudables sólo pueden cambiar el valor. Los objetos inmutables no pueden cambiar de valor.**
- 5. Cada variable en Python es un objeto de primera clase. Lo que parece una simple variable en realidad es algo más complejo.**

Veamos qué significan estos términos.

```
/usr/bin/python3
def main():
    x = 1
```

```

print(x)
print(id(x))
print(type(x))
x = 2
print(x)
print(id(x))
print(tipo(x))
) x = 1
print(x)
print(id(x))
print(type(x))

si nombre __ == " principal ":
    main()
</código>

```

Aquí está el resultado:

```

<cotización en bloque>.
1
10455040
clase 'int'
2
10455072
clase 'int'
1
10455040
clase 'int'.
</blockquoteo>

```

Como puede ver, la modificación de los valores de "x" no afecta a los objetos inmutables y el identificador único del objeto "1" sigue siendo el mismo. Lo que se ha cambiado es simplemente la referencia de la variable. Primero, nos referimos a "1" (objeto entero inmutable) a "x" (variable), y luego lo cambiamos. El ID y el tipo siguen siendo los mismos.

Recuerde, los números, las cuerdas y las "tuplas" son inmutables. Las listas, diccionarios y otros objetos son mutables (cambiables), pero eso depende.

Veamos un ejemplo muy breve que se explica en la sección de comentarios. La salida se da junto con ella.

```
<código
#!/usr/bin/python3
# in python everything is object
# una variable es una referencia a un
# objeto # cada objeto tiene una
# identidad o un ID x = 1
print(type(x))
print(id(x))
#####
# clase 'int'
# 139113568
#####
# número, cadena, tupla -> immutable
# lista, diccionario -> mutable
x = 1
y = 1
print(type(x))
print(id(x))
print(type(y))
print(id(y))
print(id(y))
if x == y:
    print
    ("Verdadero"):
        print("Falso")
si x es y:
    print
    ("Verdadero"):
        print("Falso")
#####
# see the last two lines, both are
true # class 'int'
# 139113568
# clase 'int'
# 139113568
#
Verdad
ero #
Verdad
ero
#####
#####
```

```
a = dic(x = 1, y = 1)
```

```

print(type(a))
print(id(a))
b = dic(x = 1, y = 1)
imprimir(id(b))
si a == b:
    print
    ("Verdadero"):
        print("Falso")
si a es b:
    print
    ("Verdadero"):
        print("Falso")
#####
# mira las dos últimas líneas, una es
verdadera pero la identificación no es la misma así
que es falsa.
# clase 'dict' #
# 3072650252
# 3072692524
#
Verdad
ero #
Falso
#####
para i en el rango(0,
    3): print(i, "=",
    id(i))
#####
# 0 = 139113552
# 1 = 139113568
# 2 = 139113584
#####
</código>

```

Vemos la salida dentro del código. Notarás que cada salida es comentada de manera que cuando ejecutemos este código, nunca afectará al script principal. Hay muchos valores. Números enteros, cadenas, tuplas, listas y finalmente diccionarios.

Ahora entenderemos lo que realmente son y cómo funcionan.

## Uso de números

En Python hay dos tipos de números. Uno es un entero y el otro es un flotador. Tenemos métodos incorporados en Python que pueden cambiar un entero a un flotador y cambiar un flotador a un entero. Espero que usted entienda el código de abajo.

El resultado es autoexplicativo. Lea el comentario también.

```
/usr/bin/python3
def main():
    x = 3
    print(x)
    print(id(x))
    print(type(x))
    print("*****")
    **) x = 3 /2
    print(x)
    print(id(x))
    print(type(x))
    print("*****")
    **) x = round(42
        / 9) print(x)
    print(id(x))
    print(type(x))
    print("*****")
    *)
    # Queremos redondearlo
    x = 42 // 9
    print(x)
    print(id(x))
    print(type(x))
    print("*****")
    ")
    # a cuántos dígitos queremos redondear
    x = redonda(42 / 9,
        3) print(x)
    print(id(x))
    print(type(x))
    print("*****")
    ) x = 43 % 7
    imprimir(x)
```

```

print(id(x))
print(type(x))
print("*****")
**) x =
int(34.78)
print(x)
print(id(x))
print(type(x))
print("*****")
*)
print("*****")
*)
print("*****")
*)
print(id(x))
print(type(x))
print(x)
si nombre __ == " principal ":
    main()
</código>

```

Y aquí está el resultado que obtenemos de este código:

```

<cotización en bloque>>.
3
10455104
clase 'int'.
*****
1.5
140223146811728
clase 'float'
*****
4
10455136
clase 'int'.
*****
5
140223146823568
clase 'int'.
*****
4.667
140223146811968

```

```
clase 'float'
*****
1
10455040
clase 'int'.
*****
34
10456096
clase 'int'.
*****
23.0
140223146811968
clase 'float'
*****
</blockquote>
```

Como se puede ver en la salida, cada número tiene un género y un ID. Para los números, este ID es inmutable. Así que si asigna el mismo número (suponga que es 1) a dos variables diferentes, así: a = 1 y b = 1; el ID de "a" y "b" es el mismo.

---

## Cadena

En Python la cadena es un objeto inmutable y puede escribirse entre comillas dobles o simples. Considere este código:

```
/usr/bin/python3
def main():
    "Te amo".
    print(cadenas)
    anotherStrings = "Te amo pero no sé cuánto me
amas."
    print(anotherStrings)
si nombre __ == " main " :
    main()
</código>
```

Y aquí está el resultado:

```
<cotización en bloque>.  
Yo te quiero.  
Te quiero, pero  
No sé cuánto me quieres.  
</blockquote>
```

Como ves, usamos una barra invertida para conseguir una nueva línea. Y tenemos un descanso exacto donde lo necesitábamos.

También hay salida de cadena en bruto. Vea este código:

```
/usr/bin/python3  
def main():  
    "Te amo".  
    print(cadenas)  
    anotherStrings = "Te amo pero no sé cuánto me  
amas."  
    print(anotherStrings)  
    rawStrings = r"Te amo pero no sé cuánto me  
amas."  
    print(rawStrings)  
    si nombre _ == " main " :  
        main()  
</código>
```

Y aquí está el resultado:

```
<cotización en bloque>.  
Yo te quiero.  
Te quiero, pero  
No sé cuánto me quieres.  
Te quiero, pero no sé cuánto me quieres.  
</blockquote>
```

La última declaración se llama una cadena en bruto, donde una barra invertida ya no funciona y obtenemos una salida en bruto. Y se usa en la expresión regular. Lo discutiremos en detalle en nuestro capítulo de expresiones regulares.

Podemos insertar un entero en el medio de una cadena. Les muestro los dos métodos usados en Python 2 y Python 3 pero recuerden, es mejor que se apeguen a los

utilizado en Python 3.

Veamos primero el código de Python 2:

```
< código  
días = 8  
lyrics = "%s días a la semana no es suficiente  
para amarte." %days  
print(lyrics)  
</ código>
```

La salida es así:

```
<cotización en bloque>>.  
8 días a la semana no es suficiente para amarte.  
</blockquote>
```

Veamos ahora el código de Python 3:

```
< código  
días = 8  
lyrics = "{} días a la semana no es suficiente  
para amarte."  
print(lyrics.format(days))  
</ código>
```

La salida:

```
<cotización en bloque>>.  
8 días a la semana no es suficiente para amarte.  
</blockquote>
```

¿Cuál es la mayor diferencia entre estas dos construcciones? La diferencia está en la última versión de Python; tratamos la cadena como un objeto. De ahí que un objeto "lyrics" usara un método llamado `format()` y pasara un parámetro que quería formatear en él. En la línea `print(lyrics.format(days))` usamos un punto ("."), para llamar al método `format()` que está incorporado en la clase `string`.

En tu vida de codificación necesitas usar muchas cuerdas y algunas de ellas pueden tener múltiples saltos de línea. No se puede utilizar la barra invertida "n" cada vez. Es engoroso.

Hay un truco que puedes usar en Python para usar varias líneas nuevas.

```
<code>
newLines =
"""\\ primera
línea segunda
línea tercera
línea más por
venir... """
print(newLines)
</código>
```

En la salida, las líneas se dividen automáticamente.

```
<cotización en
bloque>
primera línea
segunda línea
tercera línea
más por
venir.....
</blockquote>
```

Ahora puede usar comillas simples en lugar de dobles. Puede usar ninguna barra invertida al principio, pero eso generará un espacio al principio de la línea.

---

## Qué es Tipo e ID

Python es un lenguaje de programación orientado a objetos. Todo es un objeto aquí. Cada objeto tiene una identificación única, que se conoce como ID. Abramos nuestro terminal en Linux o, si tiene Windows o Mac, abramos el Python Shell y probemos este código:

```
<código>

>>> x = 10
>>> x
10
>>> tipo(x)
clase <clase 'int'>
>>> id(x)
10455328
```

```
>>> y = 10

>>> y

10

>>> tipo(y)

clase <clase 'int'>

>>> id(y)

10455328

>>> a = dict(name='sanjib')

>>> a

{'nombre': 'sanjib'}

>>> tipo(a)

clase <class 'dict'>

>>> DNI(a)

13998431868353592

>>> b = dic(name='sanjib')

>>> b

{'nombre': 'sanjib'}

>>> tipo(b)

clase <class 'dict'>

>>> DNI(b)

13998431868683720
```

```
>>> a == b
```

**Verdadero**

```
>>> a is b
```

**Falso**

```
>>>  
</código>
```

Aquí se asigna primero un valor entero "10" a la variable "x" y después el mismo valor a "y". Más tarde comprobamos el ID de dos variables y encontramos que el ID es el mismo. Ya lo dijimos en la sección anterior. Ahora ves el resultado.

Podemos comprobar si dos objetos asignados a dos variables diferentes son iguales o no escribiendo de esta manera:

```
<código  
>>> x == y
```

**Verdadero**

```
>>> x is
```

```
y y
```

```
Verdadero  
>>>  
</código>
```

Aquí es evidente que tanto las variables "x" como "y" apuntan al mismo objeto entero, "10". Así que el valor es el mismo y las variables también son las mismas. Pero no ocurrió en el caso de un objeto del diccionario que habíamos escrito justo después de eso.

El diccionario "a" y "b" tienen el mismo valor, pero como los objetos del diccionario son mutables, cambia el ID.

```
<código  
>>> a = dict(name='sanjib')  
  
>>> a
```

```
{'nombre': 'sanjib'}

>>> tipo(a)

clase <class 'dict'>

>>> DNI(a)

13998431868353

592

>>> b = dic(name='sanjib')

>>> b

{'nombre': 'sanjib'}

>>> tipo(b)

clase <class 'dict'>

>>> DNI(b)

13998431868683

720

>>> a == b

Verdadero

>>> a is b

Falso

>>>
< código >
```

Dice que el ID del diccionario cambia, aunque dos variables tienen los mismos valores. Cuando lo comprobamos lógicamente, dice, sí, el valor de dos variables es el mismo, pero como el ID es diferente, son objetos diferentes.

Como principiante, puede que encuentre este concepto un poco extraño. Pero más tarde, a medida que avance, encontrará que este concepto es extremadamente útil. Un objeto de diccionario necesita ser cambiado para

propósitos de programación. Si dos diccionarios

tienen el mismo ID, no podemos modificarlos.

---

## Valores Lógicos

Consideremos otro script de shell para probar los valores lógicos: Verdadero y falso.

```
< código  
->>> a, b = 0, 1  
  
->>> a == b  
  
Falso  
  
->>> a < b  
  
Verdadero  
  
->>> a > b  
  
Falso  
  
->>> a = Verdadero  
  
->>> a  
  
Verdadero  
  
->>> tipo(a)  
  
<clase de'bool'>  
  
->>> id(a)  
  
10348608  
  
->>> b = Verdadero  
  
->>> b  
  
Verdadero
```

```
>>> tipo(b)

<clase de'bool'>

>>> id(b)

10348608

>>>

</código>
```

Aquí vemos que hay clases "bool" y el operador "==" representa la prueba de calidad entre dos valores. Dado que "a" tiene un valor de 0 y "b" tiene un valor de 1, la salida es "False". ¿La "a" es menor que la "b"? Sí. Así que la salida sale como "True".

Estos "Verdadero" y "Falso" representan clases "bool". Y es "inmutable", así que si dos variables son ambas "Verdaderas" tienen el mismo ID.

---

## Tuplas y Listas.

Python tiene muchos tipos secuenciales (listas de cosas). Consideremos este código:

```
<código
x = (1, 2, 3, 4)
print(x)
print(type(x))
</código>
```

Tiene una salida como esta:

```
<cotización en bloque>.
(1, 2, 3, 4)
clase 'tuple'
</blockquote>
```

Así que es de la clase "tuple" y tiene una lista de cosas. Recuerda, la tupla es inmutable. No se puede insertar o actualizar. Pero puedes iterar a través de ella de esta manera:

```
< código  
para la i en x:  
    imprimir(i)  
</ código>
```

Le dará todos los números que tiene dentro de la tupla.

Por el contrario, "list" es otro tipo secuencial que es mutable y se puede cambiar como sea necesario. Considere este código:

```
< código  
a = [1, 2, 3, 4]  
print(a)  
print(type(a))  
</ código>
```

Tiene una salida como esta:

```
<cotización en bloque>>.  
[1, 2, 3, 4]  
clase 'list' (lista)  
</blockquote>
```

Puede insertarlo o actualizarlo según sus necesidades. Suponga que desea añadir la "tupla x" en esta lista y que también desea insertar la "tupla x" al principio. Así que el código completo se ve así:

```
/usr/bin/python3 #  
tuple  
x = (1, 2, 3, 4)  
  
# lista  
a = [1, 2, 3, 4]  
  
# añadiendo la tupla x a la lista  
a.append(x)  
imprimir(a)  
  
# insertando tupla x en la primera posición  
a.insert(0, x)
```

```
imprimir(a)

# Now iterating the final list a
para mí en una:
    imprimir(i)
</código>
```

Y la salida es así:

```
<cotización en bloque>.
1, 2, 3, 4, (1, 2, 3, 4)] # después de añadir
[(1, 2, 3, 4), 1, 2, 3, 4, (1, 2, 3, 4)] # después
de
inserción
# Cuando iteramos la lista 'a' la salida se ve así
(1, 2, 3, 4)
1
2
3
4
(1, 2, 3, 4)
</blockquoteo>
```

En Python, una cadena es también de tipo secuencial y se puede iterar a través de ella.

Considere este código:

```
<código
strings = "Esto es una cadena."
para WeWillIterateThroughIt en cadenas:
    print(WeWillIterateThroughIt)
</código>
```

Y la salida es como de costumbre:

```
<cotización en bloque>.
T
h
i
s
```

```
i  
s  
  
a  
  
s  
t  
r  
i  
n  
g  
. .  
</blockquote>
```

Un string es un tipo secuencial. Considere este código:

```
< código  
  strings = "string"  
  print(strings[1:3]  
 )  
</ código>
```

Significa que la cuerda va así:

```
0 = s  
1 = t  
2 = r  
3 = i  
4 = n  
5 = g
```

Por lo tanto, **strings[1:3]** significa que la secuencia comienza desde la posición 1 y sube hasta la posición 3, excluyendo la <sup>3<sup>a</sup></sup> posición. Significa que se detiene en la <sup>segunda</sup> posición. Así que el resultado es el esperado:

```
< cotización en bloque >.  
tr  
</blockquote>
```

---

## Diccionario

Python tiene otro tipo de valores agregados muy fuertes: el diccionario. Es una clase, como siempre. Es más bien como una matriz asociativa o hash en otros idiomas.

Considere este código:

```
< código
#!/usr/bin/python3
EnglishDictionaries = {'bare': 'jejune',
'anger': 'dungeon', 'abuse': 'vituperate',
'howl': 'ululate'}
print(EnglishDictionaries)
# consiguiendo en una forma más legible para los
humanos
para las llaves en EnglishDictionaries:
    print(teclas, "=", EnglishDictionaries[teclas])
</ código>
```

Y la salida es:

```
<cotización en bloque>.
{'abuso': 'vituperate', 'bare': 'jejune', 'howl':
'ululate', 'anger': 'dungeon'}
abuse =
vituperate bare =
jejune howl =
ululate anger =
dungeon
</blockquote>
```

Ahora podemos ordenar este diccionario en un orden alfabético como este:

```
< código
EnglishDictionaries = {'bare': 'jejune',
'anger': 'dungeon', 'abuse': 'vituperate',
'howl': 'ululate'}
para las teclas ordenadas
(EnglishDictionaries.keys()): print(teclas,
"=", EnglishDictionaries[teclas])
</ código>
```

Y obtenemos un resultado limpio en orden alfabético:

```
<cotización en bloque>.
```

```
abuso = cólera
vituperada =
dudgeon bare =
aullido jejune =
ululate
</blockquote>
```

También podemos escribir el diccionario de otra manera usando una construcción del diccionario de la clase. Considere este código:

```
< código
sinónimos = dic(bare='jejune', anger='dudgeon',
abuse='vituperate', howl= 'ululate')
</ código>
```

Acabamos de cambiar el nombre de la variable pero usamos el mismo par de palabras. Ahora podemos ordenarlas como antes para obtener el mismo resultado. Recuerda una cosa: cuando usas la función dict(), no debes escribir las claves entre comillas, sino que los valores de las cadenas deben ser citados como yo lo hice. Dado que el diccionario es mutable, puede insertar pares de valores de clave en él, como listas.

---

## Objeto

Python es un lenguaje orientado a objetos. Lo discutiremos más adelante en detalle. Digamos que hay una clase o plano y de esta clase o plano podemos obtener muchos tipos de objetos. Por ejemplo, la clase humana. Es una clase muy compleja! Tiene muchos tipos de propiedades; muchos tipos de acciones son realizadas por esta clase. Cuando creamos un objeto o instancia de esta clase, este objeto o instancia puede llevar adelante cada uno de los rasgos de esta clase. Recuerda, siempre ha habido un ser humano bueno y un ser humano malo.

Supongamos que una clase humana tiene dos tipos de humanos: uno es bueno y el otro es malo. En realidad, no es tan simple. Pero para empezar con nuestro aprendizaje, empezamos con un escenario menos complejo.

Considere el siguiente código:

```
/usr/bin/python3
clase Humano:
    def init (self, kind = "Bueno") :
        self.kind = kind
```

```
def whatKind(self):
    devuelve
    self.kind

def main():
    BuenaHumana = Humana()
    imprimir(BuenaHumana.whatKind())
    MalaHumana = Humana("Mala")
    imprimir(MalaHumana.whatKind())
si nombre __ == " principal ":
    main()
</código>
```

Y aquí está el resultado:

```
<cotización en bloque>.
Bueno
Malo
</blockquote>
```

En el código anterior el objeto es la instancia de una clase y encapsula cada propiedad y método de la clase o plano. En la clase anterior, asumimos una especie de plano donde cada ser humano es bueno. Así que en el método de inicialización, escribimos este código:

```
<código
clase Humana:
    def init (self, kind = "Bueno"):
        self.kind = kind
    def whatKind(self):
        devolver self.kind
</código>
```

Aquí, "yo" significa una referencia al objeto. Y el siguiente parámetro define el tipo de objetos humanos que queremos crear.

¿Qué significa esta línea?

```
<código
def whatKind(self):
    devolver self.kind
</código>
```

Devuelve el valor de qué tipo de objeto humano queremos crear. Los siguientes pasos se explican por sí solos:

```
< código  
def main():  
    Humano Bueno = Humano()  
    imprimir(BuenoHumano.whatKind())  
    MaloHumano = Humano("Malo")  
    imprimir(MaloHumano.whatKind())  
si nombre __ == " main " :  
    main()  
</ código>
```

Cuando creamos nuestro primer objeto "GoodHuman", no necesitamos pasar ningún valor tan "bueno" como el valor por defecto que ya ha sido pasado implícitamente a través del proceso de inicialización. Pero cuando queremos crear "BadHuman", necesitamos pasar el valor explícitamente y éste nos devuelve ese valor.

## 10. Condicionales

Sanjib Sinha 

(1) Howrah, Bengala Occidental, India

---

En Python hay dos tipos de condicionales. Son: ejecuciones condicionales y valores condicionales o expresiones condicionales. En las ejecuciones condicionales ejecutamos o comprobamos el estado de la sentencia. Sabemos que entre dos valores puede haber tres tipos de condiciones. Es menor o mayor que o es igual. Escribe este código:

```
< código
def
    conditionals_exec()
        : a, b = 1, 3
        si a < b:
            print("a es menos que
b") elif a > b:
            print ("a es mayor que b"):
            print("a es igual a b")
    conditionals_exec()

</ código>
```

La salida es:

```
#####
# a es menos que b
#####
```

El resultado es obvio. Ahora puede cambiar el valor y probar el código. Ahora trata de reescribir la declaración anterior de una manera diferente. Podemos decir que x es menor que y o mayor que y. De lo contrario, es obvio que son iguales.

```
< código
def
    valores_condicionale
    s(): a, b = 1, 2
    statements = "menos que" si a < b else " no
menos que.
        imprimir(estados de cuenta)
valores_condicionales()
</ código>
```

Estas funciones se pueden escribir de forma más cómoda y ordenada con las funciones main():

```
< código
def main():
    print("This is main function.")
    conditionals_exec()
    conditional_values()

def
    conditionals_exec()
    : a, b = 1, 3
    si a < b:
        print("a es menor que
b") elif a > b:
        print ("a es mayor que b"):
        print("a es igual a b")

def
    valores_condicionale
    s(): a, b = 1, 2
    statements = "menos que" si a < b else " no
menos que.
        imprimir(estados de cuenta)

    si nombre == " principal " : principal()
```

</codigo>

Si ejecutamos este programa ahora, la salida será:

```
#####
# Esta es la función
principal. # menos que
# a es menos que b
#####
```

Ahora podemos cambiar el lugar de conditional\_values(), y conditionals\_exec() y la salida cambiará en consecuencia:

```
#####
# Esta es la función
principal. # a es menos
que b
# menos que
#####
```

## 11. Lazos

Sanjib Sinha 

(1) Howrah, Bengala Occidental, India

---

"While loop" es la forma más simple de bucle en Python. Pero tienes que entenderlo bien. De lo contrario, puede acabar consumiendo tu memoria ejecutando el bucle infinito. Por lo general, la mayoría de los trabajos se realizan por "for loop". Pero en algunos casos especiales, es necesario usar "while loop". Un entendimiento básico es importante.

---

### Mientras Loops

En un lenguaje sencillo decimos a menudo: "Si bien es cierto, sigue corriendo. Aunque no es verdad, se detiene". Lógicamente, lo mismo sucede aquí. Mientras que una declaración es verdadera, el proceso continúa. Necesitas un mecanismo para detener ese proceso. Eso es importante. De lo contrario, esa declaración te consumirá la memoria.

Considere este código:

```
< código  
b = 1  
mientras que b < 50:  
    imprimir  
    (b) b =  
    b + 1  
</ código>
```

¿Qué significa esto? Esto significa que la afirmación "b < 50" es verdadera hasta que la suite o bloque de código es verdadera dentro de ella. Dentro del bloque escribimos "b = b + 1" y antes del inicio del bucle while definimos el valor de b como 1.

Así que en cada paso b progresó sumando 1 a su valor y terminó en 49. En

la salida obtendrá de 1 a 49.

Sigamos avanzando. Considere este código:

```
< código
#!/usr/bin/python3
# serie simple de fibonacci
# la suma de dos números define el siguiente
conjunto
a, b = 0, 1
mientras que b < 50:
    print(b, end='
')
    a, b = b, a +
b
</ código>
```

El resultado es bastante obvio:

```
< cotización en bloque >>.
1 1 2 3 5 8 13 21 34
</ blockquoteo >
```

Para los principiantes, escribamos este código de una manera más legible y dará un resultado totalmente diferente:

```
/usr/bin/python3 a,
b = 0, 1
mientras que b < 30:
    print(b, end='
')
    a = b
    b = a + b
</ código>
```

Vamos a explicar los pasos uno por uno para entenderlo correctamente.

El bucle comienza con 1. En el primer paso, el valor de "a" es 1. En el siguiente paso, el valor de "b" es 2. Ahora el valor de "a" es 2, por lo que el valor de "b" es 4. Ahora el valor de "a" es 4, por lo que el valor de "b" es 8 ( $4+4$ ). Ahora el valor de "a" es 8 de modo que el valor de "b" es  $(8 + 8) = 16$ . Ahora el valor de "a" es 16. ¿Cuál será el valor de b? Será  $16 + 16 = 32$ . Pero 32 es mayor que 30. Así que saldrá de la suite de código del bucle while.

La salida del código anterior será:

```
<cotización en bloque>.
```

```
1 2 4 8 16
</blockquote>
```

Escribamos todo el código en un nuevo formato:

```
<código
#!/usr/bin/python3
# serie simple de fibonacci
# la suma de dos números define el siguiente
conjunto
a, b = 0, 1
mientras que b < 30:
    print("a = " , a, "=" , "b = " , b, ",," , end='
    ')
        a, b = b, a + b
    imprimir
    ("*****")
a, b = 0, 1
mientras que b < 30:
    print("a = " , a, "=" , "b = " , b, ",," , end='
    ')
        a = b
        b = a + b
</código>
```

Y la salida será:

```
<cotización en bloque>.
a=      0 y b=      1 , a=      1 y b=      1, a=      1 y a
1
b=      2 , a=      2 y b=      3, a=      3 y b=      5, a
=  5 y b=      8 , a=      8 y b=     13, a=     13 y b
= 21 ,
***** Líneas de separación.
a=  0 y b=  1 , a=  1 y b=  2, a=  2 y b=
        4, a=  4 y b=  8, a=  8 y b=  16 ,
</blockquote>
```

Ahora, con suerte, esto explica cómo funcionan los bucles while.

---

## Para bucles

El bucle más común utilizado en Python es el bucle. De hecho, esencialmente casi todos los tipos de trabajos de looping se pueden hacer a través del bucle "for".

Hay una razón, por supuesto. Con la ayuda de for loop, podemos iterar a través de objetos Python y podemos iterar a través de la mayoría de los objetos Python. Veamos un ejemplo:

```
<code>
#!/usr/bin/python3
canciones =
open('file.txt')
para líneas en
    canciones.read():
        print(lines, end=' ')
</código>
```

Y el resultado de la canción es así:

```
<cotización en bloque>.
Oye, chica, me has tocado
fuerte, tu soledad me ha hecho
llorar, soy un nerd estúpido.
Pensé en las palabras, no pude mantenerlas,
así que lloré.
Un estúpido nerd
</blockquoteo>
```

Tenemos una canción escrita en un archivo llamado "file.txt" y la iteramos a través de este archivo. Podríamos iterar línea por línea a medida que se indexan. Considere este código donde acabamos de usar la función "enumerate()" y el **valor del índice**:

```
<código
# enumerar
canciones = open('archivo.txt')
para el índice, las líneas en
    enumerate(songs.readlines()): print(index,
        lines, end=' ')
</código>
```

Y la salida es así:

```
<cotización en bloque>.
0 Oye, chica, me tocaste con fuerza.
1 tu soledad me ha hecho llorar
2 Soy un nerd estúpido.
3 Pensé en las palabras, no pude mantenerme
4 Así que lloro
5 Un estúpido nerd
</blockquote>
```

Ahora bien, ¿qué significa esta función "enumerar()"? El diccionario dice: la enumeración es un tipo de numeración que es una lista numerada. Consideremos esta línea de código:

```
< código
strings = "Esto es una cadena."
# ahora vamos a encontrar cuántas 's' hay dentro de
esta cadena
para el índice, s en
    enumerate(strings): if s == 's':
        print("Hola soy 's' y estoy ubicado en la
posición {}".format(index))
</ código>
```

Y tenemos una salida:

```
<cotización en bloque>.
Hola soy 's' y estoy ubicado en la posición
3 Hola soy 's' y estoy ubicado en la
posición 6 Hola soy 's' y estoy ubicado en
la posición 10
</blockquote>
```

Esto es extremadamente útil. Puede buscar cualquier carácter dentro de cualquier cadena. En Python, las funciones o subrutinas son extremadamente importantes para la reutilización de códigos. Podemos llamar a una función varias veces y pasar muchos argumentos o parámetros para obtener diferentes efectos. Ahora vamos a pasar un parámetro dentro de la función loops(). Considere este código a continuación:

```
< código
#!/usr/bin/python3
```

```

def main():
    loops(0)
    loops()
    loops(3)

def loops(a = 4):
    para i in range(a,
                     6): print(i, " ")
    Imprimir ("*****")
si nombre __ == " principal ":
    main()
</código>

```

¿Qué significa este código? En la función loops(), hemos pasado un parámetro a y le hemos asignado un valor 4. Es el valor por defecto. Para que en el futuro si nos olvidamos de pasar cualquier argumento el código no se rompa.

Hemos llamado a esa función tres veces dentro de la función main(), pero con tres valores diferentes , y uno de ellos es NULL. Es decir, no hemos aprobado ningún argumento.

La salida cambia con el nuevo código:

```

<cotización en bloque>.
0
1
2
3
4
5
*****
4
5
*****
3
4
5
*****
</blockquote>

```

Ahora es obvio que puedes jugar con este código. Puede pasar dos argumentos dentro de la función loops() y controlar la función range() para obtener valores diferentes.

## 12. Expresiones regulares

Sanjib Sinha 

(1) Howrah, Bengala Occidental, India

---

Buscar y reemplazar con expresiones regulares es igualmente fácil y muy simple en su naturaleza. Para hacer eso vamos a ajustar nuestro viejo código un poco. Utilizamos el módulo "re" y hace los trabajos más sencillos. La expresión regular es en sí misma un gran tema. Tratamos de entender las cosas básicas para poder utilizarlas en nuestros proyectos futuros.

---

### Usando el Módulo "re"

Si desea utilizar el módulo "re", el primer paso es la importación. Primero tenemos que importar el módulo y escribirlo en la parte superior del código. Consideremos este código donde tenemos un archivo de texto llamado "file.txt" y que está almacenado en nuestra carpeta "primaria".

```
/usr/bin/python3
import re
def main():
    ReplaceWord()
    DEmarcationLine()
    MatchAndReplaceWord()

def
    ReemplazarPal
    abra():
        intentar:
            files = open(".../primary/file.txt")
            para la linea en archivos:
                # Puedes buscar cualquier palabra
                print(re.sub('lenor|more', #####',
```

```

), end=' ')
    except FileNotFoundError como e:
        print("File was not found:", e)

def
    MatchAndReplaceWord()
: intentar:
    files = open("../primary/file.txt")
    para la línea en archivos:
        # puedes buscar cualquier patrón que
coincida y luego reemplazarlo con esta palabra
        match = re.search('(len|neverm)ore',
línea
)           si coinciden:
                print(line.replace(match.group(), '#'))
            except FileNotFoundError como e:
                print("File was not found:", e)

def DEmarcationLine():
    print("*****")

```

si nombre \_\_ == " principal ":

main()

</código>

Antes de que tengamos la salida, veamos lo que está escrito dentro del archivo. El "file.txt" en la carpeta "primary" tiene estas líneas:

```

<cotización en bloque>.
primera línea lenore
es nueve, segunda línea y cenar
tercera línea y nunca más sobre
y cuarto
quinta línea de pino
lenore y el temblor
aquí hay más
línea y una nueva
línea
Te amo
donde te estás quedando ahora?

```

```
no lo sé
</blockquote>
```

Como puede ver, estas no son oraciones muy significativas. Nuestra principal preocupación es muy simple. Escribimos algunas líneas sin sentido y luego intentamos trabajar en ello con el uso del módulo "re". Ahora ejecutamos el código y aquí está el resultado:

```
<cotización en bloque>.
primera línea #####e
es nueve, segunda línea y cenar
tercera línea y nunca más y
cuarto.
quinta línea de pino y
el temblor
aquí está la línea
##### y una
nueva línea
Te amo
¿Dónde te estás quedando
ahora? No lo sé.
primera línea #####
tercera línea y #####
sobre la quinta línea de
pino #####
</blockquote>
```

Todas las palabras "lenore" y "nevermore" han sido sustituidas por cinco hashtags: "“#####”". Utilizamos dos métodos de módulo "re" que importamos y escribimos en la parte superior del código. Estos métodos son "re.sub()" y "line.replace()". Hemos suministrado la vieja cadena y la nueva palabra. Hemos dado cinco hashtags, pero usted podría haber dado cualquier otra palabra, por supuesto.

---

## Reutilización con expresiones regulares

Ya has visto cómo podemos buscar y reemplazar palabras en un archivo con la ayuda de la expresión regular. Ahora intentaremos reutilizar el código para poder utilizarlo una y otra vez. Además, también intentaremos escribirlos de una manera más legible.

Primero escribamos los pasos. Lo que queremos conseguir es muy

importante. Tengamos una idea clara primero y la mejor manera es escribirla.

- Necesitamos abrir un archivo y ponerlo en el "bloque de prueba" para evitar que aparezca un mensaje de error desagradable. Los principiantes pueden encontrar este "bloque de prueba" bastante intimidante. No lo he explicado antes y de repente empecé a usarlo. Lo he hecho intencionadamente. Se explica en el siguiente capítulo, "Excepciones, errores de detección". Pero antes de eso, quiero que las escribas y te acostumbres a un concepto que parece complejo. Una vez que aprenda este "intento de bloqueo", por favor, vuelva a visitar este código de nuevo. Le resultará muy fácil! Además, a medida que progresas, te darás cuenta de que el uso de "try block" es siempre un buen hábito.
- Obtener el patrón de las palabras que queremos buscar y, usando banderas, podemos ignorar mayúsculas y minúsculas.
- Utilice el método de búsqueda del módulo "re" para ver si ese patrón coincide con nuestra línea.
- Ahora, si coincide, reemplácelo con nuevas palabras.

Consideré este código a continuación y lea los comentarios. En los comentarios explico brevemente lo que voy a hacer.

```
/usr/bin/python3
import re

def main():
    CompilerAndReplaceWord()

def
    CompilerAndReplaceWord()
    : intentar:
        files = open(".../primary/file.txt")
        # puedes buscar cualquier patrón que
coincida ignorando las mayúsculas o minúsculas
        pattern = re.compile('(len|never)ore',
re.IGNORECASE)
        para la línea en los archivos:
            # re module search that pattern in a
renglón
```

```

        if re.search(patrón, línea):
            # Encontramos ese patrón y ahora
            es el momento de reemplazarlo con una nueva cuerda.
            print(pattern.sub("#####", 
                línea),
end=' ')
except FileNotFoundError como e:
    print("File was not found:", e)

si nombre __ == " principal ":
    main()
</código>

```

Y en la salida reemplaza todas las palabras "lenore" y "nevermore" por seis hashtags. Para ello, también comprueba las mayúsculas y minúsculas y finalmente las sustituye todas.

```

<cotización en bloque>.
primera línea #####
tercera línea y #####
sobre la quinta línea de
pino
No lo sé #####
</blockquote>

```

---

## Búsqueda con expresiones regulares

Las expresiones regulares son un método muy poderoso para emparejar patrones. La expresión regular es un lenguaje pequeño en sí mismo y puede ser muy simple y muy complejo.

Se implementa en Python con el módulo "re".

Considere este código:

```

/usr/bin/python3
import re
def main():
    FindWord()
    DEmarcationLine()
    MatchWord()
def FindWord():

```

**Inténtalo:**

```
files = open("../primary/file.txt")
para la línea en archivos:
    # Puedes buscar cualquier palabra
    if re.search('lenor|more', line):
        print(line, end=' ')
excepto FileNotFoundError como e:
    print("Fiel was not found:", e)

def
    MatchWord()
    : intentar:
        files = open("../primary/file.txt")
        para la línea en archivos:
            # Puedes buscar cualquier patrón que
coincida con esta palabra
            match = re.search('(len|neverm)ore',
línea
)
            si coinciden:
                print(match.group())
excepto FileNotFoundError como e:
    print("Fiel no fue encontrado:", e)

def DEmarcationLine():
    print("*****")
    si nombre _ == " principal ":
        main()
</código>
```

Aquí buscamos un archivo llamado "file.txt" que tiene palabras como "lenor" o "more" y que también coincide con algunas palabras que terminan en "ore". Hemos definido dos funciones para buscarlo y hemos utilizado el módulo "re".

Veamos primero cuál es el contenido dentro de "file.txt". Hay algunas palabras y líneas engañosas sólo para probar nuestra búsqueda.

```
<cotización en bloque>.
primera línea lenore
es nueve, segunda línea y cenar
tercera línea y nunca más sobre
y cuarto
quinta línea de pino lenore
```

```
y el temblor
aquí es más
línea y una
nueva línea
Te amo
¿Dónde te estás quedando
ahora? No lo sé.
</blockquote>
```

Después de ejecutar nuestro código hemos encontrado este resultado de búsqueda.

```
<cotización en bloque>.
primera línea lenore
tercera línea y nunca más sobre
la quinta línea de pino lenore
*****  

Lenore, nunca
más Lenore,
nunca más
Lenore.
</blockquote>
```

Es un ejemplo de expresión regular muy simple. Está más allá de nuestro alcance enseñar aquí la expresión regular, pero al menos podemos tener alguna idea. Le recomiendo encarecidamente que siga adelante. Búsqueda de "expresión regular" en Internet . Encontrarás muchos tutoriales. Aprender y entender la expresión regular es muy importante. Ya sea que te conviertas en un desarrollador web, un hacker ético o un programador Python, la expresión regular te ayudará.

## 13. Excepciones, errores de detección

Sanjib Sinha 

(1) Howrah, Bengala Occidental, India

---

Espero que ya hayas escrito muchos códigos. Si realmente hubieras hecho eso, habrías encontrado uno o dos errores. Hay dos tipos distinguibles de errores. El primero es "SyntaxError". Significa que tienes un error en la sintaxis. Considera este código:

```
< código
>>> for i in range(10) print(i)
SyntaxError: invalid syntax
</ código>
```

Como ves, olvidé usar ":" en el bucle. Es un error de sintaxis.

Otro error es "Excepciones". Significa que escribes un código perfectamente. No hay errores sintácticos. Pero te olvidas de definir una variable. Consideremos estas líneas de código:

```
< código
>>> 10 * x
Traceback (última llamada):
  Archivo "<pyshell#1>", línea 1, en <módulo>
    10 * x
NameError: nombre 'x' no está definido
>>> 10 / 0
Traceback (última llamada):
  Archivo "<pyshell#2>", línea 1, en
    <módulo> 10 / 0
```

```
ZeroDivisionError: división por cero
>>> '2' + 2
Traceback (última llamada):
  Archivo "<pyshell#3>", línea 1, en
    <módulo> '2' + 2
  TipoError: No puede convertir un objeto'int' a str
implícitamente
>>> inputs = input("Por favor, introduzca un
número.") Por favor, introduzca un número.
>>> Entradas + 2
Traceback (última llamada):
  Archivo "<pyshell#5>", línea 1, en
    <módulo> entradas + 2
  TipoError: No puede convertir un objeto'int' a str
implícitamente
>>> inputs = input("Por favor, introduzca un
número.") Por favor, introduzca un número.12
>>> Entradas - 10
Traceback (última llamada):
  Archivo "<pyshell#7>", línea 1, en
    <módulo> entradas - 10
  TypeError: tipo(s) de operando no soportado(s) para
-: 'str' y 'int'.
>>> int(entradas) -
10 2
>>>
</código>
```

Como puede ver, hay muchos tipos diferentes de errores. Y en la última línea hemos salido del error y obtenido una salida perfecta. En el último error obtenemos un "TypeError". Intentamos restar un número entero de un objeto de cadena. En el último paso convertimos ese entero de entrada de cadena y la sustracción se realizó sin problemas.

Siempre es bueno detectar esos errores y obtener un buen resultado. La frase "probar bloque" ya se ha utilizado anteriormente. Ahora llega el momento en que aprendemos a usar esos bloques para detectar errores. Escriba el código abajo en su editor de texto y guárdelo como "CatchError.py".

<código

```

#!/usr/bin/python3
def main():
    FileRead()
    DemarcationLine()
    LineStrip()
    DemarcationLine()
    CheckFileExtension()
def ReadFile(filename):
    files = open(filename)
    lines =
    files.readlines()
    para el índice, línea en enumerar(líneas):
        print(índice, "=", línea)
def
    StripFile(nombredelar
hivo): archivos =
    open(nombredelarchivo)
    para líneas en
    ficheros:print(lines.strip()) def
RaisingError(filename):
    si el nombre del archivo.termina con(".txt"):
        lines = open(nombre de archivo)
        para line in lines:print(line.strip())
    si no:
        aumentar ValueError("File must end
with.txt") def FileRead():
    Inténtalo:
        ReadFile("../primary/files.txt") # la ruta
está bien, lee el archivo
    excepto IOError como e:
        print("No se pudo abrir el
archivo:", e) def LineStrip():
    Inténtalo:
        StripFile("primary/files.txt")
    excepto IOError como e:
        print("No se pudo abrir el archivo:", e) #
se abrirá
dar error
def
    CheckFileExtension()
    : intentar:

```

```
RaisingError(".../primary/file.rtf"
) excepto IOError como e:
```

```

        print("No se pudo abrir el
archivo:", e) excepto ValueError como
e:
        print("Bad Filename:",
e) def main():
    Imprimir ("*****")
si nombre _ == " principal ":
    main()
</código>

```

Ejecute este archivo y obtendrá esta salida:

```

<cotización en bloque>.
No se pudo abrir el archivo: Errno 2] No existe tal
archivo o directorio: '../primary/files.txt'.
*****
No se pudo abrir el archivo: Errno 2] No existe tal
archivo o directorio: 'primary/files.txt'.
*****
Bad Filename: El archivo debe terminar en.txt
</blockquoteo>

```

Como ejercicio, intenta escribir este código con "Try" y "Excepto" y captura si hay algún error.

```

/usr/bin/python3
def main():
    GetARangeOfNumber()
def
GetARangeOfNumber():
    para el índice en IteratingStepByStep(1,123,
7):
        print(index, end=' ')
def IteratingStepByStep(start, stop, step):
    number = start
    while number <=
        stop: yield
        number number +=
        step
    si nombre _ == " principal ":
        main()
</código>

```

## 14. Funciones

Sanjib Sinha 

(1) Howrah, Bengala Occidental, India

---

Primero definamos la función e intentemos saber por qué se está usando la función en Python. Considere este código:

```
/usr/bin/python3
def main():
    print("Esta es la función principal.")
si nombre __ == " principal__":
    main()
</código>
```

Y la salida es:

```
<cotización en bloque>.
Esta es la función principal.
</blockquote>
```

¿Qué significa eso? En primer lugar, vamos a entender lo que significa la función. Una función se utiliza en cualquier lenguaje de programación para reutilizar el código. Los programadores son perezosos y no quieren escribir una y otra vez. Y no es una buena idea escribir lo mismo una y otra vez. Así que el concepto de reutilización entra en juego y usamos la función para hacerlo.

Usted puede considerar un ejemplo muy simple. Supongamos que queremos usar una línea de demarcación una y otra vez. ¿Escribirás así una y otra vez?

```
< código  
Imprimir ("*****")  
</ código>
```

¿O escribirás una función y la llamarás cuando sea necesario? Así:

```
< código  
def DemarcationLine ():  
    print("*****")  
") DemarcationLine () :  
DemarcationLine () :  
DemarcationLine () :  
< código>
```

Cada vez que llame a la función "DemarcationLine()" imprimirá una línea de demarcación.

Pasemos ahora a la primera pregunta. Siempre es una buena práctica escribir funciones dentro de la función main() y puede llamarlas en cualquier momento. El control de flujo no necesariamente sigue hacia abajo. Puedes probarlo:

```
< código  
def OtraFunción ():  
    print("I am another function.")  
def TestFunction ():  
    print("Voy a llamar a otra función.")  
    OtraFunción()  
TestFunction()  
< código>
```

Se imprimirá sin ningún problema y le dará esta salida:

```
< cotización en bloque >.  
Voy a llamar a otra función. Yo soy  
la función de otro.  
</ blockquote >
```

Ahora escribiremos el código de arriba de forma diferente.

```
< código
```

```
def TestFunction():
    print("I am going to call another function.")
    AnotherFunction()
```

### Función de prueba()

```
def OtraFunción():
    print("Soy otra función.")
</código>
```

Un poco de cambio de posición. No hemos definido `AnotherFunction()` antes de `TestFunction()` y por eso, dará una salida de error:

```
<cotización en bloque>.
Voy a llamar a otra función. Traceback
(última llamada):
  Archivo
"/home/hagudu/PycharmProjects/FirstPythonProject/functions/defining_functions.py", línea 17, en <module>
    Función de prueba()
  Archivo
"/home/hagudu/PycharmProjects/FirstPythonProject/functions/defining_functions.py", línea 15, en TestFunction
    OtraFunción()
NameError: nombre 'AnotherFunction' no está
definido
</blockquote>
```

Por lo tanto, cada vez que se llama una función dentro de otra función, es necesario definirla primero. Pero este problema puede ser resuelto si se define primero la función `main()`. Ahora considera este código:

```
/usr/bin/python3
def main():
    TestFunction()
def
TestFunction():
    print("Voy a llamar a otra función.")
    OtraFunción()
```

```
def OtraFunción():
    print("Soy otra función.")
si nombre _ == " principal ":
    main()
</código>
```

Y aquí está el resultado:

```
<cotización en bloque>.
Voy a llamar a otra función. Soy
otra función.
</blockquote>
```

Ahora veamos, no nos preocupamos por la posición porque todas las funciones están bajo la función main(). Ahora se está añadiendo mucha más flexibilidad cuando se está usando la función main() de esta manera. Otra gran ventaja de usar la función es pasar parámetros o argumentos a través de ella.

```
/usr/bin/python3
def main():
    Parámetros de paso(1,2,3)
    def PassingParameters(argument1, argument2,
argument3):
        print("Aquí están nuestros argumentos:",
argument1, argument2, argument3)
        si nombre _ == " principal ":
            main()
</código>
```

Y la salida es:

```
<cotización en bloque>.
Aquí están nuestros argumentos: 1 2 3
</blockquote>
```

Hemos pasado tres parámetros o argumentos y obtenemos el resultado esperado. ¿Pero qué pasa si nos olvidamos de pasar cualquier argumento? No queremos recibir ningún mensaje de error desagradable. Podemos lograrlo de dos maneras:

```
/usr/bin/python3
def main():
    Parámetros de paso(1)
    def PassingParameters(argumento1, argumento2 = 4,
argumento3 = 6):
        print("Aquí están nuestros argumentos:",
argumento1, argumento2, argumento3)
    si nombre __ == " principal ":
        main()
</código>
```

Y la salida:

```
<cotización en bloque>.
Aquí están nuestros argumentos: 1 4 6
</blockquote>
```

Se llama pasar los valores por defecto. Hemos pasado dos valores por defecto y cuando llamamos a la función, ésta toma ese valor por defecto. Ahora podemos anular estos valores por defecto en cualquier momento. Considere esto:

```
/usr/bin/python3
def main():
    Parámetros de paso(1, 10, 14)
    def PassingParameters(argumento1, argumento2 = 4,
argumento3 = 6):
        print("Aquí están nuestros argumentos:",
argumento1, argumento2, argumento3)
    si nombre __ == " principal ":
        main()
</código>
```

Y la salida:

```
<cotización en bloque>.
Aquí están nuestros argumentos: 1 10 14
</blockquote>
```

Hemos sobrescrito los valores por defecto pasando nuevos valores y la salida ha cambiado en consecuencia. Podemos escribir este código de esta manera también:

```
/usr/bin/python3
def main():
    Parámetros de paso(1)
    def PassingParameters(argumento1, argumento2 =
Ninguno, argumento3 = 6):
        if argument2 == Ninguno:
            print("Aquí están nuestros argumentos:",
argumento1, argumento2, argumento3)
        de los demás:
            print("Aquí están nuestros argumentos:",
argumento1, argumento2, argumento3)
        si nombre __ == " principal ":
            main()
</código>
```

Y la salida:

```
<cotización en bloque>.
Aquí están nuestros argumentos: 1 Ninguno 6
</blockquoteo>
```

¿Qué pasa si pasamos un nuevo valor para el argumento2? Considere este código:

```
/usr/bin/python3
def main():
    Parámetros de paso(1, 12)
    def PassingParameters(argumento1, argumento2 =
Ninguno, argumento3 = 6):
        if argument2 == Ninguno:
            print("Aquí están nuestros argumentos:",
argumento1, argumento2, argumento3)
        de los demás:
            print("Aquí están nuestros argumentos:",
argumento1, argumento2, argumento3)
```

```
si nombre __ == " principal ":
    main()
</código>
```

Y la salida:

```
<cotización en bloque>.
Aquí están nuestros argumentos: 1 12 6
</blockquote>
```

En la siguiente sección veremos cómo funcionan las listas de argumentos en una función.

---

## Valores de retorno

En Python una función puede devolver cualquier valor. Puede devolver cualquier tipo de datos: cadena, entero, objeto-cualquier cosa. Devolvamos un objeto.

Considere este código:

```
/usr/bin/python3
def main():
    para el índice en ReturnValues():
        print(index, end=" ")
def ReturnValues():
    #return "Returning
    string." #return 56
    rango de retorno(10)
si nombre __ == " principal ":
    main()
</código>
```

Y la salida:

```
<cotización en bloque>.
0 1 2 3 4 5 6 7 8 9
</blockquote>
```

Hemos devuelto el objeto range() y hemos obtenido el valor en nuestra función main().

## Generar funciones

En Python podemos generar funciones. Vamos a explicarlo paso a paso.

Consider this code first:

```
/usr/bin/python3
def main():
    RangeFunctions()
def
RangeFunctions():
    para i in range(10):
        print(i, end=' ')
si nombre __ == " principal ":
    main()
</código>
```

Y el resultado es bastante obvio:

```
<cotización en bloque>.
0 1 2 3 4 5 6 7 8 9
</blockquote>
```

Probablemente ha encontrado que la función RangeFunctions() tiene una limitación. Se detiene en 9, aunque el rango se menciona como 10. ¿Qué puedo hacer para incluir este número?

Escribamos RangeFunctions() de la siguiente manera:

```
/usr/bin/python3
def main():
    para el indice en RangeFunctions(0, 10, 1):
        print(index, end=' ')
    print()

def RangeFunctions(start, stop,
                  step): i = start
    while i <=
        stop: yield
        i
        i += paso
si nombre == " principal ":
```

```
main()
</código>
```

Y aquí está el resultado:

```
<cotización en bloque>>.
0 1 2 3 4 5 6 7 8 9 10
</blockquote>
```

Aquí hemos utilizado la palabra clave "yield". Se hace porque hemos imaginado que el código progresará paso a paso como si estuviéramos reproduciendo una cinta. Después de dar un paso, se detendrá y comenzará desde allí y de nuevo comenzará y dará un paso. Puedes empezar desde cualquier punto o detenerte en cualquier punto y avanzar por cualquier paso.

Si escribimos así:

```
para el índice en RangeFunctions(15, 1025, 102):
    print(index, end=' ')
```

La salida será:

```
15 117 219 321 423 525 627 729 831 933.
```

Como has visto, podemos establecer el valor de cualquier argumento por defecto. Así que podemos escribir esta función así:

```
<código
def AnotherRangeFunctions(inicio = 0, parada,
paso = 1):
    i = inicio
    while i <=
        stop: yield
        i
        i += paso
</código>
```

Y podemos tratar de obtener la salida por :

```
<código
para el índice en AnotherRangeFunctions(25):
    print(index, end=' ')
```

```
</código>
```

Pero nos da un mensaje de error:

```
Archivo  
"/home/hagudu/PycharmProjects/FirstPythonProject/functions/generate-functions.py", línea 18  
    def AnotherRangeFunctions(inicio = 0, parada,  
    paso  
= 1):  
    SyntaxError: el argumento non-default sigue al  
argumento default
```

Python no apoya esto. ¿Podemos resolver este problema para que podamos pasar cualquier número de argumentos y controlarlo sin tener ningún mensaje de error?

Considere este código:

```
<código  
def AnotherRangeFunctions(*args):  
    numberArguments =  
        len(args)  
  
    si numberArguments < 1: aumentar TypeError('Se  
requiere al menos un argumento.')  
    elif numberArguments == 1:  
        stop =  
            args[0] start  
            = 0  
        paso = 1  
    elif numberArguments == 2:  
        # start and stop will be tuple  
        (start, stop) =  
            args step = 1  
    elif numberArguments == 3:  
        # todo comienza y se detiene y el paso será  
        doble  
        (inicio, parada, paso) = args  
  
    i = inicio  
    while i <=  
        stop: yield
```

```
i  
i += paso  
</código>
```

Anote cada línea y tome notas una al lado de la otra. Añada comentarios en los que considere que es necesaria una explicación.

---

## Listas de argumentos

En Python a veces se necesita un número arbitrario de argumentos y hay que nombrarlos. Escribamos este código:

```
/usr/bin/python3
def main():
    Listas de aprobación de los documentos (1, 2,
        3, 5, 7, 45, 98.)
    56, 4356, 90876543)
        Aprobación de otras listas de argumentos(1, 2,
            3, 5, 7.)
    45, 98, 76, 987654, 3245, 2345, 98760)

    def PassingListsOfArguments(arg1, arg2, arg3, arg4,
*args):
        print(arg1, arg2, arg3, arg4, args)

    def PassingAnotherListOfArguments(param1, param2,
*params):
        print(param1,
        param2) para el
        índice en
        parámetros:
            si índice ==
                76: x = 10
                y = índice + x
                print("Vamos a sumar 10 con", index, "y"
el nuevo valor es:", y)
                    continuar
                print(index, end=' ')
            si nombre __ == " principal ":
                main()
</código>
```

Y la salida es así:

<cotización en bloque>>.

```
1 2 3 5 (7, 45, 98, 56, 4356, 90876543)
1 2 3 3 5 7 7 45 98 Vamos a sumar 10 con 76 y el
nuevo valor es: 86
987654 3245 2345 98760
</blockquote>
```

En nuestro código, **\*args** o **\*params** significan listas de argumentos. Puede pasar cualquier número de argumentos a través de ellos. En código

```
def PassingListsOfArguments(arg1, arg2, arg3,
arg4, *args):
```

significa que necesitas pasar cuatro argumentos primero. Esto es obligatorio. Después de eso, el número de argumentos puede variar. Pero el número arbitrario de argumentos sale como "tupla". Ver la salida de esta función:

```
1 2 3 5 (7, 45, 98, 56, 4356, 90876543)
```

La última parte es obviamente una tupla y se puede iterar a través de ella.

---

## Argumentos con nombre

A veces es importante usar argumentos con nombre en Python. Y obtenemos esos argumentos nombrados en un formato de diccionario.

Considere este código:

```
/usr/bin/python3
def main():
    NamedArguments(nombre = 'Sanjib', dirección =
'Pluto', hobby = "Gardening")
    def NamedArguments(**kwargs):
        para la llave en Kwargs:
            print(clave, "=", kwargs[clave])
    si nombre _ == " principal ":
        main()
</código>
```

Y la salida:

```
<Nombre de  
jardinería =  
Dirección de  
Sanjib = Plutón  
</blockquote>
```

Al ser una salida de diccionario, no está ordenada. Puedes ordenarlo alfabéticamente.

Consideremos un código bastante largo en el que podamos utilizar cualquier tipo de argumento pasajero.

```
/usr/bin/python3  
def main():  
    NamedArguments(nombre = 'Sanjib', dirección =  
'Pluto', hobby = "Gardening")  
    DemarcationLine()  
    OtrosArgumentosNombrados('Hola', 1235, 1, 2, 3,  
    uno =  
1, dos = 2, tres = 3)  
  
    def  
        NamedArguments(**kwargs)  
        : para key in kwargs:  
            print(clave, "=", kwargs[clave])  
  
        def OtrosArgumentosNombrados(arg1, arg2, *args,  
**kwargs):  
            print(arg1, arg2)  
            para el índice en  
            args:  
                print(index, end=''  
) DemarcationLine()  
            para las llaves en Kwargs:  
                print(llaves, "=", kwargs[llaves])  
  
    def  
        DemarcationLine()  
        :  
        print("*****")  
    )  
  
    si nombre __ main()  
    </código>
```

```
== "
principal ":"
```

Aquí está el resultado:

```
<Dirección de  
jardinería =  
Nombre de Plutón  
= Sanjib  
*****  
Hola,  
1235.  
1 2 3 *****  
tres = 3  
dos = 2  
uno = 1  
</blockquote>
```

## 15. Clases

Sanjib Sinha 

(1) Howrah, Bengala Occidental, India

---

Si usted es un completo principiante, probablemente esté escuchando por primera vez acerca de la "programación y clase orientada a objetos". Demos una breve introducción a la programación orientada a objetos (OOP).

---

### Metodología orientada a objetos

Se basa en la programación del mundo real. Un objeto es una representación de una entidad del mundo real. Si hay un objeto, debe haber una clase o un plano detrás de él. En esa clase, el comportamiento de ese objeto se diseña o describe en detalle. Estos detalles consisten en todas las propiedades y acciones que realiza el objeto. Podría haber muchos tipos de objetos provenientes de diferentes clases y podrían tener relaciones. Puede ser muy complicado, pero siempre se pueden romper esos objetos entre sí y hacer algunos cambios. La ventaja de la orientación por objetos es que cuando se trabaja en una parte de un proyecto grande y complicado, la otra parte no se ve afectada. Nuestra meta es simple. Queremos unir diferentes objetos para crear un software grande y complicado. Al mismo tiempo, queremos que las relaciones de esos objetos sean lo más sueltas posible.

Un objeto de coche se construye de muchos otros objetos como neumáticos, ruedas, motor, acelerador, etcétera. Si se pincha un neumático, ¿se para el motor? Están interrelacionadas y dependen unas de otras. Pero finalmente se puede trabajar en ellos individualmente sin afectar al otro. Eso es orientación a objetos.

Considere un objeto, "GoodHuman". Este objeto debe ser diferente de otro objeto, "BadHuman". Ambos provienen de la clase "Humana". Ahora bien, estos dos objetos podrían tener interrelaciones e interacciones de datos. ¿Puedes imaginarte cuántos tipos de propiedades y métodos hay en el "Humano"?

clase? Podría ser muy complejo. Imagina una situación en la que un "BadHuman" hace algo feo. Al mismo tiempo, un "buen humano" hace algo bueno.

Quien hace lo que sea, la vida sigue adelante y eso también es orientación a objetos.

---

## La base de la orientación de objetos

La orientación a objetos es un tipo de metodología utilizada para construir aplicaciones de software. Un programa orientado a objetos consiste en clases, objetos y métodos. La metodología orientada a objetos en el desarrollo de software gira en torno a un único concepto llamado objeto... Puede desarrollar software dividiendo la aplicación en objetos de componentes. Estos objetos interactúan entre sí cuando se junta toda la aplicación. Un objeto es una combinación de mensajes y datos. El objeto recibe y envía mensajes y estos mensajes contienen datos o información.

Usted (un objeto) interactúa con su televisor (otro objeto) a través de mensajes enviados a través de un mando a distancia (otro objeto).

Considere otro ejemplo real de un balón de fútbol. Un balón de fútbol tiene un límite. Tiene una propiedad específica definida como el rebote. Puedes dirigir o aplicar pocas acciones específicas pateándolo o tirándolo.

Un objeto tiene un estado. Puede mostrar comportamiento. Tiene un ID único.

La diferencia entre un objeto y una clase es sutil pero importante. Mientras que una clase es un concepto abstracto, un objeto es una entidad concreta. Desde una clase, se pueden crear o instanciar objetos con propiedades específicas. Es por eso que un objeto se llama a menudo una instancia de una clase.

Una de las principales características de la programación orientada a objetos es el "polimorfismo". El polimorfismo es la capacidad de algo de asumir diferentes formas. En la programación orientada a objetos, el polimorfismo es la propiedad de que un mensaje puede significar cosas diferentes dependiendo de los objetos que lo reciben. El mensaje "Acelerar" significa una cosa si se envía a un objeto "OldCar". Pero significa algo diferente si se envía al objeto "NewCar". Es un concepto natural que se puede aplicar a los objetos. También significa que objetos similares a menudo aceptan el mismo mensaje pero hacen cosas diferentes.

Considere una página web. Es un objeto. Hay miles de millones de esos objetos a nuestro alrededor. Cuando se envía una petición a un objeto como una página web, se aplica realmente un verbo "GET" a un sustantivo "WebPage". Ahora cada objeto "WebPage" no se comporta de la misma manera cuando se aplica el verbo "GET". Alguien abre un archivo PDF, alguien simplemente

muestra algunos textos e imágenes y alguien puede dañar su ordenador. Al hacer doble clic en un archivo, puede ejecutarse si se trata de un archivo ejecutable. O puede abrirse en un editor de texto si se trata de un archivo de texto. El mensaje

es la misma. Esto es "Double-Click". Pero el comportamiento que muestra el objeto de archivo depende del objeto mismo.

Esto es polimorfismo. Lo aprenderá de memoria a medida que avance en este capítulo.

La ventaja de las clases Python es que proporcionan todas las características estándar de la programación orientada a objetos. Tiene el mecanismo de herencia de clase. Eso permite múltiples clases base. Una clase derivada puede anular cualquier método de su clase o clases base, y un método puede llamar al método de una clase base con el mismo nombre. Los objetos pueden contener cantidades y tipos de datos arbitrarios.

Finalmente, recuerda, en Python todo es un objeto. Significa que hay una abstracción o encapsulación detrás de ella. Primero necesitas entender la abstracción y luego crear tu propia abstracción.

---

## Entendiendo Clases y Objetos

No se puede entender la teoría a menos que se implemente ese concepto en el mundo real. Veamos lo que hemos aprendido.

**1. Las clases son cuando usted crea su propio objeto.**

**2. Una clase es un plano de un objeto.**

**3. Un objeto es una instancia de una clase.**

Veamos cómo podemos construir una clase y luego crear algunas instancias a partir de ella.

Considere este código:

```
Robot de clase
<code>
#!/usr/bin/python3
:
def init (self):
    pasar
def WalkLikeARobot(self):
    print("camina como un robot")
def CareLikeARobot(self):
    print("takes care like a robot.")
robul = Robot()
```

```

print(type(robu1))
print(id(robu1))
robu2 = Robot()
print(type(robu2))
print(id(robu2))
del robu2
def main():
    robu = Robot()
    print(type(robu))
    print(id(robu))
si nombre __ == " principal__":
    main()
</código>

```

En este código, tenemos la definición de clase "Robot". Aquí "clase" es la palabra clave. Al lado hay un signo "::", que significa que una definición de clase seguirá a una suite o bloque de códigos. Después de haber definido la clase "Robot", tenemos tres métodos.

Y lo son...:

```

def init (self):
    pasar
def WalkLikeARobot(self):
    print("camina como un robot")
def CareLikeARobot(self):
    print("takes care like a robot.")

```

El primero es el método especial. Cuando una clase es instanciada, este método será llamado primero. "init" significa inicialización. Se inicializa la clase. Otros dos métodos lo siguen. Estos métodos se explican por sí solos. Los métodos son verbos de acción. Cuando creamos un objeto robot y llamamos a esos métodos, les decimos que hagan algo. En nuestra clase definimos lo que harán.

En este código creamos tres objetos robot. Y finalmente no les dijimos que hicieran nada. Acabamos de ver en qué se diferencian entre sí. Hemos probado su tipo y su identificación. Mira, cada objeto tiene un ID diferente. Así que este es un punto importante. Cada objeto o instancia creada a partir de una clase, tiene su propia individualidad.

Ahora vea el resultado:

```
<cotización en bloque>.  
<Robot'>  
14044535354614624  
<Robot'>  
14044535354668160  
<Robot'>  
14044535354668160  
</blockquote>
```

Las siguientes líneas de código son un poco más largas, pero te sugiero que las escribas en tu propio editor de texto y ejecutes el programa para asegurarte de que obtienes la misma salida.

```
/usr/bin/python3
class Robots:
    def init (self):
        pasar
    def WalkLikeARobot(self, style):
        self.style = estilo
        devolver self.style
    def
    CareLikeARobot(self):
        print ("takes care like a robot.")
class Humans:
    def init (yo, naturaleza = "bueno"):
        auto-naturaleza =
    naturaleza def Bien-
    SerHumano(a):
        print("no es necesario repetir, un buen
ser humano es siempre", self.nature)
    def BadHUMANBeing(self):
        self.nature = "no es necesario repetir, el
mal ser humano es siempre malo."
        print(self.nature)
    def WalkLikeARobot(self, style):
        self.style = estilo
        devuelve
    self.style def main():
        robu = Robots()
```

```

    robu.CareLikeARobot()
    print(robu.WalkLikeARobot("camina como un
robot")))
    GoodMan = Humanos()
    print(GoodMan.nature)
    GoodMan.GoodHumanBeing()
    BadMan = Humanos()
    BadMan.nature =
        Impresión "mala"
    print(BadMan.nature)
    BadMan.BadHUMANBeing()
    print(BadMan.WalkLikeARobot("es humano pero camina
como un robot"))
    si nombre _ == " principal ":
        main()
</código>

```

En el fragmento de código anterior, tenemos dos clases. Uno es "Robot", que escribimos antes. La otra clase es "Humano". En la clase "Humana", hemos definido este método especial así:

```

def init (yo, naturaleza =
            "bueno"): yo, naturaleza =
            naturaleza

```

¿Qué significa esto? Significa que cuando creamos una instancia humana de esta clase, asumimos que la naturaleza del objeto humano será buena por defecto.

Desafortunadamente, esto no sucede en el mundo real. Teniendo eso en cuenta, también escribimos esta línea: "auto-naturaleza = naturaleza". Significa que la auto-naturaleza o la naturaleza de la instancia será buena si no mencionamos explícitamente que es "mala" o algo más.

En los siguientes pasos, cuando creamos una mala instancia humana, cambiamos explícitamente la naturaleza. Recuerde, cada método es la parte de acción de ese objeto. Un objeto es un sustantivo y hace algo. En cualquier aplicación de software sigue la misma regla. También hay un ejemplo de polimorfismo. En ambas clases, "Robot" y "Humano", hemos definido un método :

```

def WalkLikeARobot(self, style):
    self.style = estilo
    volver a self.style

```

Cuando aplicamos este mismo verbo a los diferentes objetos Robot y Humano, muestra un comportamiento diferente. Si ejecutas este código, nos da una salida como esta:

```
<cotización en bloque>.  
Cuida como un robot.  
Camina como un robot.  
bueno  
no es necesario repetir, un buen ser humano es  
siempre bueno malo  
no hace falta repetir, el ser humano malo  
siempre es malo. es humano pero camina como un  
robot  
</blockquote>
```

Cuando una instancia de Robot camina como un robot, se muestra: **camina como un robot**; pero cuando una instancia de Humano camina como un robot, se muestra: **es humano pero camina como un robot**. Esto no es más que un simple ejemplo de polimorfismo.

Cuando el mismo verbo se aplica a dos objetos diferentes, dependiendo de la naturaleza del objeto, da una salida diferente.

En realidad, cambiamos este comportamiento pasando dos argumentos diferentes.

Supongamos, en lugar de un solo argumento, que pasamos un diccionario de valores. Mira cómo se magnifica la potencia. Considere un código simple a continuación:

```
<código>  
print(type(BadMan.WalkLikeARobot(dic(uno=1,  
dos=2)))))  
st = BadMan.WalkLikeARobot(dic(uno=1, dos=2))  
para llaves en orden(st):  
    print(keys, st[keys])  
ws = BadMan.WalkLikeARobot({'uno':56, 'dos':2})  
para llaves en orden(ws):  
    print(keys, ws[keys])  
</código>
```

Aquí está el resultado:

```
<cotización en bloque>.  
class <class  
'dic'> one 1
```

**dos 2**

```
uno 56  
dos 2  
</blockquote>
```

Puede agregar más pares de claves y valores a este diccionario y ejecutar este código para ver qué sucede.

---

## Escribe tu propio juego, "Bueno vs. Malo"

Hasta ahora hemos aprendido muchas cosas. Espero que hayas escrito los códigos y los hayas probado y se haya ejecutado perfectamente. Ahora ha llegado el momento de escribir un juego sencillo en Python. Es un juego llamado "Bueno contra Malo". El juego es simple. Pero como principiante puedes encontrar este código un poco más largo. Escríbelo. Intenta añadir más funciones.

Si está en un entorno Linux, guarde este archivo como "good-vs-bad.py" y cambie el archivo ejecutable ejecutando este comando:

```
Sudo chmod+x good-vs-bad.py
```

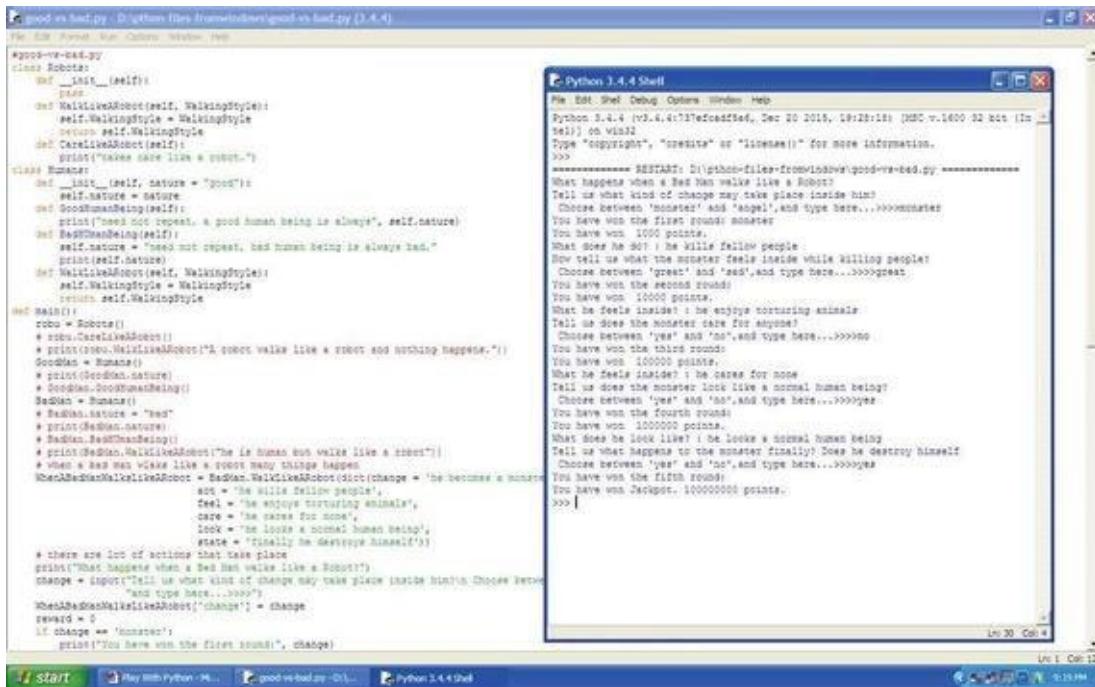
Y luego póngalo en su terminal de esta manera:

```
./ good-vs-bad.py
```

Si está en Windows, ejecute el programa IDLE y guarde el archivo como "good-vs-bad.py".

Presione F5 y juegue el juego.

Se ve así en la pantalla de la computadora:



**Figura 15-1.** Jugando al juego de Python en el IDE de Windows

En el fondo, el código se muestra y puedes jugar el juego en Python Shell.

El código es así:

```
/usr/bin/python3
class Robots:
    def __init__(self):
        pasar
    def WalkLikeARobot(self, WalkingStyle):
        self.WalkingStyle = WalkingStyle
        volver self.WalkingStyle
    def CareLikeARobot(self):
        print("takes care like a robot.")

humanos de la clase:
    def __init__(yo, naturaleza =
                "bueno"): yo, naturaleza =
                naturaleza

    def BuenoSerHumano(a):
        print("no es necesario repetir, un buen
             ser humano es siempre", self.nature)
```

```

def BadHUMANBeing(self):
    self.nature = "no es necesario repetir, el
mal ser humano es siempre malo."
    print(self.nature)
def WalkLikeARobot(self, WalkingStyle):
    self.WalkingStyle = WalkingStyle
    volver self.WalkingStyle
def main():
    robu = Robots()
    # robu.CareLikeARobot()
    # print(robu.WalkLikeARobot("Un robot camina como
un robot y no pasa nada."))
    GoodMan = Humanos()
    # imprimir(GoodMan.nature)
    # GoodMan. GoodHumanBeing()
    BadMan = Humanos()
    # BadMan.nature = "bad"
    # print(BadMan.nature)
    # BadMan. BadHumanBeing()
    # print(BadMan.WalkLikeARobot("es humano pero
camina como un robot"))
    # cuando un hombre malo golpea como un robot
    # muchas cosas #
acontecer
    CuandoABadManWalksLikeARobot =
BadMan.WalkLikeARobot(dict(cambio = 'se convierte en
un monstruo dentro',
                            actúa = 'mata
"compañeros", "tort
urando
animales", "ninguno
",
ser humano normal",
se destruye a sí
mismo'))))
    # hay muchas acciones que tienen lugar

```

```
print("¿Qué pasa cuando un Hombre Malo camina  
como un
```

```

¿Robot?")
    change = input("Dinos qué tipo de cambio puede
ocurrir dentro de él? Elige entre 'monstruo' y 'ángel', "
                    "y escriba aquí...>>>>")
CuandoABadManWalksLikeARobot['cambiar'] =
cambiar recompensa = 0
si cambio == 'monstruo':
    print("Has ganado la primera ronda:",
cambio)
    recompensa = 1000
    print("Has ganado", recompensa, "puntos.")
    print("¿Qué hace? :",
CuandoABadManWalksLikeARobot['act'])
    cambio = entrada ("Ahora dinos que siente
el monstruo dentro mientras mata a la gente? Elige
entre 'grande' y 'triste', "
                    "y escriba aquí...>>>>")
CuandoABadManWalksLikeARobot['cambiar'] =
alteració
n
    si cambio == 'grande':
        print("Has ganado la segunda ronda:")
        recompensa = 10000
        print("Has ganado", recompensa,
"puntos.")      print ("What he feels inside? :"),
CuandoABadManWalksLikeARobot['feel'])
    change = input("Tell us does the
monster care for anyone?\N Choose between 'yes' and
`no', "
                    "y escriba aquí...>>>>")
CuandoABadManWalksLikeARobot['cambiar'] =
alteraci
ón
    si cambio == 'no':
        print("Has ganado la tercera
redonda:
                    recompensa = 100000
        print("Has ganado", recompensa,
")

```

```

"puntos .")
                print ("What he feels inside? :"),
CuandoABadManWalksLikeARobot[ 'care' ])
                cambio = entrada ("Dinos si el
monstruo se parece a un ser humano normal? Elige
entre 'sí' y 'no', "
                        "y escriba aquí...>>>>")
CuandoABadManWalksLikeARobot[ 'cambiar
...] =
cambio           si el cambio es ==
'sí':
                print("Has ganado la cuarta
redonda:")
                    recompensa = 1000000
                    print("Has ganado", recompensa,
"puntos .")
                print("¿Qué aspecto tiene?
", WhenABadManWalksLikeARobot[ 'look' ])
                change = input("Dinos que le
pasa al monstruo finalmente? ¿Se destruye a sí mismo?
Escoge entre "sí" y "no".
                        "y escriba
aquí...>>>>""
CuandoABadManWalksLikeARobot[ 'ch
) nge' ] =
cambiar           a if change == 'yes':
                    print("Usted ha ganado el

                    reward =
1000000000
                    print("Has ganado
Premio mayor", recompensa, "puntos")
                    de los demás:
                        print("Has cambiado el
curso del juego. Esto termina aquí. Has perdido",
recompensa - 100000, "puntos")
                    de los demás:
                        print("Has cambiado el curso
del juego. Esto termina aquí. Has perdido", recompensa
- 1000, "puntos")

```

```
        de los demás:  
            print("Has cambiado el curso del  
juego. Esto termina aquí. Has perdido", recompensa  
- 100, "puntos")  
        de los demás:  
            print("Has cambiado el curso del juego.  
Esto termina aquí. Has perdido", recompensa - 10,  
"puntos")  
        de los demás:  
            print("Has cambiado el curso del juego.  
Termina aquí y no has ganado ningún punto.")  
    si nombre _ == "principal":  
        main()  
</código>
```

Y la salida de tu Python Shell se ve así:

<cotización en bloque>.

¿Qué pasa cuando un hombre malo camina como un  
robot?

Díganos qué tipo de cambio puede ocurrir dentro de  
él.

Elija entre 'monster' y 'angel', y escriba  
aquí...>>>monster

Has ganado la primera ronda: monster  
Tieneswon 1000 puntos.

¿Qué hace? Mata a sus compañeros.

Ahora dinos qué siente el monstruo dentro mientras  
mata a la gente.

Elija entre 'grande' y 'triste', y escriba  
aquí...>>>grande

Has ganado la segunda ronda:  
Tieneswon 10.000 puntos.

¿Qué siente por dentro? Disfruta torturando  
animales ¿Dinos si el monstruo se preocupa por  
alguien?

Elija entre 'sí' y 'no', y escriba aquí...>>>no

Has ganado la tercera ronda:  
Tieneswon 100.000 puntos.

¿Qué siente por dentro? No le importa nada.  
Díganos, ¿el monstruo parece un ser humano normal?  
Elija entre 'sí' y 'no', y escriba aquí...>>>sí  
Has ganado la cuarta ronda:  
Tieneswon 10.000.000 puntos.  
¿Qué aspecto tiene? Parece un ser humano normal.  
Díganos qué le pasa al monstruo finalmente. ¿Se destruye a sí mismo  
Elija entre 'sí' y 'no', y escriba aquí...>>>sí  
Has ganado la quinta ronda:  
Has ganado el Jackpot. 10000000000 puntos.  
</blockquote>

Desde que escribí el código, gané el juego. Pero hay algunos trucos. En esas partes difíciles, si fallaba y suministraba entradas incorrectas, perdería.

---

## Clase Primaria y Objeto

Ahora la clase primaria y el objeto ya no deberían ser difíciles. Puede escribir una clase Human y pasar un argumento por defecto como "kind" en el proceso de inicialización. Puede establecerlo como "bueno". Ahora bien, si quieres crear un buen ser humano no necesitas pasar ningún argumento extra. En el siguiente paso, cuando aplicas un verbo como "SerHumano()" al buen ser humano, es por defecto bueno. Si quieres crear un ser humano malo, puedes cambiar ese argumento por defecto y hacerlo malo.

```
/usr/bin/python3
clase Humano:
    def init (self, kind = "bueno"):
        self.kind = kind
    def SerHumano(a):
        devuelve
        self.kind
def main():
```

```

bueno = Humano()
malo =
    Humano("malo")
    print(bueno.BeingHuman
        ())
    print(malo.BeingHuman
        ())
si nombre_ == " principal ":
    main()
</código>

```

El resultado es bastante obvio:

```

<cotización en bloque>.
bueno
malo
</blockquote>

```

Hay algunas cosas que necesitas entender. ¿Por qué usamos el "yo"? ¿Qué significa eso? Considere el código a continuación.

```

/usr/bin/python3
class MySelf:
    def init (self, nombre, cantidad):
        self.name = nombre
        self.quantity =
    cantidad def Eat(self):
        print(nombre propio, "come", cantidad
propia, "plátanos cada día.")
    def main():
        hagu = MySelf("Hagu",
        2)           mutu           =
        MySelf("Mutu",           3)
        hagu.Eat()
        mutu.Eat()
si nombre_ == " principal ":
    main()
</código>

```

En este código de clase "MySelf" tenemos dos métodos. Uno es el método especial de construcción "init" y el otro es "Eat()". Notarás que cada método tiene un argumento especial: "yo mismo". En realidad, hace

referencia al objeto que es

va a ser creado. Cuando escribimos una clase, asumimos que se crearán instancias. En este caso, creamos dos objetos, "hagu" y "mutu". Cuando aplicamos el verbo "Eat()" o llamamos al método a los objetos, es como si pasaran por el método. Establecemos los nombres y el número de plátanos que se comen. Y la salida de este código es así:

```
<cotización en bloque>.  
Hagu come 2 plátanos al día.  
Mutu come 3 plátanos al día.  
</blockquote>
```

Pero necesitamos ejemplos más concretos. Queremos conectarnos a nuestras bases de datos desde nuestras aplicaciones. Para ello necesitamos una clase donde tengamos métodos y propiedades que se conecten a las bases de datos.

Supongamos que tenemos dos configuraciones diferentes. Tenemos una base de datos MySQL y, además, queremos crear una conexión SQLite. Para ello podemos escribir dos clases separadas y establecer la conexión en la parte del constructor o en el método de inicialización. De modo que cuando creamos una instancia, la conexión a la base de datos se establece automáticamente.

Consideré el código:

```
<code>  
#!/usr/bin/python3  
importar sqlite3  
importar el conector mysql.connector  
desde la importación de  
mysql.connector Clase de error  
MySQLiteConnection:  
    def init (self):  
        db =  
            sqlite3.connect('testdb.db')  
        db.commit()  
        print("Connected to  
SqLite3") clase MyMySQLConnection:  
    def init (self):  
        Inténtalo:  
        ##### puedes usar un objeto del  
diccionario o puedes conectarte directamente ####  
        ##### using a dictioanry object ####  
        kwargs = dic(host = 'localhost', base
```

```
de datos = 'python_mysql', usuario = 'root',  
contraseña ='root'.
```

```

    pass')
            conn =
Conektor.mysql.connect(**kwargs)
        ##### conectando directamente #####
        conexión =
mysql.connector.connect(host = 'localhost',
                        database
se = 'python_mysql',
                        user =
'root',
                        password
rd = 'pass')
            si
            connection.is_connected():
                print("Conectado a MySQL desde el
objeto connection")
                # if conn.is_connected():
                #     print("Conectado desde 'conn')
objeto")
            excepto Error
                como e:
                    print(e)
            finalmente:
                conexión.close()
def main():
    ConnectToMySQL = MyMySQLConnection()
    ConenctToSqlLite = MySQLiteConnection()
si nombre _ == " principal ":
    main()
</código>

```

Creamos dos instancias u objetos de las clases **MyMySQLConnection()** y **MySQLiteConnection()** y los ponemos en dos variables separadas. Las conexiones se están configurando y en la sección de salida vemos esto:

```

<cotización en bloque>>.
Conectado a MySQL desde el objeto 'connection'
Conectado a SQLite3
</blockquote>

```

Pero este es un ejemplo extremadamente simple y mal escrito. Deberíamos desarrollar este código para que cada instancia de las clases MySQLConnection y SQLiteConnection pueda no sólo conectarse a la base de datos sino también recuperar datos de una tabla.

Reemplazemos nuestro antiguo código con esto:

```
< código >
#!/usr/bin/python3

importar sqlite3
importar el conector mysql.connector
desde el conector mysql.connector importar
MySQLConnection, Error

clase MySQLiteConnection:
    def init (self, db =
sqlite3.connect('test.db')):
        self.db = db
        db.row_factory = sqlite3.Row
        print("Connected to
SQLite3")
    def Recuperar (self):
        print("Retorno de valores de la prueba de
tabla1 de la prueba de base de datos SQLite")
        read = self.db.execute('select * from test1
orden por i1')
        para la fila leída:
            print(fila['t1'])
clase MyMySQLConnection:
    def init(self,         kwargs = dic(host =
'localhost', database = 'testdb', user =
'root', password = 'pass')):
        Inténtalo:
        ##### puedes usar un objeto del
diccionario o puedes conectarte directamente #####
        ##### using a dictioanry object #####
        self.kwargs =
        kwargs conn =
Conector.mysql.connect(**kwargs)
```

```

        if conn.is_connected():
            print("Conectado a la base de
                  datos MySql
testdb de 'conn' object")
        excepto Error
            como e:
                print(e)
        finalmente:
            conn.close()
def Retrieve(self):
    print("Retener registros de la base de
datos MySql testdb.")
    Inténtalo:
        conn = MySQLConnection(**self.kwargs)
        cursor = conn.cursor()
        cursor.execute("SELECT * FROM
EMPLEADO")
        rows = cursor.fetchall()
        print('Total Fila(s) :', cursor.rowcount)
        para fila en filas:
            print("Nombre = ", fila[0])
            print("Segundo Nombre = ", fila[1])
            print("Edad=", fila[2])
            print("Sexo = ", fila[3])
            print("Salario =", fila[4]) excepto Error como e:
            print(e)
        finalmente:
            cursor.close()
            conn.close()

def main():
    ConnectToMySQL = MyMySQLConnection()
    ConnectToMySQL.Retrieve()
    ConenctToSQLite = MySQLiteConnection()
    ConenctToSQLite.Retrieve()
    si nombre _ == " principal ":
        main()
</código>

```

Nos hemos conectado a cada base de datos con el proceso de inicialización y luego aplicamos un verbo, "Retrieve()", a cada objeto. También hemos importado muchos módulos de bases de datos que aún no has aprendido.

Usted los aprenderá en el debido proceso. Pero nuestro propósito está cumplido. Creamos dos objetos de base de datos separados. Uno es un objeto de conexión MySQL y otro es un objeto de conexión SQLite. Después de eso, con esos objetos podemos recuperar datos separados de dos tablas diferentes.

Primero mira la salida:

```
<cotización en bloque>.  
Conectado a la base de datos MySql testdb desde  
el objeto 'conn  
Recuperación de registros de la base de datos  
MySql testdb. Total Filas: 3  
Primer Nombre= Mac  
Segundo Nombre  
=Mohan Edad= 20  
años  
Sexo= M  
Salario= 2000.0  
Primer Nombre= Mac  
Segundo Nombre  
=Mohan Edad= 20  
años  
Sexo= M  
Salario= 2000.0  
Primer Nombre= Mac  
Segundo Nombre  
=Mohan Edad= 20  
años  
Sexo= M  
Salario= 2000.0  
Conectado a SqLite3  
Retorno de valores de la tabla test1 de la prueba  
de base de datos SqLite  
Babu  
Mana  
Bappa  
Babua  
Anju  
Patai
```

**GasaBuddhu**

## **Tapas**

```
</blockquote>
```

La salida dice, la base de datos MySQL "testdb" tiene una tabla llamada "Empleado" y hay varias filas como nombre, sexo, salario, etc. Segundo, tenemos una base de datos SQLite3 "test1" que tiene una tabla llamada "test1" que tiene muchas filas que contienen pocos nombres.

---

## Acceso a los datos de objeto

Cuando se crea un objeto a partir de una clase es bastante obvio que tendrá algún tipo de datos. La cuestión es cómo podemos acceder a esos datos. ¿Cuál es el camino correcto? Debemos acceder a esos datos de manera que podamos hacer un seguimiento de ellos.

Considere este código a continuación:

```
/usr/bin/python3
clase Humano:
    def init (self, height = 5.08):
        self.height = height
def main():
    ramu = Human()
    imprimir(ramu.height)
    ramu.height = 5.11 # se llama efecto
secundario y difícil de seguir
    imprimir(ramu.height)
si nombre __ == " principal ":
    main()
</código>
```

En este código vemos la clase Humana con una altura por defecto, que es 5.08. Cuando creamos un objeto, esta altura se establece automáticamente a menos que lo cambiemos o lo mencionemos explícitamente. También podemos establecer cualquier propiedad fuera de ese objeto. En la siguiente línea hemos escrito **ramu.height = 5.11**.

Podemos establecer cualquier propiedad de objeto como esta. Pero esto se llama efecto secundario y es muy difícil de rastrear. Así que tenemos que hacerlo de una manera más estructurada. ¿Cómo podemos hacer eso? Veamos primero el resultado de este código.

```
<cotización en bloque>>.
```

**5.08**

**5.11**

</blockquote>

Usted ve los cambios de altura y no sabemos cuál es la altura correcta del objeto "ramu". Para resolver este problema, se implementa el método accessor.

Los métodos accessor son métodos que primero establecen el valor y luego a través de ese método se puede obtener el valor.

```
/usr/bin/python3
clase Humano:
    def init (self):
        pasar
    # Accesorio
    def set_height(self,
                  height): self.height =
                  height
    def get_height(self):
        retornar a la
        altura propia
def main():
    ramu = Humano()
    # ramu.height = 5.11 # se llama efecto
secundario y difícil de rastrear
    ramu.set_height(5.12)
    print(ramu.get_height())
si nombre __ == " principal ":
    main()
</código>
<cotización en bloque>.

5.12
</blockquote>
```

Pero todavía nos falta algo. Queremos añadir más flexibilidades para que con menos código podamos hacer más trabajos.

```
/usr/bin/python3
clase Humano:
    def init (self, **kwargs):
```

```

        variables propias = kwargs
    def set_manyVariables(self, **kwargs):
        self.variables = kwargs
    def set_variables(self, key, value):
        self.variables[key] = valor
    def get_variables(self, key):
        return self.variables.get(key, None)
def main():
    mana = Humano(nombre = 'Mana')
    print("Nombre del maná del
objeto:", 
mana.variables['nombre'])
    ManaName = mana.variables['nombre']
    mana.set_variables('class', 'two')
    print(ManaName, "reads at class",
mana.get_variables('class'))
    mana.set_manyVariables(school = 'balika
school', height = 4.54)
    print(ManaName, "has height
of",
mana.variables['altura'], "y el nombre de su escuela
es", mana.variables['escuela'])
    babu = Humano (nombre = `Babu', estudiante_de =
`Clase Tres', dice_at = ' Balak School', altura =
5.21)
    BabuName = babu.variables['nombre']
    print(BabuName, "he is a student of",
babu.variables['student_of'], "and he reads at",
    babu.variables['reads_at'], "y su
height is", babu.variables['height'])
    si nombre _ == " principal ":
        main()
</código>

```

En este fragmento de código tenemos muchas opciones disponibles. Hemos establecido nuestras variables en un formato de diccionario. Después de eso podemos obtener el valor a través de la tecla.

<cotización en bloque>.

**Nombre del objeto Mana**

**Mana: Mana Mana lee en**

**la clase dos.**

**Mana tiene una altura de 4.54 y su escuela se llama Escuela Balika.**

**Babu es alumno de la clase 3 y lee en la escuela Balak y su estatura es de 5.21.**

</blockquote>

Este no es el único método para abordar los datos de objetos. A medida que avance, verá muchos ejemplos diferentes de manejo de datos.

---

## Polimorfismo

El polimorfismo es un concepto muy importante en la programación orientada a objetos. Lo básico es que cuando aplicamos el mismo verbo a dos objetos diferentes, dependiendo de los objetos, reaccionan de forma diferente. Cuando ponemos una casa vieja a la venta, alcanza un cierto valor. Pero cuando ponemos una casa nueva a la venta, alcanza un precio y valor más altos. Así que en este caso cuando aplicamos el método de "venta" o el verbo "venta" a diferentes objetos, se comportan de manera diferente.

```
/usrbin/python3
class Tabla:
    def __init__(self):
        pasar
    def ItHolds(self):
        print("Una mesa contiene libros, blocs de
notas en
...")
    def YouCanWriteOnIt(self):
        print("Puedes escribir en una tabla.")

Libro de la clase:
    def __init__(self):
        pasar
    def ItHelps(autoayuda):
        print("Un libro nos ayuda a saber algo
nuevo.")

def main():
    MyTable =
    Table()
    MyBook =
    Book()
```

```

MyTable.ItHolds()
MyTable.YouCanWriteOnit()
MyBook.ItHelps()
si nombre _ == " principal ":
    main()
</código>

```

Estas son clases bastante simples y el resultado es también muy simple.

```

<cotización en bloque>.
Una mesa tiene cosas en
ella. Puedes escribir en
una mesa.
Un libro nos ayuda a conocer algo nuevo.
</blockquote>

```

Esta salida puede cambiar drásticamente cuando se aplican los mismos verbos o métodos a los objetos de las clases "Table" y "Book". Considere los siguientes códigos.

```

/usrbin/python3
class Tabla:
    def init (self):
        pasar

    def Get(self):
        print("Please get me that table.")
    def Put(self):
        print("Por favor, ponga la mesa en la
esquina de la habitación.")
    def Destruir (a sí mismo):
        print ("Algunas personas vinieron y no
querían que leyéramos y escribiéramos. Ellos destruyeron
la mesa.")
    Libro de la clase:
        def init (self):
            pasar
        def Get(self):
            print("Please get me that book.")
        def Put(self):

```

```

        print("Pusimos algunos libros nuevos en el
mesa.")
    def Destruir (a sí mismo):
        print ("Algunas personas vinieron y no
querían que leyéramos y escribiéramos. Destruyeron el
libro.")
    def main():
        MyTable = Table()
        MyBook = Book()
        InMistake(MyBook)
        Intentionally(MyTable)
def
    InMistake(Table)
    : Table.Get()
    Table.Put()
    Table.Destroy()
    Table.Destroy()
def Intencionalmente
    (Libro): Book.Get()
    Book.Put()
    Book.Destroy()
si nombre _ == " principal ":
    main()
< código

```

Hay tres métodos: Get, Put y Destroy. Se puede ver cómo reaccionan de forma diferente los objetos de tabla y los objetos de libro a estos métodos.

<cotización en bloque>.

**Por favor, consígueme ese libro.**

**Pusimos algunos libros nuevos sobre la mesa.**

**Algunas personas vinieron y no querían que  
leyéramos y escribiéramos. Destruyeron el libro.**

**Por favor, consígueme esa mesa.**

**Por favor, pon la mesa en la esquina de la  
habitación.**

**Algunas personas vinieron y no querían que  
leyéramos y escribiéramos. Destruyeron la mesa.**

</blockquote>

En Python, un objeto generador se utiliza en un contexto donde la iteración es necesaria. Normalmente, en este caso, nos basamos en dos métodos: **def init (self, \*args)** y **def iter (self)**. Configuramos la lógica en el método constructor e iteramos a través de ella mediante la función **def iter (auto)**.

```
/usr/bin/python3
class InclusiveRange:
    def init (self, *args):
        numberArguments = len(args)
        if numberArguments < 1:
            raise
        TypeError('Se requiere al menos un argumento.')
    elif numberArguments == 1:
        self.stop =
            args[0] self.start
            = 0
            paso propio = 1
    elif numberArguments == 2:
        # start y stop serán tuple
        (auto.start, stop) = args
        auto.step = 1
    elif numberArguments == 3:
        # todo empieza y se detiene y el paso
        será
tupla
        (arranque.automático,
args
        parada.automática, .paso.automático)

        =.si.no:.subir TypeError("Máximo tres
argumentos. Usted dio
{}".format(numberArguments)".format)

    def iter (self): i
        = self.start
        mientras que i <=
            self.stop: yield
            i
            i += paso propio

def main():
```

```
ranges = InclusiveRange(5, 210, 10)
para x en ranges:
    print(x, end=' ')
```

```
si nombre __ == " principal ":
    main()
</código>
```

Este código significa que puede controlar el rango de iteración. Empezamos desde las 5 y terminamos en 210. En cada paso progresamos por 10.

```
<cotización en bloque>.
5 15 25 35 45 55 65 75 85 95 105 115 125 135 145
155 165 175 185 195 205
</blockquote>
```

Podemos obtener el mismo efecto sin usar esos métodos. Podemos simplemente escribir de esta manera.

```
<código>

## La función de abajo también funciona
perfectamente, pero no es un generador ##

def RangeFunctions(self, *args):
    numberArguments = len(args)
    si numberArguments < 1:
        raise
TypeError('Se requiere al menos un argumento.')
    elif numberArguments == 1:
        self.stop =
            args[0] self.start
            = 0
        paso propio = 1
    elif numberArguments == 2:
        # start y stop serán tuple
        (auto.start, stop) = args
        auto.step = 1
    elif numberArguments == 3:
        # todo empieza y se detiene y el paso
        será
tupla
        (arranque.automático,
args
        parada.automática, .paso.automático)
```

```
=.si.    no:.subir TipoError("Máximo tres  
argumentos. Usted dio  
{ }".format(numberOfArguments) ".format)
```

```
i = auto.start
mientras que i <=
    self.stop: yield
    i
    i += paso auto.
```

</código>

---

## Herencia

La herencia es un concepto igualmente importante en la programación orientada a objetos. Hay una clase para padres y otra para niños. La clase secundaria generalmente hereda todas las propiedades y métodos de la clase principal. Al mismo tiempo, puede cambiar todas las propiedades y métodos según la situación.

La forma en que una clase infantil hereda es muy simple. Cuando declaramos una clase infantil, escribimos el nombre de la clase de los padres dentro de la clase infantil de esta manera: ChildClass(ParentClass).

```
/usr/bin/python3
class AllUsers:
    def init (self):
        pasar
    def Register(self):
        print("Please
              Register")
    def Login(self):
        print("Bienvenido
              Miembro")
class Admin(AllUsers):
    def init (self):
        de paso
    def Register(self):
        print("Los administradores no
              necesitan registrarse") def
    Login(self):
        print("Welcome
              Admin") class
Members(AllUsers):
    def init (self):
        pass
```

```
def main():
    admin = Admin()
```

```

admin.Register()
admin.Login()
member =
Members()
member.member.Reg
ister()
member.Login()
si nombre_ == "principal":
    main()
</código>

```

La clase Parent es "**AllUsers()**". Hay dos clases para niños: "**Admin**" y "**Miembros**". A través de las clases para niños heredamos todas las propiedades y métodos de la clase para padres. En la clase de padres, mencionamos que todos los usuarios deben registrarse y conectarse. Ahora en la clase infantil "**Admin**" anulamos los métodos, pero en la clase "**Members**" no los cambiamos. Cuando creamos una instancia de la clase "**Admin**", tiene sus propias propiedades y métodos. Pero en la clase "**Members**", decidimos no anular los métodos de la clase padre. Es evidente en el siguiente resultado.

```

<cotización en bloque>>.
Los administradores no
necesitan registrarse.
Por favor,
regístrate
como miembro
de bienvenida.
</blockquoteo>

```

---

## Decorador

Los decoradores son funciones especiales que devuelven funciones. Normalmente, para establecer una propiedad de objeto normalmente la obtenemos a través de otra función.

```

/usr/bin/python3
class Dog:

    def init(self, **kwargs):
        self.properties = kwargs
    def get_properties(self):

```

```
    devuelve propiedades
    propias
def set_properties(self, key):
```

```

    self.properties.get(key, None)

def main():
    lucky = Perro(naturaleza ='obediente')
    print(lucky.properties.get('naturaleza'))

if nombre __ == " principal__":
    main()
</código>

```

El resultado es bastante simple.

```

<cotización en bloque>>.
obedientes
</blockquoteo>

```

En Python, "Decorator" es simplemente un método mediante el cual decoramos un método de accesorios para una variable, y la función comienza a comportarse como una propiedad. La belleza de este decorador es que puede utilizar la función como una propiedad y después de crear el objeto puede controlar la configuración de la propiedad y obtenerla. Vea el siguiente código.

```

/usr/bin/python3
class Dog:

    def init (self, **kwargs):
        self.properties = kwargs
    Propiedad @property
    def Color(self):
        return self.properties.get('color', None)
    @Color.setter
    def Color(self, color):
        self.properties['color'] = color
    @Color.deleter
    def
    Color(self):
        del
    self.properties['color'] def
main():
    lucky = Perro()

```

```
# ahora vamos a usar la función de decorador como  
una propiedad normal  
lucky.Color = impresión'negro y amarillo'  
(lucky.Color)  
  
si nombre __ == " principal ":  
    main()  
</código>
```

El resultado es el esperado:

```
<cotización en bloque>.  
negro y amarillo  
</blockquote>
```

Es un ejemplo muy simple donde vemos que una sintaxis habitual de función puede ser escrita como una sintaxis de propiedad. Es más conveniente cuando usamos este método de decoración para guardar archivos dentro de una base de datos.

En el último capítulo, veremos la aplicación web "Flask." Veremos cómo podemos usar este decorador para enrutar nuestras páginas web.

## 16. Métodos de cuerdas

Sanjib Sinha 

(1) Howrah, Bengala Occidental, India

---

En Python una cadena es un objeto. Como instancia de "class string" puede llamar a cualquier función o propiedad. Podemos cambiar una cadena a mayúsculas simplemente llamando a una función `upper()`.

Abramos nuestro terminal y escribamos esto:

```
< código  
hagudu@hagudu-H81M-S1:~ $ pitón3  
  
Python 3.4.0 (por defecto, 19 Jun 2015,  
14:20:21) [GCC 4.8.2] en linux
```

Para más información, escriba "ayuda", "copyright", "créditos" o "licencia".

```
>>> "Esto es una  
cadena""Esto es una  
cadena  
  
>>> s = 'esto es una cadena'.  
  
>>> s  
  
estos es una cadena''.  
  
>>> s.upper()
```

```
ESTO ES UNA CADENA''.

>>> s = 'esto es una cadena ahora vamos a añadirle
un número entero como cadena {}''.

>>> formato.s(100)

esta es una cadena, ahora vamos a añadir un número
entero como cadena 100'.

>>> en python2 se escribió como %d' % 100 'en
python2 se escribió como 100

>>>

</código>
```

Acabamos de cambiar una cadena a mayúsculas y también hemos añadido un número entero a esa cadena.

En Python 2 se hizo así:

```
en python2 se escribió como %d' % 100
```

Pero en Python 3.4 en adelante no lo usaremos más. Usaremos la función format() de esta manera:

```
>>> s = 'esto es una cadena ahora vamos a añadirle un número
entero como cadena {}'.
>>> formato.s(100)
esta es una cadena, ahora vamos a añadir un número entero como
cadena 100'.
```

```
<código
>>> s = 'esto es una cadena'.
>>> s
'esto es una cadena''.

>>> s.upper()

'ESTO ES UNA
CUERDA' ''.
```

```

>>> s.lower()

`esto es una

cadena' ?

>>> "Esto es una cuerda

>>> s

Esto es una cuerda.

>>> s.swapcase()

SU      Es un truco.
MEMOR
IA

>>> s

Este      es una
cuerda''.

>>> s = esto es un cadena'

'.'

>>>

s.find('is') 2

>>>
</código>
```

Escribamos algunos métodos de cadenas más. Usted puede hacer casi todo con estos métodos. Puede usar upper(), lower(), strip(), replace, find(), y muchos más.

```

<código
#!/usr/bin/python3
s = 'esto es una
cadena'
print(s.find('is'))
newstring = s.replace('esto', 'eso')
print(newstring)
UpperString =
```

```
s.upper()  
print(UpperString)  
# la cadena es mutable, por lo que el id ha sido  
cambiado por el parámetro
```

```

misma cadena
print(id(s))
print(id(UpperString))
a = `esto es una cadena con muchos espacios en blanco
al principio y al final''.
# por defecto elimina el espacio en blanco desde el
principio y el final
RemovingWhiteSpace = a.strip()
print(RemovingWhiteSpace)
print(RemovingWhiteSpace.strip('esto'))
</código>

```

En el código anterior, primero averiguamos la posición de "es" y aparece como

2. Por qué? Porque la primera palabra es "esto" y la secuencia del carácter comienza como 0, 1, 2 y siguientes. Así que en la posición 0 hay "t", después en la posición 1 hay "h", y en la posición 2 hay "i", y empieza a leer desde allí.

Recuerde, la cadena es mutable. Por lo tanto, para el mismo contenido de cadena, el "ID" cambia. Lo hemos visto en nuestro código.

Finalmente, en este bloque de código vemos una función importante: strip(). Por defecto, elimina los espacios en blanco desde el principio y el final. De lo contrario, debe proporcionar el carácter que desea eliminar de la oración.

```

<cotización en bloque>.
2
que es una
cadena ESTO ES
UNA CUERDA
140141176379480
140141176379768
esta es una cadena con muchos espacios en blanco al
principio y al final
es una cadena con muchos espacios en blanco al
principio y al final
</blockquote>

```

Considere este código:

```

<código
x, y = 10, 11
f = "esto {} es añadido y después añadimos {}"

```

```

FormattedString = f.format(x, y)
print(FormattedString)
# Podríamos haberla escrito al estilo C
m, n = 10, 11
f = "este %d es añadido y después añadimos %d"
FormattedString = f % (x, y)
print(FormattedString)
</código>

```

La salida es la misma.

```

</blockquoteo>
este 10 se añade y después añadimos 11 este
10 se añade y después añadimos 11
</blockquoteo>

```

Pero la diferencia es que en la última parte hemos utilizado el estilo Python 2. En ese estilo, formateamos en estilo "C" y mencionamos qué tipo de valor queremos formatear. Aquí queríamos formatear "decimal", así que hemos escrito "%d".

A partir de Python 3.1 en adelante este estilo ha cambiado, porque esta **envoltura de dos llaves rizadas**, "{}", y la función **format()** hacen la magia. Ahora ya no es necesario mencionar el valor. Antes de eso, tenías que mencionar el valor que querías formatear. Así que se están añadiendo más libertad y poder.

Mira cómo podemos formatear un valor de diccionario en nuestra cadena:

```

<código
a, b = 10, 11
s = "This is {}, and that is
{}"      FormattedStirng      =
s.format(a,                      b)
print(FormattedStirng)
# cambiamos la posición
FormattedStirng = s.format(b, a)
print(FormattedStirng)
s = "Esto es {0}, y eso es {1} y esto también es
{0} y eso también es {1}"
FormattedStirng = s.format(a, b)
print(FormattedStirng)
# podemos cambiarla de acuerdo a nuestro deseo con
el argumento posicional

```

```

s = "Esto es {1}, y eso es {1} y esto también es
{0} y eso también es {1}"
FormattedStirng = s.format(a, b)
print(FormattedStirng)
# podemos usarlo como diccionario
s = "Esto es {mío}, y eso es {tu} y esto también
es {tu} y eso también es {mío}"
FormattedStirng = s.format(mine = a, your = b)
print(FormattedStirng)
# más personal del diccionario
s = "Este es mi deseo: {el mío}, y ese es tu deseo
Y esto también es mío: y eso también es mío: {mío}"
FormattedStirng = s.format(mine = "I want to remove
'I'", your = "Do you want to remove 'yourself' ")
print(FormattedStirng)
</código>

```

Y aquí está el resultado:

```

<cotización en
bloque>>.
Este es el 10, ese es 11
y
Este es el 11, ese es 10
y
Este es el 10, ese es 11 y esta tam es 10 y
y                               bié
                               n
eso también es 11
Esto es 11, y eso es 11 y esto también es 10 y eso
también es 11.
Esto es 10, y eso es 11 y esto también es 11 y eso
también es 10.
Este es mi deseo: quiero quitarme el"yo", y ese es
tu deseo: ¿quieres quitarte el"tú mismo"? y esto
también es mío: Quiero quitar el"yo" y eso también es
mío: Quiero quitarme el"yo".
</blockquote>

```

¿Cómo podemos probar que la cadena es inmutable?

```

<código
strings = "Esto es una cadena"

```

```
print(type(strings))
print(id(strings))
AnotherStrings = "This is a string"
print(type(AnotherStrings))
print(id(AnotherStrings))
print(strings.split())
palabras = strings.split()
words.append("and that ia also a string.")
print(type(words))
print(palabras[0])
NewWords =
"::".join(words)
print(NewWords)
NewWords =
",,".join(palabras)
print(NewWords)
words[0] =
"That"
print(words)
</código>
<cotización en bloque>.
<Clase "str">
13995620959543256
<Clase "str">
13995620959543256
['Esto', 'es', 'a', 'cadena']
class <class
'list'> Esto
Esto:es:a:cadena:y esto ia también una cadena.
Esta, es, una, cuerda, y esa es también una
cuerda. ['Eso', 'es', 'a', 'cadena', 'y que ia
también un
cadena."]
</blockquote>
```

## 17. Entrada y salida de archivos

Sanjib Sinha 

(1) Howrah, Bengala Occidental, India

---

Python tiene algunas funciones incorporadas para tratar con archivos. Puede abrir un archivo y leer lo que hay dentro. Puedes escribir un archivo. Ese archivo podría ser un archivo de texto o una imagen.

Cada vez que usamos el método `open()` y pasamos el modo como argumento. Para leer un archivo escribimos "`r`" y para escribir usamos "`w`". Consideremos un código donde en un objeto leemos un archivo y lo escribimos en otro archivo usando otro objeto en el siguiente paso.

```
< código
  infile = open('files.txt', 'r')
  outfile = open('new.txt', 'w')
  para la línea en el archivo:
    print(line,
  file=outfile) print("Done")
</ código>
```

Si copiamos de esta manera el tamaño del archivo se incrementa en el nuevo archivo de texto. Ahora tenemos un archivo comparativamente grande. "Files.txt" ahora es "5.4 KB" y el "new.txt" sólo tiene 134 bytes.

Si copiamos a la antigua manera, el nuevo archivo se convierte en "5,7 KB", un poco más grande que el anterior. Pero Python tiene la técnica de copiar por búfer para que el tamaño del búfer permanezca intacto.

Ahora vamos a escribir el contenido de "files.txt" en "new.txt", pero no a la antigua. El nuevo código es:

```

< código
BufferSize = 500000
infile = open('files.txt', 'r')
outfile = open('new.txt', 'w')
buffer = infile.read(BufferSize)
while len(buffer):
    outfile.write(buffer)
    print ("Está copiando, puede tardar un
poco....por favor espere ",..., end='')
    buffer = infile.read(BufferSize)
print()
print("Copiando
hecho.")
</ código>

```

El resultado es el esperado.

```

<cotización en bloque>>.
Está copiando, puede llevar algún tiempo....por
favor espere.....
Copia finalizada.
</blockquote>

```

Leer y escribir un archivo binario es lo mismo. Todo lo que tiene que hacer es cambiar el modo de "r" a "rb" y cambiar el modo de "w" a "wb". Eso es todo. Su código tiene este aspecto:

```

BufferSize = 500000000
infile = open('home.jpg', 'rb')
outfile = open('newimageofHome.jpg', 'wb')
buffer = infile.read(BufferSize)
mientras que len(buffer):
    outfile.write(buffer)
    print ("Está copiando una imagen, puede tardar
un poco....por favor espere ",..., end='')
    buffer = infile.read(BufferSize)
print()
print("Copiando
hecho.")
</ código>

```

## 18. Contenedores

Sanjib Sinha 

(1) Howrah, Bengala Occidental, India

---

En Python las tuplas y listas son tipos de array. Las tuplas son inmutables, pero las listas son mutables. Las tuplas se utilizan con el operador de comas y se puede iterar a través de la tupla con bastante facilidad. Como las tuplas son inmutables, no se puede añadir o actualizar el valor de una tupla. En las listas, puede actualizar o añadir nuevos valores con bastante facilidad. Abra su terminal en Linux e IDLE en Windows. Escriba el código a continuación y vea el resultado usted mismo. Por favor, lea los comentarios que se adjuntan con el código .

```
<Código>
#!/usr/bin/python3
tuples1 = 1, 2, 3, 4
imprimir(tipo(tuples1
))
imprimir(id(tuples1))
) tuples2 = (1, 2, 3,
4) imprimir(tipo(tuples2))
imprimir(id(tuples2))
imprimir(tuples1[0])
imprimir(tuples2[0])
# dará el último ítem
print(tuples2[-1])
print(type(tuples1[0]))
print(type(tuples2[0]))
print(id(tuples1[0]))
print(id(tuples2[0]))
# tuple is immutable we can not change any value
# 'tuple' object does not support item assignment
```

```

# tuples2[0] =
120 #
imprimir(tuples2)
# para hacer una tupla entera necesitas añadir un
separador de comas
IsItTuple = (1)
print(type(IsItTuple))
IsItTuple = (1,)
print(type(IsItTuple))
# veamos cómo se comporta la lista
lista1 =[1, 2, 3, 4]
print(type(list1))
print(id(list1))
# primera
 impresión del
artículo
(lista1[0])
# último artículo
print(lista1[-1])
# podemos cambiar el valor de un ítem de la lista
lista1[0] = 120
print(list1) # output: [120, 2, 3, 4]
</código>

```

La salida es así:

```

<cotización en bloque>.
<clase "tupla"
>
13979472590108
0
<clase "tupla"
>
13979472590092
0
1
1
4
clase <clase 'int'>
<clase
'int'>
10455040

```

10455040  
clase <clase 'int'>  
clase <tupla'tupla'>  
<clase "lista" >  
139794725273480

```
1
4
[120, 2, 3, 4]
</blockquote>
```

---

## Funcionamiento en Tuple y Listar Objeto

Abramos nuestro terminal y probemos cómo funcionan juntas las tuplas y las listas.

```
< código
root@kali:~# pitón3
Pitón 3.4.4.4 (predeterminado, Jan 5 2016,
15:35:18)
GCC 5.3.1 20160101] en linux
Para más información, escriba "ayuda", "copyright",
"créditos" o "licencia".
>>> t = (1,2,3,4)
>>> t
(1, 2, 3, 4)
>>> t[0]
1
>>> t = tupla(rango(25))
>>> tipo(t)
clase <tupla'tupla'>
>>> 50 in t
Falso
>>> 10 in t
Verdadero
>>> para i in t:print(i)
...
0
1
2
3
4
5
6
7
8
9
```

```
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
>>> l = lista(rango(20))
>>> tipo(l)
clase <clase 'lista'> > lista
>>> para i in l:
... imprimir(i)
    Archivo "<stdin>",
    línea 2 imprimir(i)
    ^
IndentaciónError:           un sangrado taco
esperado
>>> para i en l:print(i)
...
0
1
2
3
4
5
6
7
8
9
10
11
12
```

```
13
14
15
16
17
18
19
>>> l[2]
2
>>> 50 en l
Falso
>>> 12 en l
Verdadero
>>> t[0] = 25
Traceback (última llamada): Archivo
    "<stdin>", línea 1, en <módulo>
TypeError: el objeto 'tuple' no soporta la
asignación de ítems
>>> l[0] = 25
>>> imprimir(l)
[25, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
15, 16, 17, 18, 19]
>>> t.append(50)
Traceback (última llamada): Archivo
    "<stdin>", línea 1, en <módulo>
AttributeError: 'tuple' object has no attribute
'append' 'append'.
>>> l.append(120)
>>> imprimir(l
l           lambda      len()       license( lista(l
ocals(
>>> imprimir(l)
[25, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
15, 16, 17, 18, 19, 120]
>>> t.count()
Traceback (última llamada): Archivo
    "<stdin>", línea 1, en <módulo>
TypeError: count() toma exactamente un argumento (0
```

```
dado)
>>> t.count(5)
1
>>> l.append(25)
>>>
L.count(25) 2
>>>
t.index(10) 10
>>> Índice de
1.1.(10) 10
>>> l.extend(range(25))
>>> para i en l:print(i)
...
25
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
120
25
0
1
2
3
```

```
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
>>> inserto de l.l.(0, 4656)
>>> l[0]
4656
>>> l.insert(12, 147)
>>> índice de
l.l(12) 14
>>> l[12]
147
>>> l.remove(12)
>>> l[12]
147
>>> imprimir(l)
[4656, 25, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 147, 11,
13, 14, 15, 16, 17, 18, 19, 120, 25, 0, 1, 2, 3, 4,
5,
6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
20, 21, 22, 23, 24]
>>> l.remove(12)
>>> imprimir(l)
```

```
[4656, 25, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 147, 11,
13, 14, 15, 16, 17, 18, 19, 120, 25, 0, 1, 2, 3, 4,
5,
6, 7, 8, 9, 10, 11, 13, 14, 15, 16, 17, 18, 19, 20,
21, 22, 23, 24]
>>> L.pop(0)
4656
>>> imprimir(1)
[25, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 147, 11, 13,
14, 15, 16, 17, 18, 19, 120, 25, 0, 1, 2, 3, 4, 5, 6,
7, 8, 9, 10, 11, 13, 14, 15, 16, 17, 18, 19, 20, 21,
22, 23, 24]
>>> l.pop()
24
>>> imprimir(1)
[25, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 147, 11, 13,
14, 15, 16, 17, 18, 19, 120, 25, 0, 1, 2, 3, 4, 5, 6,
7, 8, 9, 10, 11, 13, 14, 15, 16, 17, 18, 19, 20, 21,
22, 23]
>>>
</código>
```

Escriba el mismo código y vea cómo funciona en su máquina. Los errores pueden salir como sucedió en el código anterior. Pero recuerda, cada error te ayudará a aprender algunas cosas nuevas.

---

## Funcionamiento en el objeto de diccionario

Como ha probado tuplas y listas, puede probar el objeto diccionario y ver cómo funciona.

```
<código
root@kali:~# pitón3
Pitón 3.4.4.4 (predeterminado, Jan  5 2016,
15:35:18)
GCC 5.3.1 20160101] en linux
Para más información, escriba "ayuda", "copyright",
"créditos" o "licencia".
>>> x = {'uno':1, 'dos':2, 'tres':3}
>>> tipo(x)
clase <class 'dict'>
```

```
>>> y = dic(cuatro = 4, cinco = 5, seis = 6)
>>> tipo(y)
clase <class 'dict'>
>>> z = dic(siete = 7, ocho = 8, nueve = 9, **x,
**y)
      Archivo "<stdin>",
      línea 1 z = dic(seite ocho = 8, nuev = 9, **x,
**y)      = 7
                                         ^
SyntaxError: inválidosintaxis
>>> z = dict(seven = 7, siete = 8, nuev = 9, **x)
>>> tipo(z)
clase <class 'dict'>
>>> imprimir(z)
{'ocho': 8, 'dos': 2, 'nueve': 9, 'uno': 1,
'siete': 7, 'tres': 3}
>>> para i en z:print(i)
...
ocho
dos
nueve
uno
uno
siete
tres
>>> para la tecla, valor en z.items():print(key,
value)
...
ocho 8
dos 2
nueve 9
uno 1
siete 7
tres 3
>>> para la tecla, valor en z.items():
...     si la tecla == dos:
...         print(valor)
...
Traceback (última llamada): Archivo
"<stdin>", línea 2, en <módulo>
```

```
NameError: nombre 'dos' no está definido
>>> z.pop()
Traceback (última llamada): Archivo
  "<stdin>", línea 1, en <módulo>
TypeError: pop espera al menos 1 argumentos, tiene
0
>>> z.pop(tres)
Traceback (última llamada): Archivo
  "<stdin>", línea 1, en <módulo>
NameError: nombre 'tres' no está definido
>>> z.pop('tres')
3
>>> para i en z:print(i)
...
ocho
dos
nueve
uno
uno
siete
>>> para la tecla, valor en z.items():
...     si la tecla == 'nueve':
...         print(valor)
...
9
>>>
</código>
```

Cuanto más tiempo pases con tuplas, listas y diccionarios, más aprenderás sobre Python. Hay un montón de funciones incorporadas y puedes usarlas muy fácilmente para sacar más provecho de tu código. Otro concepto clave del diccionario es el par "key=>value". A medida que progresas y aprendes más idiomas junto con Python, te darás cuenta de que cada idioma utiliza este concepto, llevándolo más allá para resolver problemas importantes. Los marcos de trabajo web, en particular, utilizan este concepto muy intensamente.

## 19. Base de datos

Sanjib Sinha 

(1) Howrah, Bengala Occidental, India

---

Las operaciones de base de datos en Python son bastante simples. Por la pequeña cantidad de trabajo, el SQLite3 incorporado es bastante competitivo. Puede mantenerlo fácilmente creándolo, recuperándolo, actualizándolo y borrándolo.

El término básico es "CRUD..." "C" significa crear, "R" significa recuperar, "U" significa actualizar y "D" significa borrar. Con cualquier base de datos, generalmente se realizan estas acciones.

---

### Empecemos con SQLite3 .

Hay una gran biblioteca dentro de la casa de los Python. Todas las funciones y propiedades de SQLite3 se almacenan allí, por lo que puede importarlas fácilmente y utilizarlas para su proyecto. Considere este código:

```
/usr/bin/python3
import sqlite3
def main():
    db = sqlite3.connect('test.db')
    db.row_factory = sqlite3.Row
    db.execute('drop table if exists test1')
    db.execute('create table test1 (texto t1,
    i1
int)')
    db.execute('insertar en valores de pruebal (t1,
    i1)
(?, ?, ('Babu', 1))
    db.execute('insertar en valores de pruebal (t1,
    i1)
```

```

(?, ?, ('Mana', 2))
    db.execute('inserta en prueb (t1, i1) valore
    r           a1           s
(?, ?, ('Bappa', 3))
    db.execute('inserta en prueb (t1, i1) valore
    r           a1           s
(?, ?, ('Babua', 4))
    db.execute('inserta en prueb (t1, i1) valore
    r           a1           s
(?, ?, ('Anju', 5))
    db.execute('inserta en prueb (t1, i1) valore
    r           a1           s
(?, ?, ('Patai', 6))
    db.execute('inserta en prueb (t1, i1) valore
    r           a1           s
(?, ?, ('GasaBuddhu', 7))
    db.execute('insertar en el test1 (t1, i1)
valores (?, ?, ('Tapas', 8))
    db.commit()
DatabaseRead = db.execute('select * from test
order by i1')
para la linea en DatabaseRead:
    # imprimir(dic(fila))
    print(fila['t1'])
    # print(row['t1'], row['i1'])
    # print(type(row))
si nombre __ == " main ":main()
</código>

```

Si ejecuta este código, verá una lista de nombres que acabo de añadir. Como veis, nos hemos conectado con una base de datos, "test". A continuación añadimos una tabla con dos columnas. La primera columna es el número entero de identificación y guardamos el número de identificación de cada nombre dentro de ella. La segunda columna es el marcador de posición del texto. Allí guardamos algunos nombres.

Puedes escribir el mismo código y probarlo. Te dará el mismo resultado. Una vez que ejecute el código, encontrará que se ha creado un archivo "test.db" dentro de su proyecto.

## MySQL para Big Project

SQLite3 es bueno para una pequeña cantidad de trabajo. Pero para un gran

proyecto, es mejor optar por una base de datos como MySQL. Para trabajar con MySQL en Python3 es necesario descargar e instalar el conector MySQL. La parte de descarga e instalación

es bastante fácil.

En Python2\* puedes importar por defecto MySQL Connector. Pero para Python3, necesitas descargar el archivo. Abra <https://python.org> y busca MySQL Connector. Descargue el archivo y ejecute "setup.py".

Una vez que descargue e instale el módulo MySQL Connector es bastante simple y fácil conectarse a cualquier base de datos MySQL.

Consideré este código donde simplemente nos conectamos a una base de datos MySQL y tenemos una impresión "conectada".

Si MySQL o cualquier operación de base de datos es completamente nueva para usted, es mejor aprender sobre las operaciones simples de la base de datos y el lenguaje de consulta de la base de datos. En Windows o Linux, la instalación de PHPMyAdmin es muy fácil. Simplemente instálelo y no necesita escribir todo el código SQL para construir una base de datos y todas las tablas.

Supongamos que tenemos una base de datos llamada "python-mysql". En esa base de datos tenemos algunas tablas. Ahora vamos a conectarnos a esa base de datos primero.

```
<Importar conector
mysql./usr/bin/python3
desde la importación de
mysql.connector Error def
ConnectionTest():
    #### conectando con la base de datos MySQL ####
    Intentalo:
        #### puedes usar un objeto del diccionario
    o puedes conectarte directamente ####
        #### using a dictioanry connection object
    ####
        kwargs = dic(host = 'localhost', base de
            datos
= python_mysql', usuario = 'root', contraseña =
            'pass') conn =
            mysql.connector.connect(**kwargs) ####
            conectando directamente ####
            conexión = mysql.connector.connect(host =
'localhost',
            se = python_mysql',
            base de
            datos
            'root',
            usuario
```

**io = passwo**

```

rd = 'pass')
    si conn.is_connected():
        print("Connected from 'conn' object")
    except Error as e:
        print(e)
finalmente:
    conexión.close()
si nombre == " principal ":
    Prueba de conexión()
</código>

```

Nos dará una impresión "Connected from a conn object". Significa que se ha configurado la conexión a la base de datos. Ahora es el momento de recuperar el valor de la tabla.

En esta base de datos tenemos dos tablas. Uno es de "autores" y el otro de "libros". La clase MySQL Connector tiene todas las funciones necesarias para realizar cualquier tarea en esas tablas. Puedes traer todos los discos. Usted puede decidir cuántos libros o cuántos autores le gustaría buscar. El siguiente código le muestra ambos. Pero algunas partes han sido comentadas.

Para probar este código es necesario tener una base de datos primero. Dígale "python-mysql". A continuación se necesitan dos tablas llamadas "autores" y "libros". También tienes que llenar esas tablas. Siempre es mejor buscar en línea y descargar una base de datos y tablas MySQL ya hechas. Están disponibles. Se recomienda encarecidamente que busque MySQL Connector y vea lo que encuentra.

En el siguiente código, por favor revise también las secciones comentadas. Eso dice mucho sobre cómo puede recuperar sus registros y mostrarlos al mundo.

```

<Importar conector
mysql ./usr/bin/python3
desde la importación de
mysql.connector.Error def
RetrieveValues():
    Inténtalo:
        kwargs = dic(host = 'localhost', base de
                    datos
= python_mysql', usuario = 'root', contraseña =
                    'pass') conn =
                    mysql.connector.connect(**kwargs)
##### le muestra cómo consultar datos desde un
MySQL

```

```
en Python usando el conector MySQL/Python API # como
    fetchone() , fetchmany() , y
fetchall() #####
    si conn.is_connected():
        cursores =
            conn.cursor()
        cursores.execute('SELECCIONAR * DESDE
de los
autores")      # row = cursores.fetchone()
                # salida (1,'Bel y el Dragón',
'123828863494')
                #####
                # ahora intentamos conseguir
                todos los libros # row =
                    cursores.fetchall()
                    # print(type(row))
                    # output <class <class 'list'>, para
                    que podamos usar
para
bucle          # para los libros en fila:
                # imprimir (libros)
                # Nos dará una lista de todos los
                libros #
                #### now we give the size of how many
libros que queremos conseguir
                # HowManyBooks = 8
                # row =
                    cursores.fetchmany(HowManyBooks) # para
                    libros en row:
                    # imprimir (libros)
                    # tenemos la salida de 8 libros
                    row =
                        cursores.fetchall() para
                        los libros en la fila:
                        print(books)
exceptúe Error as e:
    print(e)
finalmente:
    conn.close()
si nombre == " principal ":
    RecuperarValores()
```

</código>

Hemos utilizado el método de prueba y error para que si la conexión falla, pueda

no mostrar un feo mensaje de "Error" en su proyecto. Segundo, este método es bastante directo. También puede utilizar un archivo de configuración para hacer lo mismo.

Es muy recomendable utilizar un archivo de configuración (se dice "archivo de configuración"). El archivo de configuración tiene todo lo necesario para conectarse a la base de datos.

Podemos escribir en el archivo de configuración así y guardarlo como "mysql\_config.ini".

```
< código
[mysql]
host = localhost
database =
YourDatabaseName usuario =
root
contraseña = pass
</ código>
```

Veamos cómo este archivo ".ini" puede ser analizado a través de nuestro código Python. Guardamos este archivo como "MySQL\_Connector.py".

```
< código
#!/usr/bin/python3
desde la importación de configparser
ConfigParser def
ReadingMySQLConfig(filemame =
'mysql_config.ini', sección = 'mysql'):
    parser = ConfigParser()
    parser.read(filemame)
    db = dic()
    if parser.has_section(section):
        items =
            parser.items(section) for
            item in items:
                db[item[0]] = item[1]
    de los demás:
        raise Exception('{0} not found in the
{1} archivo'.format(section,
        filemame))) devolver db
</ código>
```

Verá que hemos importado los módulos necesarios para analizar el archivo

de configuración y finalmente hemos utilizado ese archivo de configuración para conectarnos a la base de datos. Y en el código anterior de "MySQL\_Connector.py" tenemos

incluye el archivo "mysql\_config.ini" en esta definición de línea  
**ReadingMySQLConfig(filename = 'mysql\_config.ini', section = 'mysql'):**  
-...como argumento.

A continuación se muestra cómo podemos usar este archivo de configuración para probar nuestra conexión.

```
<código
#!/usr/bin/python3
desde mysql.connector importar MySQLConnection,
Error desde MySQL_Connector.mysql_config importar
LecturaMiSQLCconfigur
ar def Connect():
    kwargs = ReadingMySQLConfig()
    MyConnection = MySQLConnection(**kwargs)
    try:
        si
            MyConnection.is_connected(
                ): print("Connected")
    excepto Error
        como e:
            print(e)
    finalmente:
        MiConexión.close()
si nombre == " principal ":
    Connect()
</código>
```

Ahora hemos desacoplado más nuestro código. Podemos dividirlo en pequeños segmentos para que nuestro código de conexión parezca extremadamente pequeño y organizado. Pero siempre puede conectarse a su base de datos MySQL como se muestra a continuación.

```
<código
#!/usr/bin/python3
# -*- codificación: utf-8 -*-
importar el conector mysql.connector
desde la importación de
mysql.connector Error def
connect():
    "Conectarse a la base de datos MySQL"
    Inténtalo:
        conn =
```

```

mysql.connector.connect(host='localhost',
                        database='Yo
urDatabase',
                        user='root',
                        password='Yo
urPassword')
    si conn.is_connected():
        print('Connected to MySQL database')
excepto Error as e:
    print(e)
finalmente:
    conn.close()
si el nombre == 'principal__':
    connect()
</código>

```

Ahora ha llegado el momento de recuperar los registros de la base de datos. Podemos conectarnos a la base de datos. Ahora, no debería haber ningún problema para obtener registros de las tablas de la base de datos. Tenemos dos métodos incorporados en nuestra biblioteca Python. Los métodos son "fetchmany()" y "fetchall()". El primer método, "fetchmany()", le da la libertad de decidir cuántas filas va a buscar. Veamos el código:

```

//consulta con fetchmany()
<código
#!/usr/bin/python3
desde mysql.connector importar MySQLConnection,
Error desde Databases.python_mysql_dbconfig
importar
read_db_config
def iter_row(cursor,
size=10): mientras que
True:
    row = cursor.fetchmany(size)
    si no son filas:
        romperse
    para la linea
        en líneas:
            linea de
            rendimiento
def
query_with_fetchmany()

```

```
: try:
```

```

        dbconfig = read_db_config()
        conn = MySQLConnection(**dbconfig)
        cursor = conn.cursor()
        cursor.execute("SELECT * FROM
EMPLOYEE")

# EMPLEADO es el nombre de la tabla
        para la fila en iter_row(cursor,
        10): print(row)

excepto Error
        como e:
        print(e)
finalmente:
        cursor.close()
        conn.close()

si el nombre == ' principal__':
        query_with_fetchmany()
</código>

```

El método "fetchall()" trae todos los registros de una tabla.

```

<código
#!/usr/bin/python3
desde mysql.connector importar MySQLConnection,
Error desde Databases.python_mysql_dbconfig
importar
read_db_config
def
    query_with_fetchall()
: try:
        dbconfig = read_db_config()
        conn = MySQLConnection(**dbconfig)
        cursor = conn.cursor()
        cursor.execute("SELECT * FROM
EMPLOYEE") filas = cursor.fetchall()
print('Total Fila(s) : ', cursor.rowcount)
para fila en filas:
        print("Nombre = ", fila[0])
        print("Segundo Nombre = ", fila[1])
        print("Edad=", fila[2])
        print("Sexo = ", fila[3])
        print("Salario =",
```

```
fila[4]) excepto Error como e:
```

```

        print(e)
finalmente:
    cursor.close()
    conn.close()
si el nombre == 'principal__':
    query_with_fetchall()
</código>

```

Usted ve cómo podemos obtener los registros como nuestros requisitos. Ahora intentemos probar el proceso de inserción. En nuestra aplicación CRUD, la primera "C" significa "Crear". Aquí la palabra "Create" no significa nada más que la inserción de nuevos registros. A través de MySQL Connector es bastante simple. Todo lo que necesitas es que la conexión esté encendida. Después de eso necesita insertar sus registros.

Aquí está el código. Tenemos una tabla "Book" en nuestra base de datos y vamos a insertar dos registros en ella. Uno es el título del libro y el otro es el código ISBN del libro.

```

<código
#!/usr/bin/python3
desde mysql.connector importar MySQLConnection,
Error desde MySQL_Connector.mysql_config importar
LecturaMySQLConfigurar
def InsertBooks(libros):
    consulta = "INSERTAR EN libros(título, isbn)
VALORES(%s, %s)"
    Inténtalo:
        kwargs = ReadingMySQLConfig()
        MyConnection = MySQLConnection(**kwargs)
        si MyConnection.is_connected():
            cursor = MyConnection.cursor()
            cursor.executemany(query, books)
            MyConnection.commit()
    excepto Error
        como e:
            print(e)
    finalmente:
        MyConnection.close()
def main():
    books = [("TestBook", 1236547890)]

```

```

InsertarLibros impresos
("Insertar un libro")
si nombre _ == " principal ":
    main()
</código>

```

Hemos insertado con éxito un título de libro y un código ISBN. El siguiente proceso será la actualización del título y del código ISBN. Eso también es muy fácil. Todo lo que necesitas es el ID único del libro. Una vez que haya proporcionado el ID único del libro, podrá actualizarlo fácilmente.

```

<ocde> <ocde
#!/usr/bin/python3
desde mysql.connector importar MySQLConnection,
Error desde MySQL_Connector.mysql_config importar
LecturaMiSQLConfigurar
def UpdateBooks(book_id, title):
    kwargs =
        ReadingMySQLConfig() data =
        (title, book_id)
    query = "UPDATE books SET title = %s WHERE id =
%s"
    Intentalo:
        MyConnection = MySQLConnection(**kwargs)
        cursor = MyConnection.cursor()
        cursor.execute(query, data)
        MyConnection.commit()
    excepto Error
        como e:
            print(e)
    finalmente:
        MiConexión.close()
def main():
    para id en el rango
        (1, 25): si id ==
            3:
                Impresión de UpdateBooks(id, "I Have A
                Dream") ("One book has been updated")
            elif id == 4:
                Impresión de UpdateBooks(id, "Laravel 5
                Unfolded") ("Un libro ha sido

```

**actualizado")**

```
    elif id == 5:
        Imprimir UpdateBooks(id, "Play With
        Python") ("Un libro ha sido
        actualizado")
    si nombre _ == " principal ":
        main()
</código>
```

Hemos actualizado con éxito tres libros que tienen identificaciones únicas de 3, 4 y 5, respectivamente. Finalmente veremos cómo podemos borrar un registro.

Para borrar un registro, de nuevo necesita el ID único .

```
<código
#!/usr/bin/python3
desde mysql.connector importar MySQLConnection,
Error desde MySQL_Connector.mysql_config importar
LecturaMySQLConfigurar
def DeleteBooks(book_id):
    kwargs = ReadingMySQLConfig()
    query = "DELETE FROM books WHERE id = %s"
    try:
        MyConnection = MySQLConnection(**kwargs)
        cursor = MyConnection.cursor()
        cursor.execute(query, (book_id,))
        MyConnection.commit()
    excepto Error:
        como e:
            print(e)
    finalmente:
        MyConnection.close()
def main():
    id = 87
    BorrarLibros(id)
    print("Deleted", id, "number of book from books")
    si nombre _ == " principal ":
        main()
</código>
```

En este código, esta línea -"cursor.execute(query, (book\_id,))"- es extremadamente

importante. Probablemente notará que hemos utilizado un separador "," después del "book\_id". Es su tarea averiguar por qué se ha utilizado este separador de comas. La única pista es que está relacionado con "tuples" o "listas". Es su tarea averiguar cuál es la verdadera razón.

Como toda versión moderna de bases de datos relacionales, MySQL también le permite mantener un objeto binario grande dentro de él. Normalmente cuando escribes números o cadenas no ocupan mucho espacio. ¿Pero qué hay de las imágenes? Supongamos que tenemos una tabla de autores donde necesitamos guardar las imágenes para los autores. También es posible que queramos mantener las fotos de la portada de los libros en nuestra mesa de libros.

Normalmente esta imagen o cualquier objeto binario grande se llama, en pocas palabras, "BLOB".

. Vamos a actualizar nuestra tabla de autores con una imagen y ver cómo funciona.

```
<código
#!/usr/bin/python3
desde mysql.connector importar MySQLConnection,
Error desde MySQL_Connector.mysql_config importar
LecturaMySQLConfigurar
def ReadFile (nombre de archivo):
    con open(nombre de archivo,
        'rb') como f: imágenes =
        f.read()
    devolver imágenes
def UpdateImage(author_id, nombre de
    archivo): kwargs =
    ReadingMySQLConfig() data =
    ReadFile(nombre de archivo)
    query = "UPDATE authors SET photo = %s WHERE id
= %s"
    args = (datos,
    author_id) intentar:
        MyConnection = MySQLConnection(**kwargs)
        cursor = MyConnection.cursor()
        cursor.execute(query, args)
        MyConnection.commit()
    excepto Error
        como e:
            print(e)
finalmente:
```

```
M      iConexión.close()  
def main():
```

```

id = 47
UpdateImage(id, "/home/hagudu/Pictures/ss.jpg")
print("Image of author ID", id, "has been
actualizado.")
si nombre _ == " principal ":
    main()
</código>

```

El código es bastante simple. Por lo menos en esta etapa usted debe encontrarlo simple.

Los pasos son los siguientes

1. Leer el archivo con la ayuda de la palabra clave "con" y guardarla en una variable y devolverlo. Pasamos el parámetro a través de la función. Vea la primera función: "ReadFile(nombre de archivo)".
2. La segunda función es crucial porque pasa el mismo nombre de archivo que uno de los parámetros. También se conecta a la base de datos y confirma. Véase la segunda función: "UpdateImage(author\_id, nombre de archivo)".
3. Finalmente, llamamos a la segunda función y pasamos la ruta del archivo de imagen como argumento para que nuestro código Python llegue hasta allí y recupere la imagen abriéndola y finalmente ingresando a la base de datos.

Ahora vamos a recuperar una imagen de la base de datos y escribirla en nuestro disco local. En el código anterior hemos leído el archivo. Ahora es el momento de escribir el archivo en nuestro disco. El código es casi similar excepto por algunos cambios.

```

<código
#!/usr/bin/python3
desde mysql.connector importar MySQLConnection,
Error desde MySQL_Connector.mysql_config importar
LecturaMySQLConfigurar
def WriteFile(datos, nombre del archivo):
    con open(nombre de archivo, 'wb') como
        archivos: files.write(data)
def ReadImage(author_id, nombre de
    archivo): kwargs =
        ReadingMySQLConfig()

```

```
query = 'SELECCIONAR foto DE LOS autores  
DONDE id ='SELECCIONAR foto
```

```
%$ '

Intentalo:
    MyConnection = 
        MySQLConnection(**kwargs) cursor = 
            MyConnection.cursor()
            cursor.execute(query, (author_id,))
            photo = cursor.fetchone()[0]
            WriteFile(photo, filename)
excepto Error
    como e:
        print(e)
finalmente:
    MiConexión.close()

def main():
    id = 47
    ReadImage(id, "/home/hagudu/Pictures/ss1.jpg")
si nombre __ == " principal ":
    main()
</código>
```

## 20. Módulo

Sanjib Sinha 

(1) Howrah, Bengala Occidental, India

---

En Python cuando dejas el shell o terminal o el intérprete de Python, el script se pierde. Después de todo, usted no escribe programas que perder al final del día. Puede ser un simple programa de cálculo. Pero quieres usarlo de nuevo. Otra cosa importante es que usted necesita utilizar su un código en su otro código. Es posible que también quieras usar el código de otras personas.

Para resolver este dilema, entra en juego el concepto de "módulo".

Usted escribe un simple programa de cálculo y guarda el archivo como "cal.py". Si estás en el directorio raíz de tu proyecto, puedes usar fácilmente tu calculadora en tu otro programa. Una vez que escribes un código Python y lo guardas con un nombre, ese nombre se convierte en un módulo.

En este caso, "cal" se convierte en un módulo. Ahora puede "importar" ese módulo "cal" a cualquier otro código o módulo. En la gran biblioteca de Python hay toneladas de módulos. Siempre puede importarlos y utilizarlos. Considere el siguiente código. En este código hemos importado tres módulos. El primero es "sys" o módulo específico del sistema. El segundo es "os" o módulo específico del sistema operativo y el tercero es "urllib", que significa una biblioteca específica de URL. Notará que escribimos "urllib.request". La notación "punto" significa que en realidad llamamos a algo llamado "petición" de la biblioteca de URLs de Python. En realidad, la arquitectura web depende principalmente de dos cosas: solicitar y responder... Aquí vamos a solicitar algo de una URL.

```
< código
#!/usr/bin/python3
importar sys, os,
urllib.request def main():
```

```

        print("Esta es la versión de Python : {}.{}.{}".
{}".format(*sys.version_info))
        # os module
        print(os.name)
        print(os.getenv('PATH'))
        print(os.getcwd())

    módulo #urllib
    página = urllib.request.urlopen(
http://arshinagar.in/')
    para la línea en la página:
        print(str(line, encoding='utf-8') , end=' ')
    si nombre _ == " principal ":
        main()
</código>

```

Verás que en la primera parte del código hemos utilizado el módulo "sys" y queríamos saber la versión de Python que nuestro sistema está utilizando. La segunda parte se refiere al sistema operativo. Nos da el nombre, el camino y muchas otras cosas. Y en la última parte estamos solicitando una página web.

Veamos primero la salida en una distribución Debian de Linux como Ubuntu. La primera línea es la versión y la segunda línea es sobre el sistema operativo, que es "posix". La tercera línea es la ruta del entorno y la cuarta línea es la ruta real donde se almacena este archivo.

A partir de la quinta línea se ve que el "urllib.request" comienza a funcionar y obtiene toda la página de índice de un sitio web. He utilizado el sitio web de mi amigo. No imprimo toda la salida HTML, ya que ocuparía mucho espacio. Revise cada línea y vea cómo funcionan los diferentes módulos.

```

<cotización en bloque>.
Esta es la versión de Python
: 3.4.3 posix
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/
sbin:/bin:/bin:/usr/games:/usr/local/games
/home/hagudu/PycharmProjects/FirstPythonProject/modul
es ules
DOCTYPE html> <!DOCTYPE html>
<html lang="es">
Cabeza <cabeza>
```

```

<meta charset="UTF-8" />
<name="viewport" content="width=device-width"
/>
<name="viewport" content="initial-scale=1.0"
/>
<name="HandheldFriendly" content="true"/>
<link rel="perfil" href=" http://gmpg.org/xfn/11"
/>
<"Enlace rel="pingback"
href=" http://www.arshinagar.in/xmlrpc.php" />
<título>Arshinagar - Sólo otro sitio de WordPress
</título>
<Enlace rel="alternativo" type="application/rss+xml"
title="Arshinagar &raquo; Feed" href="
http://www.arshinagar.in/feed/" />
<link rel="alternate" type="application/rss+xml"
title="Arshinagar &raquo; Comments Feed" href="
http://www.arshinagar.in/comments/feed/" />
//los detalles se eliminan por brevedad
Proceso finalizado con el código de salida 0
</blockquote>
```

Ahora podemos probar este mismo código en Windows y comparar la salida.

```

<cotización en bloque>.
Esta es la versión de Python
: 3.4.4 nt
C: Windowsystem 32: Windowsystem 32: Windowsystem 32:
Wbem, C: Program Files: Microsoft SQL Server 90:Tools
Binn.
D:\pthon-files-fromwindows
</blockquote>
```

En esta salida se ve que la versión de Python ha sido cambiada. El sistema operativo ya no es "posix". Es "nt" ahora. La ruta del entorno y la ruta del archivo también son polos opuestos. Eliminé la salida del módulo "urllib.request" para mayor concisión.

Podemos ver más ejemplos de módulos aquí.

```

< código
#!/usr/bin/python3
importar sys, os, urllib.request, random,
datetime def main():
    print("Esta es la versión de Python : {}.{}}.
{}".format(*sys.version_info))

    # random module
    print(random.randint(1,
1000)) x = list(range(25))
    print(x)
    random.shuffle(x)
    print(x)
    random.shuffle(x)
    print(x)
    random.shuffle(x)
    print(x)
    PresentTime =
    datetime.datetime.datetime.now()
    print(PresentTime) print(PresentTime.year,
PresentTime.month,
PresentTime.day, PresentTime.hour,
PresentTime.minute, PresentTime.second,
PresentTime.microsecond)

    si nombre __ == " principal ":
        main()
</ código>

```

En este código añadimos dos módulos más. Son "aleatorios" y "fechados". Obtenemos el resultado a continuación para ver cómo funcionan.

```

</blockquote>
Esta es la versión de Python
: 3.4.3 366
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
15, 16, 17, 18, 19, 20, 21, 22, 23, 24]
[23, 6, 22, 3, 7, 19, 10, 16, 8, 12, 15, 21, 11,
17, 9, 13, 4, 14, 24, 18, 0, 2, 1, 20, 5]
[0, 8, 21, 5, 13, 3, 2, 18, 24, 12, 4, 19, 14, 17,
20, 10, 11, 22, 15, 9, 6, 23, 1, 7, 16]

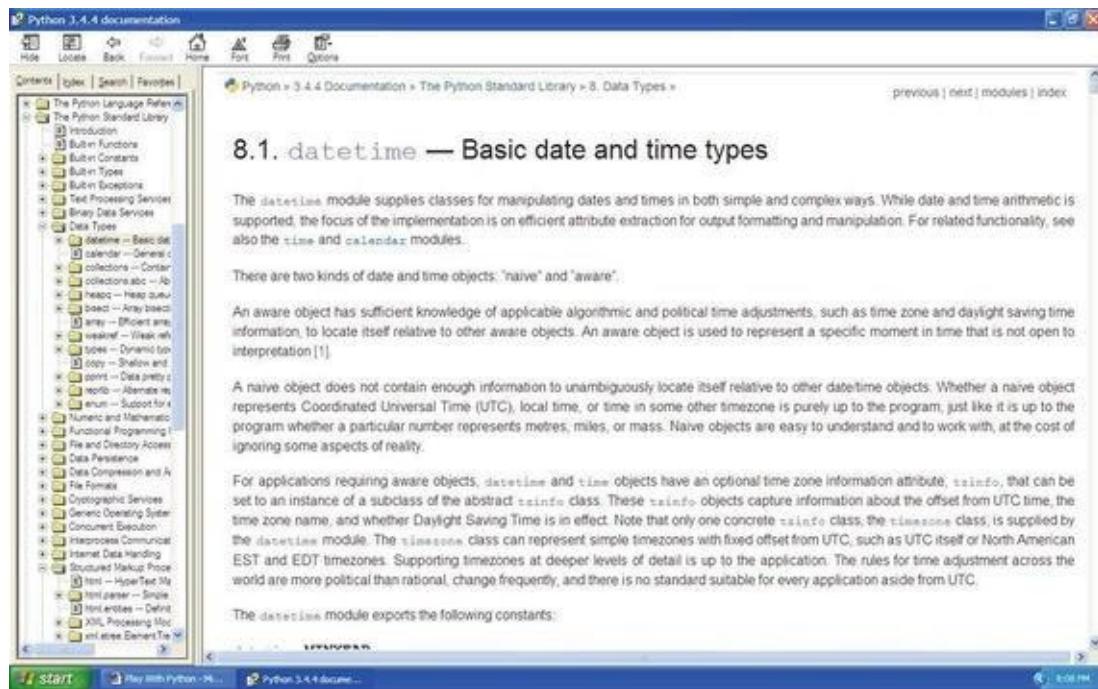
```

```

[11, 6, 23, 14, 9, 7, 3, 5, 15, 2, 19, 0, 16, 24,
21, 12, 4, 13, 22, 20, 10, 8, 1, 17, 18]
//Aquí está la salida del módulo de fecha y hora
2016-03-23 08:34:37.253888
2016 3 23 8 34 37 253888
</blockquote>

```

Cada vez que ejecute el código, obtendrá un nuevo número, ya que el módulo "aleatorio" siempre produce nuevos números. Para obtener más ideas, necesita ir a través de la Biblioteca Python Standard en el sitio web oficial de Python o descargar la documentación de Python 3.4.4. Está disponible en muchos tipos de archivos, incluyendo archivos de texto simple o PDF. La página del módulo "datetime" en Python Standard Library en la documentación tiene el siguiente aspecto:



**Figura 20-1.** Biblioteca Python Standard

Ahora puedes volver a tus viejos códigos y verlos de nuevo bajo una nueva luz.

Ahora comprenderá fácilmente por qué hemos utilizado el módulo MySQL Connector o el módulo Configuration Parser.

Sanjib Sinha 2017

Sanjib Sinha, Comenzando el Hacking Ético con Python, DOI 10.1007/978-1-4842-2541-7\_21

---

## 21. Depuración, Módulo Unitest

Sanjib Sinha 

(1) Howrah, Bengala Occidental, India

---

Ahora has progresado mucho. En el proceso de codificación usted debe haber encontrado o visto muchos tipos de errores. Es bastante obvio. Los programadores experimentados también cometan errores. También has aprendido a detectar tus errores. Pero la situación puede llegar cuando necesite actualizar su código. Podría suceder. Necesitas modificar o añadir algunas líneas en tu código. Puede funcionar o puede fallar. En sus nuevas líneas de código podría haber errores "sintácticos". Puede haber errores de "tiempo de ejecución". Normalmente, el intérprete de Python intenta guiarle en estos casos. Por lo general, indica dónde se ha producido el error. Pero no siempre.

En estos casos, el módulo "unittest" viene en su ayuda.

En la biblioteca estándar de Python se obtiene mucha información sobre este módulo. También puede buscar en Internet sobre la herramienta "nariz", que hace algo similar. El concepto básico es, usted tiene un repositorio de código en alguna parte y tiene un programa de pruebas de unidad separado. Es una prueba automatizada.

Supongamos que tenemos una carpeta llamada "MyTest/BrainAndSoul". Dentro de esta carpeta tenemos un archivo Python llamado "saytimedate.py". Es un archivo muy simple que nos dirá la versión de Python y la fecha y hora actual. Para obtener esa salida, necesitamos dos módulos: "sys" y "datetime". Tenemos dos métodos para obtener esos resultados. Para obtener la salida, todo lo que tenemos que hacer es llamarlos bajo la función "main()". Hacemos exactamente eso.

Al mismo tiempo tenemos dos métodos separados que comienzan con la palabra "test". Los métodos son "test\_PyVar()" y "test\_main()".

```
/usr/bin/python3 #
coding=utf-8
```

```

import sys,
datetime def
PyVer():
    print("Esta es la versión de Python : {}.{}.{}".
format(*sys.version_info))) def
PyTime():
    PresentTime =
        datetime.datetime.datetime.now()
    print(PresentTime) print(PresentTime.year,
    PresentTime.month,
PresentTime.day, PresentTime.hour,
                PresentTime.minute, PresentTime.second,
PresentTime.microsecond)
    #imprimir(obj)
def main():
    PyVer()
    PyTime()
def
    test_Pyvar()
    : PyVer()
def
    test_Main()
    : PyTime()
si nombre __ == " principal ":
    main()
< código

```

Cuando ejecuta este código, su función main() llama a los dos métodos definidos dentro de él. Y el resultado a continuación es lo que se espera.

```

</blockquote>
Esta es la versión de Python
: 3.4.2 2016-04-22
23:30:30.435691
2016 4 22 23 30 30 435691
</blockquote>

```

Ahora, en una carpeta completamente separada, nos gustaría ejecutar el módulo "unittest" y ver si este código pasa o falla. Puesto que ya hemos ejecutado el código y obtenido una salida exitosa, podemos decir con seguridad que este código pasará la prueba.

El nombre de nuestro código de prueba de unidad es "TestUnitTest.py" y el código tiene el siguiente aspecto:

```

/usr/bin/python3 #
coding=utf-8
importar MyProject.BrainAndSoul.saytimedate
importar unittest
clase SayTiemDate(unittest.TestCase):
    def setUP(self):
        de paso
    def test_Version(self):
self.assertEqual(MyProject.BrainAndSoul.saytimedate.Py
Ver(),
MyProject.BrainAndSoul.saytimedate.test_Pyvar())

    def test_Time(self):
self.assertEqual(MyProject.BrainAndSoul.saytimedate.ma
in(), MyProject.BrainAndSoul.saytimedate.test_Main())

si nombre == " principal ":
    unittest.main()
</código>

```

¿Qué dice este código? Como puede ver, hay dos métodos: "test\_Time()" y "test\_Version()". No hemos pasado ninguna discusión. Ambos métodos llaman a un método por defecto del módulo "unittest". Y eso es "assertEqual()".

A través de este método hemos pasado dos métodos que hemos definido anteriormente en la carpeta "MyTest/BrainAndSoul". Dentro de esa carpeta tenemos un archivo Python llamado "saytimedate.py". Ahora estamos comparando dos métodos a través de nuestro módulo "unittest".

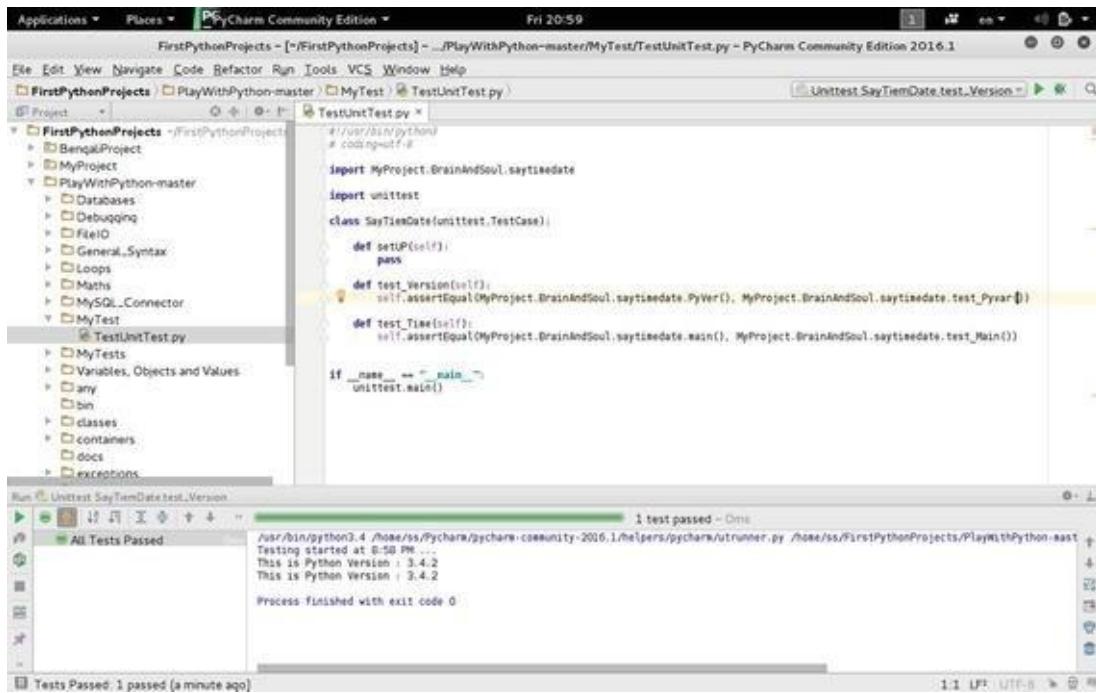
Por último, da una salida agradable como esta si todo funciona correctamente.

```

<cotización en bloque>.
Las pruebas comenzaron a las 8:58
PM... Esta es la versión de
Python : 3.4.2 Esta es la versión
de Python : 3.4.2 El proceso
terminó con el código de salida 0
</blockquoteo>

```

Cuando ejecuta el código, se parece a la siguiente imagen en su IDE "PyCharm".



**Figura 21-1.** Prueba de unidad en PyCharm IDE

Si ejecutamos ese código de nuevo podemos obtener una salida como esta:

<cotización en bloque>>.

**Esta es la versión de Python**

: 3.4.2 2016-04-23

05:47:23.608853

2016 4 23 5 47 23 608853

2016-04-23 05:47:23.608951

2016 4 23 5 47 23 608951

**Esta es la versión de Python**

: 3.4.2 Esta es la versión de

Python : 3.4.2

..

Hizo 2 pruebas en

0.001s OK

**Proceso finalizado con el código de salida 0**

</blockquote>

Ahora, para propósitos de prueba, cambiamos nuestro código fuente y hacemos algunos

errores intencionados para ver si nuestro módulo "unittest" falla o no.

Si hay algún error, la salida cambiará y dará un mensaje de error como este:

```
<cotización en bloque>.
Esta es la versión de Python
: 3.4.2 2016-04-23
05:51:45.994547
2016 4 23 5 51 45 994547
Esta es la versión de Python
: 3.4.2 Esta es la versión de
Python : 3.4.2 E.
=====
=====
ERROR: test_Time ( main.SayTiemDate)
-----
-----
Traceback (última llamada):
Archivo
"/home/ss/FirstPythonProjects/PlayWithPython-
master/MyTest/TestUnitTest.py", línea 17, en
test_Time
    self.assertEqual(MyProject.BrainAndSoul.saytime
date.main(),
MyProject.BrainAndSoul.saytimedate.test_Main())
    Archivo
"/home/ss/FirstPythonProjects/MyProject/BrainAndSoul/s
aytimedate.py", línea 20, en principal
        PyTime()
        Archivo
"/home/ss/FirstPythonProjects/MyProject/BrainAndSoul/s
aytimedate.py", línea 15, en PyTime
        print(obj)
NameError: nombre 'obj' no está definido
-----
-----
Realizó 2 pruebas en
0.001s FALLANDO
(errores=1)
```

**Proceso finalizado con el código de salida 1**  
</blockquoteo>

Ahora puede intentar ejecutar más módulos de pruebas unitarias. Aquí hay otro ejemplo en el que la prueba es exitosa.

## 22. Enchufe y conexión en red

Sanjib Sinha 

(1) Howrah, Bengala Occidental, India

---

Este capítulo es una especie de introducción a los conceptos avanzados de Python. Ya que este es el capítulo final, me gustaría decirles a dónde pueden ir a partir de aquí.

Puedes construir aplicaciones web con la ayuda de Python o puedes hacer algunas cosas de redes de seguridad. Finalmente, como yo, puedes elegir el interesante camino del hacking ético. Todas estas cosas y más que puedes hacer a través de Python.

Veamos cómo podemos aplicar nuestros conocimientos básicos de Python en socket y networking. Escriba este código en su IDE y vea qué salida obtiene.

```
<código>
# coding=utf-8
importar socket print(socket.gethostname()
www.mesanjib.wordpress.
com"))
print(socket.gethostname(" www.sanjib.pythonanywhe
re.com"))

</código>
```

La salida es así en mi máquina. Puede probar cualquier otro sitio web para obtener su dirección. Esta es la punta del iceberg. Hay muchas cosas dentro. Es mejor que veas todo lo que hay dentro que yo te lo diga, ya que siento que deberías concentrarte en intentar escribir cada vez más los conceptos básicos de Python.

<cotización en bloque>>.

192.0.78.12

**50.19.109.98**

**Proceso finalizado con el código de salida 0**  
</blockquote>

En el estudio posterior de la relación entre el hacking ético y Python 3, encontrará que estos conceptos de socket y networking son extremadamente útiles.

Sigamos con la tercera parte del libro, donde aprenderemos sobre el anonimato.

**Un hacker ético siempre debe permanecer anónimo.**

Por qué? Veamos.

## 23. Importación del módulo Nmap

Sanjib Sinha 

(1) Howrah, Bengala Occidental, India

---

Nmap (Network Mapper) es un escáner de seguridad. Fue escrito originalmente por Gordon Lyon (también conocido por su seudónimo Fyodor Vaskovich). Esta herramienta se utiliza especialmente para descubrir hosts y servicios en una red informática. Al encontrar los hosts y servicios, crea un "mapa" de la red. Por esta razón ha sido ampliamente llamado 'Nmap' o también se le puede llamar 'Network Mapper'. Es considerado como una herramienta esencial en su búsqueda para ser un buen y competente hacker ético.

Para obtener los mejores resultados, Nmap suele enviar paquetes especialmente diseñados al host de destino y, a continuación, analiza las respuestas y encuentra los puertos abiertos. También evalúa la vulnerabilidad de una red informática.

Este software ampliamente utilizado por los hackers tiene varias características. En realidad sondea las redes informáticas, descubriendo hosts y servicios. También detecta el sistema operativo y decide la vulnerabilidad de los sistemas encontrando los puertos abiertos.

Python realmente amplía estas funciones para que pueda realizar fácilmente una detección de servicios más avanzada, detección de vulnerabilidades y otras tareas.

Primero comprobemos si el módulo 'Nmap' de python ya ha sido instalado en nuestro sistema o no, emitiendo un simple comando en el terminal.

```
mapa
```

Nos da una larga lista que es muy importante. Muchas cosas que puedes aprender de este listado, ya que dice acerca de la versión, los usos y al final también dice dónde puedes conseguir el manual para más lectura.

Nmap 6.40 ( http://nmap.org )  
Uso: nmap [Tipo(s) de escaneo] [Opciones]  
{especificación del objetivo}

**ESPECIFICACIÓN DEL OBJETIVO:**

Puede pasar nombres de host, direcciones IP, redes, etc. Ej: scanme.nmap.org, microsoft.com/24, 192.168.0.1; 10.0.0-255.1-254

-iL <nombredearchivo de entrada>: Entrada de la lista de hosts/redes

-iR <num hosts>: Elegir objetivos aleatorios

--excluye <host[,host2][,host3],...>: Excluir hosts/redes

--excludefile <exclude\_file>: Excluir la lista de archivar

**DESCUBRIMIENTO DE HOST:**

-SL: Escaneo de listas - simplemente lista los objetivos a escanear

-sn: Escaneo de Ping - deshabilitar el escaneo de puertos

-Pn: Tratar todos los hosts como en linea: omitir la detección de hosts

-PS/PA/PY[lista de puertos]: Detección de TCP SYN/ACK, UDP o SCTP en puertos determinados

-PE/PP/PM: Sondas de detección de solicitudes de eco ICMP, marca de tiempo y máscara de red

-PO[lista de protocolos]: Protocolo IP Ping

-n/-R: Never do DNS resolution/Always resolve[por defecto: a veces]

--servidores-dns <serv1[,serv2],...>: Especificar servidores DNS personalizados

---.sistema-dns: Usar el resovedor de DNS del sistema operativo

--traceroute: Rastree la ruta de salto a cada una de las TÉCNICAS DE ESCANEADO del host:

-sS/sT/sA/sW/sM: TCP SYN/Connect() /ACK/Window/Maimon scans

---.sU: Escaneo UDP

-sN/sF/sX: Escaneos TCP Null, FIN y Xmas

--Escanear banderas <banderas>: Personalice los indicadores de escaneo TCP

-sI <zombie host[:probeport]>: Escaneo en reposo  
-sY/sZ: escaneos SCTP INIT/COOKIE-ECHO  
-sO: Escaneo de protocolo IP

-b <FTP relay host>: FTP bounce scan  
ESPECIFICACIÓN DE PUERTO Y PEDIDO DE  
ESCANEO:

-p <rangos de puertos>: Sólo escanear  
puertos especificados Ex: -p22; -p1-  
65535; -p U:53,111,137,T:21-  
25,80,139,8080,S:9

-F: Modo rápido - Analizar menos puertos que los  
predeterminados

escu  
driñ -r: Escanear puertos consecutivamente - no  
ar aleatorizar  
--puertos superiores <número>: Escanear <número>  
más ~~en~~

compuertas  
--port-ratio <ratio>: Los puertos de escaneo son  
más comunes que  
<ratio> <ratio>

DETECCIÓN DE SERVICIO/VERSIÓN:

-sv: Puertos abiertos de la sonda para  
determinar la información de servicio/versión  
--intensidad de la versión <nivel>: Ajuste de 0  
(luz) a 9 (pruebe todas las sondas)  
--...versión-luz: Límite de las sondas más  
probables (intensidad 2)  
--Version-all: Pruebe cada una de las sondas  
(intensidad

9)  
--Rastreo de versión: Mostrar el escaneado  
detallado de la versión  
(para depurar) SCRIPT

SCAN:

-sc: equivalente a --script=default  
--script=<Lua scripts>: <Lua scripts> es una  
lista separada por comas de  
directorios, archivos de scripts o  
scripts.

tipos

--script-args=<n1=v1,[n2=v2,...]>:  
proporciona argumentos a los scripts  
--script-args-file=nombredarchivo: proporciona  
args de script NSE en un archivo  
--Rastreo de guión: Mostrar todos los datos

enviados y recibidos

--....actualizado por scriptb: Actualizar la base  
de datos del script.

--script-help=<Lua scripts>: Mostrar ayuda sobre  
scripts.

<Lua scripts> es una lista de archivos  
de scripts separados por comas o

categorías de guiones.

#### DETECCIÓN DEL SO:

-O: Habilitar la detección de SO

--Límite: Límite la detección del sistema operativo a objetivos prometedores.

--Adivina: Adivina el SO de forma más agresiva: TIEMPO Y RENDIMIENTO:

Las opciones que toman <tiempo> son en segundos, o añaden 'ms' (milisegundos),

's' (segundos), 'm' (minutos), o 'h' (horas) al valor (por ejemplo, 30m).

-T<0-5>: Establecer la plantilla de tiempo (cuanto más alto, más rápido)

--mini-grupo anfitrión/grupo anfitrión máximo  
≤tamaño>: Tamaños de grupo de escaneo de host paralelo  
--miniparalelismo/paralelismo máximo <numprobes>:  
Paralelización de la sonda

--time-out-min-art-timeout/max-artt-timeout/initial-artt-tiempout <time>: Especifica tiempo de viaje de ida y vuelta de la sonda.

--max-retries <tries>: Capsula el número de retransmisiones de la sonda de exploración de puertos.

--Tiempo de espera del host <tiempo>: Ríndete en el blanco después de tanto tiempo.

--Escanear-retardo/--max-escanear-retardo  
<tempo>: Ajustar el retardo entre las sondas  
----a un precio mínimo de <número>: Enviar paquetes no más despacio que  
<Número> por segundo

--tasa máxima <número>: Enviar paquetes no más rápido que

<Número> por segundo

#### EVASIÓN DE FIREWALL/IDS Y SPOOFING:

-f; --mtu <val>: paquetes de fragmentos (opcionalmente con MTU)

-D <decoy1,decoy2[,ME],...>: Ocultar un escaneo con señuelos

-S <Dirección\_IP>: Dirección de origen falsa

-e <iface>: Usar la interfaz especificada

-g/--fuente-puerto <portnum>: Utilizar un número de puerto determinado

--longitud de los datos <num>: Añadir datos aleatorios a los paquetes enviados

--opciones-ip <opciones>: Enviar paquetes con opciones ip especificadas  
--ttl <val>: Configurar el campo de tiempo de vida de la IP  
--spoof-mac <dirección/prefijo/nombre del proveedor>: Falsear su dirección MAC  
--Badsum: Envía paquetes con una suma de comprobación TCP/UDP/SCTP falsa.

SALIDA:

-oN/-oX/-oS/-oG <archivo>: Escaneo de salida en normal, XML, s|<rIpt kIddi3, y Grepable, respectivamente, al nombre de archivo dado.

-oA <nombre de base>: Salida en los tres formatos principales a la vez

-v: Aumentar el nivel de verbosidad (usar -vv o más para un mayor efecto)

-d: Aumentar el nivel de depuración (use -dd o más para un mayor efecto)

--...razón: Mostrar la razón por la que un puerto se encuentra en un estado determinado

--abierto: Mostrar sólo puertos abiertos (o posiblemente abiertos)

--...rastreo de paquetes: Mostrar todos los paquetes enviados y recibidos

--iflist: Imprimir interfaces de host y rutas (para depuración)

--errores en el registro: Registrar errores/avisos en el archivo de salida de formato normal

--...salida de propulsión: Añada a los archivos de salida especificados por clobber en lugar de hacerlo.

--Reanudar <nombredearchivo>: Reanudar un escaneo abortado

--hoja de estilo <ruta/URL>: Hoja de estilo XSL para transformar la salida XML a HTML

--...webxml: Hoja de estilo de referencia de Nmap.Org para un XML más portátil

--Sin sábana de estilo: Evite la asociación de hojas de estilo XSL con salida XML

MISC:

-6: Habilitar el análisis de IPv6  
-A: Habilitar detección de SO, detección de versión,

escaneo de scripts y traceroute  
--datadir <dirname>: Especificar la ubicación del archivo de datos de Nmap personalizado  
--send-eth/--send-ip: Envío mediante tramas ethernet en bruto o paquetes IP  
--Privilegiado: Suponga que el usuario es totalmente privilegiado  
--...sin privilegios: Suponga que el usuario carece de privilegios de socket en bruto  
-V: Imprimir número de versión  
-h: Imprimir esta página de resumen de la ayuda.

EJEMPLOS:

```
nmap -v -A scanme.nmap.org
nmap -v -sn 192.168.0.0/16 10.0.0.0/8
nmap -v -iR 10000 -Pn -p 80
```

VER LA PÁGINA DE MANIPULACIÓN (<http://nmap.org/book/man.html>) PARA MÁS OPCIONES Y EJEMPLOS

**Puede obtener más información sobre Network Mapper en Internet.  
Por favor, siga estos enlaces.**

<http://nmap.org/> DIFUNDE LA PALABRA-  
<https://nmap.org/book/inst-other-platforms.html> <https://nmap.org/book/inst-windows.html> <https://nmap.org/book/vscan.html>

Si en tu versión 'Linux' del sistema operativo por defecto no obtienes este listado, puedes instalar 'Nmap' emitiendo un simple comando.

```
sudo apt-get install nmap
```

En su máquina virtual, si ejecuta kali Linux, encontrará que 'Nmap' ya ha sido instalado.

Ahora, una vez finalizada esta parte de la instalación, podemos tener muy rápidamente un pequeño script python para ver cómo funciona nuestro módulo 'Nmap'.

Ya has aprendido a usar el editor de texto 'nano' en tu terminal. Así que ábrelo con este comando:

```
sudo nano test.py
```

Primero le pedirá su contraseña de root y luego abrirá el editor de texto nano en su terminal. Escribe un guión corto como este:

```
#!/usr/bin/python
import nmap
nm = nmap.PortScannerAsync()
def callback_result(host,
    scan_result): print '(' - )
    print (host, scan_result)
nm.scan('127.0.0.1', arguments="-O -v",
callback=callback_result
) mientras
nm.still_scanning():
    print("Waiting >>>")
    nm.wait(2)
nm1 =
nmap.PortScanner() a =
nm1.nmap_version()
imprimir (a)
```

Si ejecuta su script 'test.py', obtendrá esta salida:

```
Esperando >>>
-----
('127.0.0.1.1',
Ninguno)
(6, 40)
```

Es tu dirección local. Pero nos interesa el objetivo remoto. Ejecute el Linux kali en su Caja Virtual y abra el navegador 'Tor'.

Buscar 'cuál es mi dirección IP'. Le dará una dirección IP anónima todo el tiempo. Cada vez que usted busca esa dirección IP cambia.

En su caso, puede salir como:

```
x.x.x.xx.xxx
ISP: Algunos Internet LTD
```

Por lo general, está demasiado lejos de su ubicación original! De todos modos, puedes probar la IP y ver el resultado. Pero es una buena práctica probar la IP de <http://nmap.org>

## 24. Construyendo un Escáner de Red Nmap

Sanjib Sinha 

(1) Howrah, Bengala Occidental, India

---

Ahora estamos listos para hacer más pruebas de red usando scripts python. Y esta vez intentaremos construir un escáner más robusto y también intentaremos detectar los puertos abiertos y ver si hay alguna vulnerabilidad.

Escribamos primero el guión en pitón. Y después de eso veremos la salida. Cambiemos el script 'test.py' a esto:

```
#!/usr/bin/python
import nmap
nm = nmap.PortScanner()
imprimir
(nm.nmap_version())
nm.scan('x.x.x.xx.xxx', '1-1024', '-v')
print(nm.scaninfo())
print(nm.csv())
```

Aquí'-v' significa versión y'1-1024' significa el rango de los números de puerto.

Es un guión muy pequeño, pero mira su potencia en la salida.

```
hagudu@hagudu-H81M-S1:~ $ ./test.py
(6, 40)
{'tcp': {'servicios': `1-1024`, `método`:
`conectar`} }".
host;hostname;hostname_type;protocol;port;name;state;
product;extrainfo;reason;version;conf;cpe
x.x.x.xx.xxx;host3.x0x;PTR;tcp;22;ssh;open;syn-
```

```

ack;;3;
  x.x.x.xx.xxx;host3.x0x;PTR;tcp;25;smtp;open;;syn-
ack;;3;
  x.x.x.xx.xxx;host3.x0x;PTR;tcp;53;dominio;abierto;;syn
- ack;;3;
  x.x.x.xx.xxx;host3.x0x;PTR;tcp;80;http;open;;syn-
ack;;3;
  x.x.x.xx.xxx;host3.x0x;PTR;tcp;137;netbios
- ns;filtrado;;sin respuesta;;3;
  x.x.x.xx.xxx;host3.x0x;PTR;tcp;138;netbios
- dgm;filtrado;;sin respuesta;;3;
  x.x.x.xx.xxx;host3.x0x;PTR;tcp;139;netbios
- ssn;filtrado;;sin respuesta;;3;
  x.x.x.xx.xxx;host3.x0x;PTR;tcp;445;microsoft
- ds;filtrado;;sin respuesta;;3;

```

Muestra que todos juntos cuatro puertos están abiertos. Lo son: 22, 25, 53 y 80.

Y los otros son filtrados.

Antes de ir a probar otro puerto y esta vez podemos mostrar la IP tal y como es de <http://nmap.org>, vamos a tener unos datos muy breves sobre la terminología del puerto. También puede encontrar la parte legal del escaneado aquí: <https://nmap.org/book/legal-issues.html>.

El puerto es una ubicación de red direccionable. Está idealmente implementado dentro del sistema operativo y este sistema operativo nos ayuda a discriminar el tráfico web. Este tráfico está destinado a diferentes aplicaciones o servicios, como algunos para `mail', otros para `HTTP' y así sucesivamente.

A continuación nos interesa el análisis de puertos. En una palabra, es un tipo de proceso y este proceso usualmente trata de conectarse a un número de puertos secuenciales, como acaba de ver en la salida anterior. Queremos saber qué puertos están abiertos y qué servicios y sistemas operativos están detrás de ellos.

Escaneemos otra dirección IP (<http://nmap.org>) y al hacerlo hemos cambiado un poco el script de python.

```

#!/usr/bin/python
import nmap
nm = nmap.PortScanner()
imprimir
(nm.nmap_version())
nm.scan('192.168.146.1', '1-1024', '-v')

```

```
print(nm.scaninfo())
```

```
print(nm.csv())
```

La salida es así:

```
(6, 40)
{'tcp': {'servicios': `1-1024', `método':
`conectar'}}}.
host;hostname;hostname_type;protocol;port;name;state;product;extrainfo;reason;version;conf;cpe
192.168.146.1;;;tcp;25;smtp;open;;syn-ack;;3;
192.168.146.1;;;tcp;53;dominio;abierto;;syn-ack;;3;
192.168.146.1;;;tcp;80;http;open;;syn-ack;;3;
```

Los puertos abiertos son 25, 53 y 80. No se muestran puertos filtrados en esta máquina.

Consigamos todos los hosts de esa IP con un pequeño cambio en nuestro script anterior.

Esta vez reducimos el rango para que nuestro programa no se ejecute por mucho tiempo.

```
#!/usr/bin/python
import nmap
nm = nmap.PortScanner()
imprimir
(nm.nmap_version())
nm.scan('192.168.146.1', '22-455', '-v --version-all')
print(nm.all_hosts())
```

Hemos cambiado el número de puertos de la línea número cinco.

También hemos eliminado las dos últimas líneas y queremos ver si podemos obtener más datos de esa máquina.

La salida muestra que sólo hay un host.

```
(6, 40)
{'tcp': {'servicios': `22-455', `método':
`conectar'}}}.
['192.168.146.1']
```

Cambiemos y volvamos al IP anterior y veamos el resultado.

```
#!/usr/bin/python
import nmap
```

```
nm = nmap.PortScanner()
imprimir
(nm.nmap_version())
nm.scan('x.x.x.xx.xxx', '22-455', '-v --version-
all') print(nm.all_hosts())
```

Nada cambia. La salida nos dice sobre el único host. Hay más por venir.

Como queremos más información, lo ideal sería cambiar nuestro código `test.py'.

```
#!/usr/bin/python
import nmap
nm = nmap.PortScanner()
imprimir
(nm.nmap_version())
nm.scan('192.168.146.1', '22-1024', '-v --version-
todos')
imprimir (nm.scanstats())
print (nm['192.168.146.1'].state())
print (nm['192.168.146.1'].all_protocols())
print (nm['192.168.146.1']['tcp'].keys())
```

Esta vez la salida es más verbosa.

```
(6, 40)
{'uphosts': '1', 'timestr': 'Mon 3 09:53:35
2016', 'downhosts': '0', 'totalhosts': '1',
'elapsed':
'5.73'}
up
['tcp']
[80, 25, 53]
```

Verás que un huésped está arriba.

No hay downhosts y el número total de anfitriones es de 1 como se esperaba. También vemos la hora exacta en la que se está realizando el escaneo y el tiempo transcurrido.

Excavemos un poco más.

Hemos utilizado el rango de puertos '1-1024'. Normalmente, los puertos por debajo de 1024 están asociados con servicios similares a Linux y Unix. Estos sistemas operativos se consideran vitales para las funciones esenciales de la red.

Por esa razón debe tener privilegios de root para asignar servicios a este tipo de sistema operativo.

Si desea ir más allá de 1024, hay ya sea `registrados' o `privados'.

puertos. Los puertos entre 49152 y 65535 se supone que son para uso privado.

Consideremos el primer resultado y tratemos de entender qué puerto se utiliza para qué fines.

```
x.x.x.xx.xxx;host3.x0x;PTR;tcp;22;ssh;ssh;open;;syn-
ack;;3;
x.x.x.xx.xxx;host3.x0x;PTR;tcp;25;smtp;open;;syn-
ack;;3;
x.x.x.xx.xxx;host3.x0x;PTR;tcp;53;dominio;abierto;;syn-
ack;;3;
x.x.x.xx.xxx;host3.x0x;PTR;tcp;80;http;open;;syn-
ack;;3;
x.x.x.xx.xxx;host3.x0x;PTR;tcp;137;netbios
- ns;filtrado;;sin respuesta;;3;
x.x.x.xx.xxx;host3.x0x;PTR;tcp;138;netbios
- dgm;filtrado;;sin respuesta;;3;
x.x.x.xx.xxx;host3.x0x;PTR;tcp;139;netbios
- ssn;filtrado;;sin respuesta;;3;
x.x.x.xx.xxx;host3.x0x;PTR;tcp;445;microsoft
- ds;filtrado;;sin respuesta;;3;
```

El puerto 22 se utiliza para 'SSH'. Significa 'Secure Socket Shell'. Es un protocolo de red con el que los administradores acceden a un ordenador remoto de forma segura.

El puerto 25 es para SMTP o correo. El puerto 53 es sinónimo de servicios DNS. El puerto 80 es para el tráfico web.

Los puertos 137, 138 y 139 son capturados por Microsoft para transportar su protocolo NetBIOS a través de redes LAN y WAN basadas en IP.

Por último, el puerto 445 se utiliza para Microsoft Directory Services. Para más información sobre este puerto, puede que le interese el siguiente enlace: [https://www.grc.com/port\\_445.htm](https://www.grc.com/port_445.htm).

# Parte III

## 25. Proteger el anonimato en Internet

Sanjib Sinha 

(1) Howrah, Bengala Occidental, India

---

Esto es muy importante para los hackers éticos. Usted necesita permanecer anónimo y ocultar su dirección IP mientras se encuentra en el mundo del hacking ético. Hay varias maneras de hacerlo. En este capítulo discutiremos cómo podemos hacerlo.

Hay apoderados. Significa que está enrutiando a través de diferentes enruteadores, pero puede ser muy lento y no está a su alcance. Otra desventaja del uso de proxies es que no sabes nada sobre el otro lado. Usted está en la oscuridad sobre los servidores a través de los cuales sus paquetes se están moviendo. Eso podría ser muy arriesgado. Usted se preguntará por qué es peligroso. Haría algún tipo de trabajo de "mapeo de la red". Es inofensivo. Tal vez sea así. Pero no sólo se limita a esa parte. Usando el proxy, es posible que desee iniciar sesión en algún servidor. Una vez que haya escrito su contraseña, puede ser secuestrada.

¿Cómo puede resolver este problema?

Hay un término: "VPN". Usted probablemente ha oído hablar de "red privada virtual"

." ¿Qué es eso? Es básicamente un tipo de servicio que está utilizando para encriptar su tráfico. Y es muy rápido. En el futuro, cuando trabaje como profesional, tendrá que contratar un servicio de VPN. No es muy costoso. Por el momento podríamos hacerlo de forma gratuita, sólo con fines educativos.

Pero una vez que encriptas tu tráfico a través de VPN, es reconocible. ¿Qué sucede si una agencia solicita sus datos a los proveedores de servicios? Normalmente, para evitar que tengas que ser exigente. Usted necesita contratar un servicio de una cierta parte del mundo donde la privacidad es estrictamente mantenida.

Pero después de decir esto y esperar lo mejor, definitivamente no te recomendaría que hicieras tu hacking ético usando proxies o VPNs. Básicamente, es posible que desee hacerlo para evitar la configuración del firewall o ese tipo de cosas.

Además, algunos servicios VPN no permiten que las direcciones IP usen sus servicios.

más allá de su alcance. Suponga que su dirección IP está fuera de ese rango. Pero la gente a menudo utiliza proxys o VPNs, no siempre para hacer algo malicioso como derribar un servidor o robar datos. La gente podría querer esconder su ubicación justo cuando está viajando, o ese tipo de cosas. Aparentemente este tipo de actividad se mantiene dentro de la ley.

Hay otro problema que puede surgir cuando se accede a un determinado tipo de servidor que normalmente permite direcciones IP de una determinada región. En ese caso, si utiliza una dirección IP de China o Rusia, el administrador de la red seguramente irá tras de usted. Así que es un problema consistente que sigue viniendo y atormentándolos de vez en cuando y en los próximos capítulos nos gustaría tratar esos problemas.

## 26. Dark Web y Tor

Sanjib Sinha 

(1) Howrah, Bengala Occidental, India

---

Mientras tanto, echaremos un vistazo rápido a la web oscura o a la web oculta. No sé si has oído hablar de ello antes.

El rumor es que la "red oscura" o "red profunda" consiste en una gran parte de Internet. Es algo así como la "materia oscura" que consiste en el 97 o 98 por ciento de la masa del universo. Todavía se desconoce lo que es en realidad, excepto por unas pocas cosas.

La gente dice que la web oscura está llena de información que normalmente no obtenemos. Y no puedes acceder a la web oscura a través de tu navegador normal. Necesitas un tipo especial de navegador para entrar en ese laberinto de misterios.

Debo advertirte antes de que intentes Tor y entres en la red oscura. Hay muchas actividades ilegales, generalmente fuera de nuestra percepción normal. Podría ser como el tráfico de personas. Podría ser como el tráfico ilegal de armas. Podría ser como contratar asesinos y todo eso. Pero en este libro no nos interesan. Nuestra principal preocupación es el conocimiento. Llegamos allí para poder tener una idea de lo que está sucediendo en la red oscura.

Como un hacker ético, usted necesita saber todo por un solo y único propósito. Necesitas luchar contra un ataque malicioso. Estás aprendiendo a defenderte, no a atacar a alguien. Pero para defenderte, necesitas saber todas las tácticas que tu enemigo usa a menudo. Tal vez la policía busque su ayuda para localizar a un abusador de niños. Sin conocer el carácter apropiado de la telaraña oscura, no se puede hacer eso. Si no sabe cómo ocultar su dirección IP, ¿cómo puede localizar a un delincuente que oculta su ubicación real?

Además, necesitas saber otra cosa importante. La red oscura no siempre es mala en ese sentido. Usted puede encontrar muchos hackers reputados de sombrero blanco o sombrero gris en ciertos foros que se mantienen completamente ocultos de los vigilantes.

ojos de las agencias gubernamentales. Usted puede encontrar gente muy útil allí que le puede ayudar a resolver su problema instantáneamente. Al igual que Wikipedia, hay wiki ocultos que pronto vamos a ver, donde puedes encontrar un montón de cosas interesantes para aprender.

---

## Wikipedia oculta

Para leer el wiki oculto necesitamos instalar el navegador Tor. Kali Linux no viene con él por defecto, por lo que es necesario instalarlo.

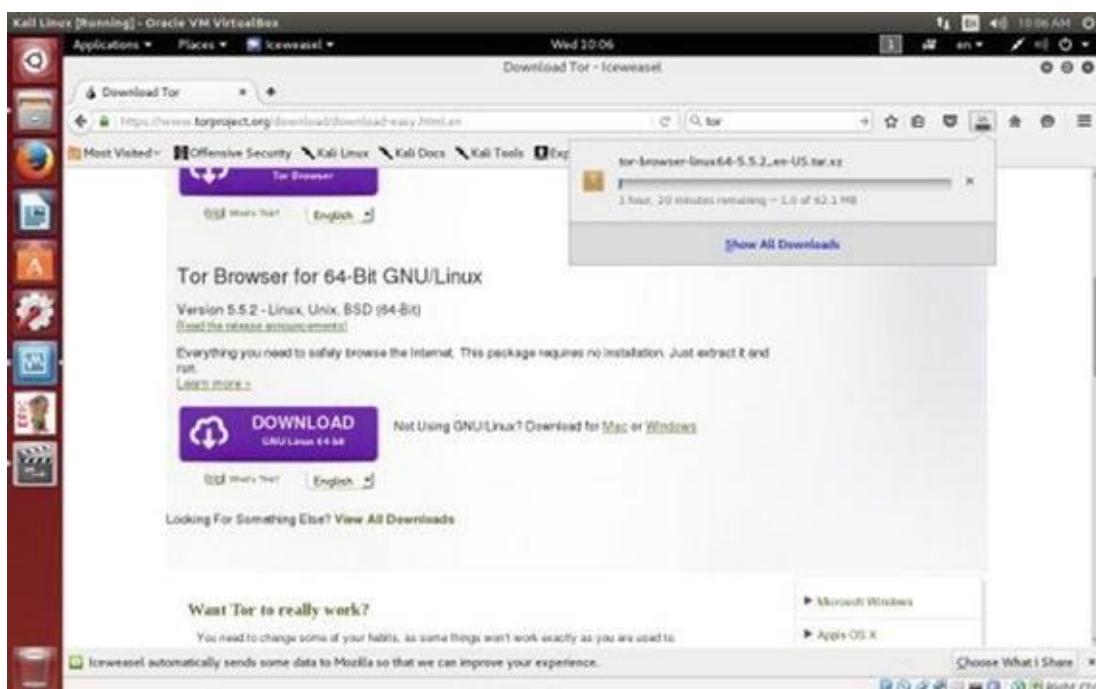


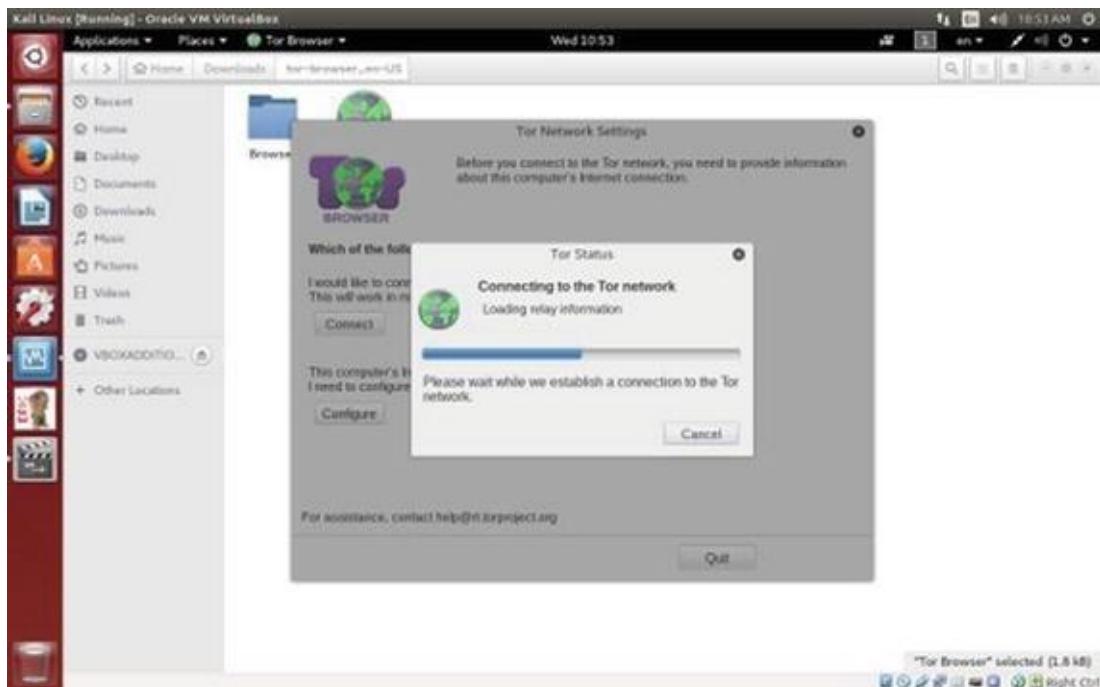
Figura 26-1. Sección de descargas del sitio web torproject.org

Para ello, primero debe iniciar sesión como el nuevo usuario: xman. Luego abra el navegador normal y busque el navegador Tor. Sólo tienes que ir al sitio oficial y descargar la última versión para Kali Linux. Tenga cuidado al comprobar que se trata de <https://torproject.org>, no de cualquier otra cosa. Puede venir con "http://" sin la "s". Simplemente evita eso.

Hay dos versiones: una de 32 bits y la otra de 64 bits. De acuerdo con la arquitectura de su sistema, usted necesita descargar la versión exacta. Antes de descargar, es una buena práctica aprender acerca de Tor a partir de su documentación.

Hay términos y condiciones que usted debe cumplir. Y el término principal es que usted debe permanecer dentro de la ley. No puedes usar Tor para ningún proceso ilegal. Tor también oculta tu dirección IP. Pero esa es una cuestión diferente.

Una vez finalizada la descarga, podrá acceder al archivo necesario en su carpeta "Download". Sólo háganlo.



**Figura 26-2.** El navegador Tor se está conectando .

Una vez conectado, se abrirá su primera página por defecto, la cual encontrará muy diferente a la del navegador normal. En primer lugar, puedes escribir "what is my IP" y comprobar lo que muestra.

Definitivamente será algo más que la región en la que te encuentras. Pero necesitamos páginas web wiki ocultas originales que nos lleven a la web oscura.

Recuerde, hay varios sitios web que afirman ser wiki ocultos originales. Así que tienes que ser sensato a la hora de elegir. Normalmente vienen con dominios ".cebolla" y la URL cambia continuamente. Así que puedes escribir algo como "hidden wiki url" y ver lo que obtienes.

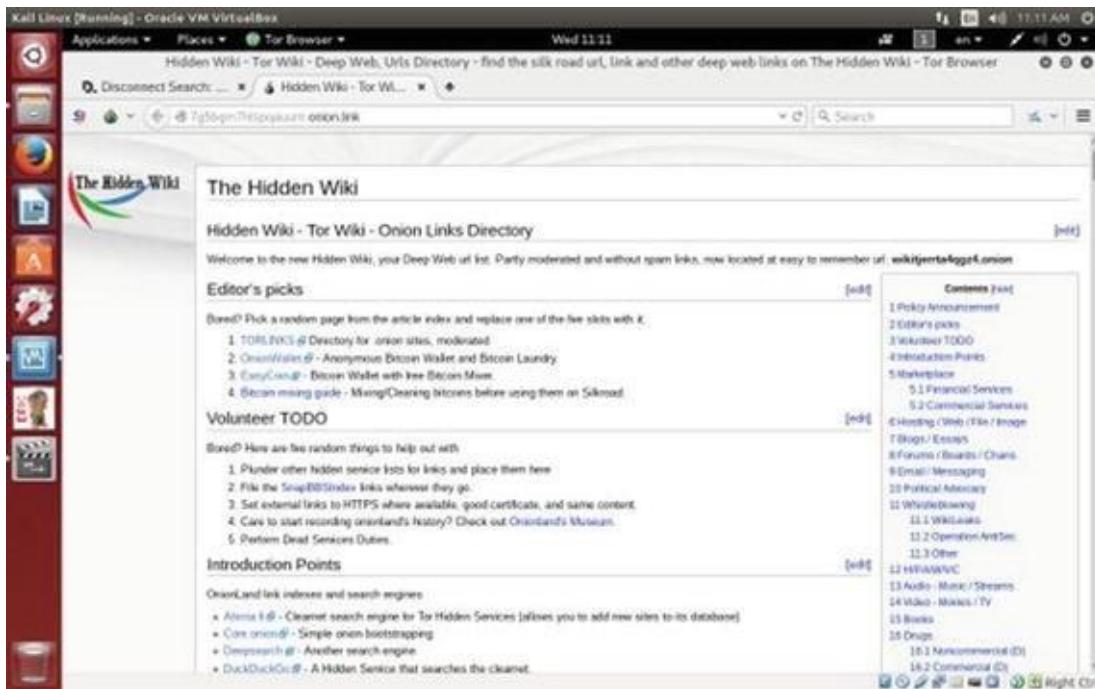


Figura 26-3. La página Wiki original oculta

El problema principal es que no puedes diferenciar el wiki original oculto de las otras versiones falsas. La imagen de arriba muestra cómo puede verse. La extensión es siempre ".cebolla".

El wiki oculto consiste principalmente en una gran cantidad de varios enlaces. Muchos de ellos son simplemente ilegales y baratos. Parece un gran mercado en el que se venden muchas mercancías de contrabando. Nunca intentes comprar nada de aquí. Aunque es tentador comprar algo muy costoso a un tercio de su precio original, no es seguro que le llegue. Además, existe la posibilidad de que su número de débito o crédito esté agrietado.

Pero en este llamado mercado interesante, hay muchas cosas realmente útiles que pueden venir en su ayuda. Uno de ellos es el foro o la sección de chat donde los hackers de renombre a menudo discuten muchas cosas interesantes que no se ven por lo general en cualquier foro abierto.

Al mismo tiempo, debe tener cuidado al usar cualquier código que provenga de estos foros o chats sólo por el anonimato. No es aconsejable usar ese código en su máquina original.

¡Eso podría ser peligroso!

Abramos un sitio en el foro y veamos cómo se ve. Suelen tener un fondo negro, como si representaran bien la telaraña oscura.

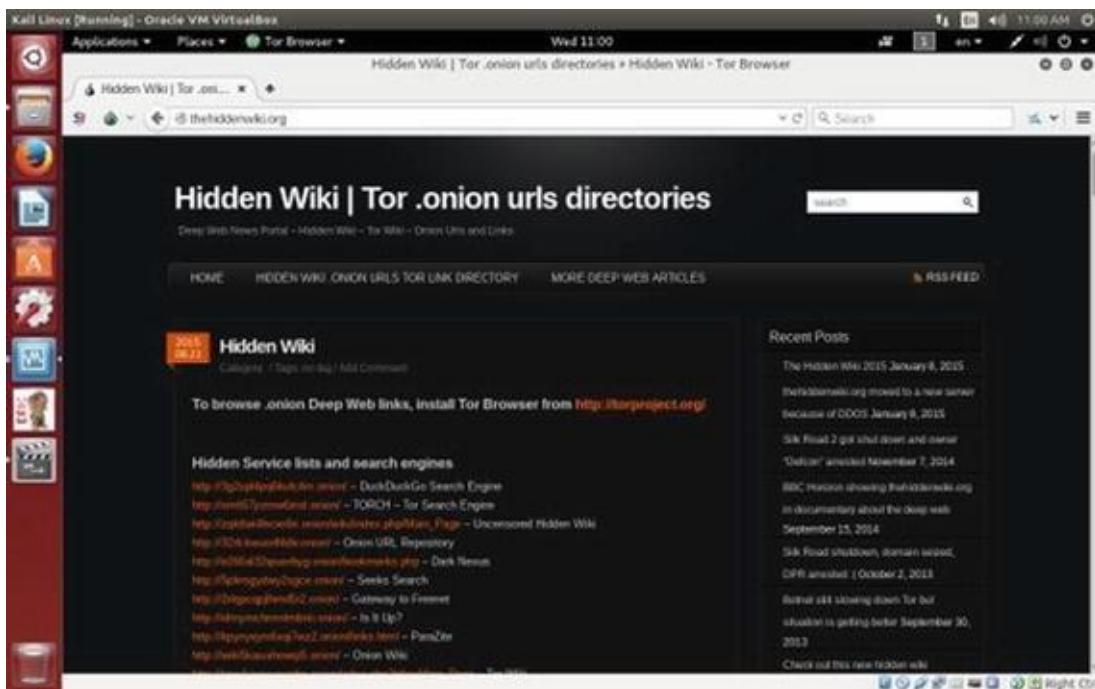


Figura 26-4. Una de las páginas ocultas de la Wiki - que podría ser vaga

Una cosa que es mejor que tengas en cuenta: El navegador Tor está bien mientras quieras aprender algo nuevo. No es para hacer cosas dudosas. Hay un montón de atracciones baratas que definitivamente tratarían de atraer su atención o incluso forzarlo a ir hacia ellos. Tenga cuidado al elegir los sitios que visita. Mientras sea un foro de hackers, está perfectamente bien. Pero una vez que se sobrepasa el límite sin cumplir con la ley, puede ser peligroso.

Ahora pasaremos a las cosas que están más directamente relacionadas con el pirateo ético en el mundo real. Pero antes de eso, necesitamos ver cómo funcionan las cadenas de proxy y las VPNs.

Su poco conocimiento de los comandos de Linux le será útil. A partir de ahora, todo lo que hagamos estará en el terminal de Kali Linux... Así que encienda su máquina virtual Kali y abra su terminal. Primero aprenderemos sobre las cadenas de proxy y, con la ayuda de esta herramienta, cómo podemos ocultar nuestra dirección IP y obtener acceso a un servidor remoto.

## 27. Cadenas de proxy

Sanjib Sinha 

(1) Howrah, Bengala Occidental, India

---

El nombre sugiere su verdadero significado. Para mantener el anonimato necesitamos varios poderes. Detrás de estos poderes podemos ocultar nuestra verdadera identidad. No siempre tiene éxito. Pero Kali Linux te da una oportunidad especial para cambiar la configuración en la raíz para que puedas ocultar tu verdadera identidad mientras navegas por la web usando Tor . En realidad, en este caso necesita configurar su archivo "proxychain.conf". Ya has instalado Tor.

Necesitamos abrir el archivo de configuración usando el editor de texto "nano".

Abra su terminal Kali Linux como usuario root y escriba este comando.

```
root@kali:~# nano /etc/proxchains.conf
```

Se abrirá el archivo "proxchains.conf". Hay tres tipos de proxies que puede utilizar. Pero no puedes usar todos los proxies al mismo tiempo. Veamos primero cómo se ve este archivo. Tiene 68 líneas de largo. Pero no es muy complicado si usted entiende las líneas. Las documentaciones son claras y directas. Aquí están las primeras líneas importantes.

```
# La siguiente opción identifica cómo se trata la
lista proxy.
# sólo una opción debe ser descomentada a la
vez, # de lo contrario, la última opción que
aparecerá será
aceptado
#
cadena_dinámica
```

```

#
# Dinámico - Cada conexión se hará a través de
proxies encadenados
# todos los proxies encadenados en el orden en
que aparecen en la lista
# al menos un proxy debe estar en línea para
jugar en cadena
# (los apoderados muertos se saltan)
# de lo contrario EINTR es devuelto a la aplicación

```

¿Qué significa esto? Dice que la lista proxy tiene varias opciones. Usted debe saber cómo debe tratar estas opciones. Si lees cada línea, te harás una idea de cómo funciona. Hay tres tipos de apoderados. Tienes que descomentar a cualquiera de ellos.

El primer proxy es "dynamic-chain". Usted ve la línea de arriba y el color rojo muestra que la he descomentado. Hay dos proxies más: "strict\_chain" y "random\_chain". Se comentan a cabo. Tienen sus propias descripciones. Leamos los dos.

```

#
Cadena_estri
ta #
# Estricto - Cada conexión se hará a través de
proxies encadenados
# todos los proxies encadenados en el orden en
que aparecen en la lista
# todos los proxies deben estar en línea para
jugar en cadena # de lo contrario EINTR es
devuelto a la aplicación
#
# cadena_al
azar #
# Aleatorio - Cada conexión se hará a través de un
proxy aleatorio
# (o cadena de proxy, see chain_len) de la
lista. # esta opción es buena para probar su IDS
;
```

Se describe claramente en la documentación que se adjunta. Así que no lo elaboraré de nuevo. La ventaja de elegir "dynamic\_chain" por encima de los demás está claramente indicada. Si su conexión no obtiene un "proxy en funcionamiento", automáticamente salta al otro. Los otros dos no te lo dan.

oportunidad de enrutar su tráfico.

Déjame explicarte más. Suponga que tiene dos apoderados en su lugar: A y B. Lo que sucede en el caso de "strict\_chain" es que cuando navegas por las páginas web, tu conexión se enruta a través de A y B estrictamente. Significa que A y B deben estar en orden y vivir. De lo contrario, la conexión simplemente fallará. En el caso de "dynamic\_chain" esto no sucede. Si A está abajo entonces salta para tomar a B. Funciona de esa manera.

Espero que el primer paso sea claro. Consideremos otros pasos importantes. En el medio tienes una línea como esta:

```
# Peticiones DNS de proxy - no hay fuga de datos
DNS proxy_dns
```

Es una línea muy importante que hay que considerar seriamente. Verá, he descomentado el "proxy\_dns". No puede permitir que se filtren datos DNS. En otras palabras, su dirección IP real no debe ser filtrada por casualidad. Es por eso que he descomentado esta línea, para que sus apoderados estén en el lugar adecuado trabajando sin ningún problema.

Al final de la lista se encuentra esta línea:

```
[Lista de proxy]
# añadir proxy
aquí.... # meanwhile
# default set to "tor"
calcet 127.0.0.1      9050
ines4
calcet 127.0.0.1      9050
ines5

calcet 185.43.7.146    1080
ines5
calcet 75.98.148.183   45021
ines5
```

Por favor, inspeccione las dos últimas líneas en rojo. Los he añadido. Permítanme explicarles por qué las añadí. Pero antes de hacer eso, me gustaría explicar las líneas de ejemplo que acabamos de dar. Leen así:

```
# Formato ProxyList
#           type host puerto[user pass]
#           (valores separados por 'tab' o
'blank') #
#
```

```

#          Ejemplos:
#
#
#                      socks5  192.168.67.78  1080
#                      escaparse
er    secretas
#                      http     192.168.89.3   8080
#                      apenas
tu    escondidos
#                      socks4  192.168.1.49   1080
#                      http     192.168.39.93  8080

```

Indica claramente cómo debe formatearse su lista proxy. Considere la primera línea:

```

#                      socks5  192.168.67.78  1080
#                      escaparse
er    secretas

```

Significa: el primero es el "tipo" del proxy. Debe ser "calcetines5". La segunda lo es: "anfitrión". El tercero es "port" y las dos últimas palabras significan "nombre de usuario" y "contraseña" en caso de que pague por ello. Otra cosa importante es: debe separar las palabras, ya sea usando "tabulador" o presionando "blanco".

Hay varios proxies gratuitos que encontrará, así que no se preocupe por los dos últimos ahora mismo. Ahora podemos volver de nuevo a las últimas líneas que hemos estado debatiendo. En las últimas líneas se ha mencionado que "default set to tor".

Antes de añadir las dos últimas líneas, es necesario añadir esta línea:

```
socks5  127.0.0.1      9050
```

Deberíamos hacerlo porque normalmente su archivo "proxychains.conf" sólo contiene "socks4", por lo que necesita añadir "socks5", que es compatible con la tecnología moderna actual. Ahora puede probar su estado "Tor".

Abra su terminal y escriba: **service tor status**

Fallará si no lo enciendes. So type: **service tor start**

Se iniciará el servicio.

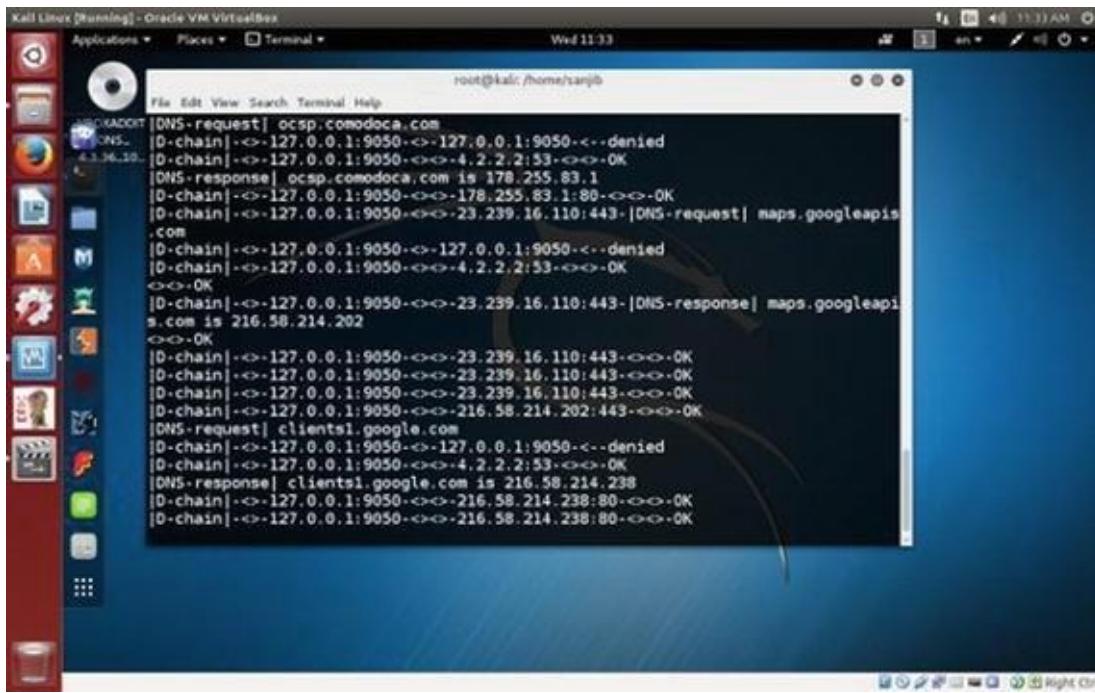
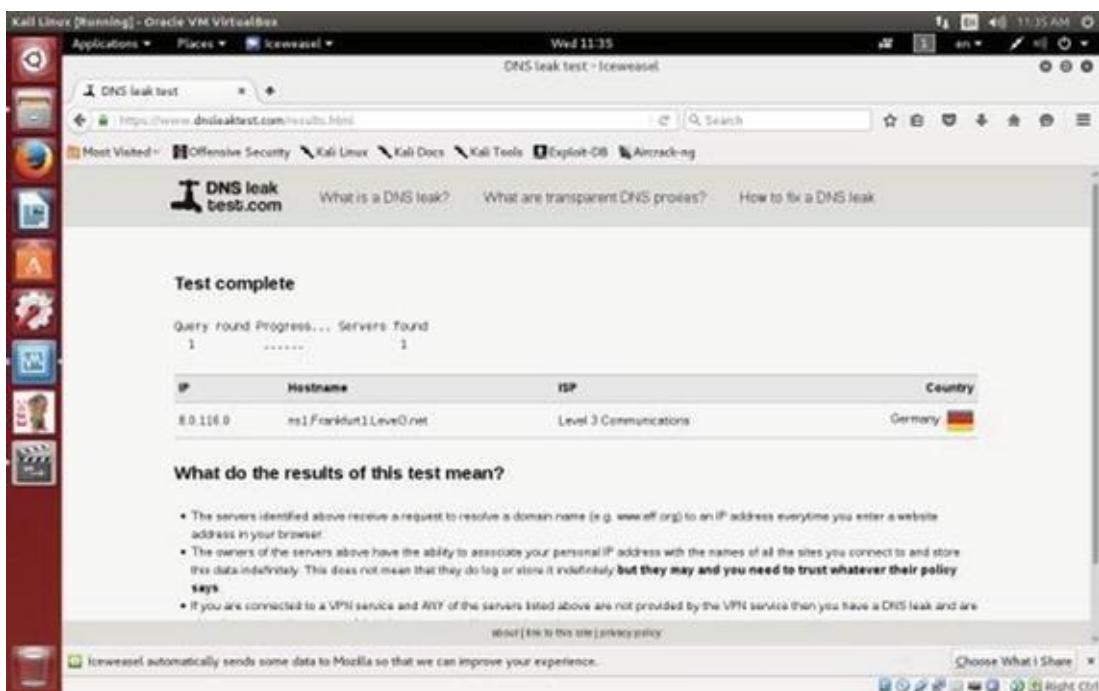


Figura 27-1. Tor está corriendo a través de la terminal

Y puedes abrir tu navegador a través de la terminal. Simplemente escriba:  
**proxychains firefox [www.duckduckgo.com](http://www.duckduckgo.com)**  
<http://www.duckduckgo.com/>

Este motor de búsqueda no suele rastrear las direcciones IP. Su navegador se abrirá y podrá comprobar su dirección IP. También nos gustaría ver el resultado de la prueba de fuga de DNS. Hagámoslo escribiendo "dns leak test" en el motor de búsqueda. Hay varios servicios; puede hacer clic en cualquiera de ellos para ver lo que dice.



**Figura 27-2.** Prueba de fugas DNS

He encontrado que el "www.dnsleaktest.com" está funcionando para encontrar mi dirección IP original y no lo consigue. Muestra una IP como "8.0.116.0" y es de Alemania. Esto está mal, ya que estoy escribiendo esto cerca de Calcuta.

Usted puede probar lo mismo simultáneamente en su navegador normal y encontrará su dirección IP real.

## 28. Red privada virtual o VPN

Sanjib Sinha 

(1) Howrah, Bengala Occidental, India

---

Desde el principio trato de enfatizar una cosa. El hacking ético comienza con un solo concepto: el anonimato.

Primero debes asegurarte de que eres anónimo. No has dejado ningún rastro a tus espaldas. Todo su viaje está oculto y nadie puede rastrear su ruta más tarde.

Hemos discutido el navegador "Tor" y las "cadenas de proxy". Hemos visto cómo podemos utilizarlos. Otro concepto muy importante en este sentido es la red privada virtual o VPN, para abreviar.

Se trata básicamente de la configuración del servidor DNS. Un servidor DNS normalmente comprueba el filtrado de tráfico. Así que si puedes cambiar la configuración de tu servidor DNS en tu raíz, puedes guiar mal esa lectura.

¿Cómo podemos hacer eso?

Abra su terminal Kali Linux y escriba:

```
cat /etc/resolv.conf
```

Mostrará algo como esto:

```
# Generado por el servidor de nombres
NetworkManager 192.168.1.1
```

En su terminal hay muchas posibilidades de que muestre algo más. Esta es la puerta de enlace de tu casa, qué tipo de enrutador estás usando; sólo está mostrando esa información. Básicamente vamos a cambiar esto para que cuando volvamos a probar nuestra dirección IP, el servidor DNS no pueda filtrar el tráfico correctamente.

En mi terminal, cuando escribo el mismo comando, se lee así:

```
servidor de nombre 208.67.222.222
nameserver 208.67.220.220
```

Si adivinaste que yo había cambiado esto, tienes razón. Lo he cambiado. ¿Por qué he cambiado esto? Déjame explicarte.

Necesita entender el concepto de "nameserver" primero... ¿Qué es lo que hace? La dirección IP de la LAN realmente reenvía el tráfico a los servidores DNS, que a su vez resuelven las consultas y envían el tráfico de vuelta en consecuencia.

Al hacer esto también registra la cantidad de tráfico que está teniendo a través de su puerta de enlace. No necesitamos eso. ¿Por qué no necesitamos eso? Necesitamos ser anónimos. Así que esa es la razón principal detrás de cambiar este servidor de nombres.

Podemos hacerlo a través de una red privada virtual o VPN. Volvamos a abrir el terminal y escribamos este comando:

```
nano /etc/dhcp/dhclient.conf
```

Se abrirá el archivo de configuración donde cambiaremos la dirección del servidor de nombres.

Veamos cómo se ve.



The screenshot shows a terminal window titled 'hagudu@hagudu-HELM-51: ~\$'. The window contains the contents of the /etc/dhcp/dhclient.conf file. The file is a configuration script for the dhclient program, which handles dynamic host configuration. It includes comments explaining the purpose of various options like 'option rfc3442-classless-static-routes', 'send host-name', and 'request subnet-mask'. The file also specifies lease times, routers, and domain names. At the bottom of the file, there is a 'script' directive pointing to '/etc/dhcp3/dhclient-script' and a 'reject' command for IP 192.33.137.209. The terminal window has a standard Linux-style interface with icons for file operations and a menu bar at the top.

```
hagudu@hagudu-HELM-51: ~$ nano /etc/dhcp/dhclient.conf
GNU nano 2.2.6
File: dhclient.conf

# Configuration file for /sbin/dhclient, which is included in Debian's
# dhcp3-client package.

# This is a sample configuration file for dhclient. See dhclient.conf's
# man page for more information about the syntax of this file
# and a more comprehensive list of the parameters understood by
# dhclient.

# Normally, if the DHCP server provides reasonable information and does
# not leave anything out (like the domain name, for example), then
# few changes must be made to this file, if any.

option rfc3442-classless-static-routes code 121 = array of unsigned integer 8;

#send host-name "andare.fugue.com";
send host-name = gethostname();
#send dhcp-client-identifier 1:0:a0:24:ab:fb:9c;
#send dhcplease-time 3000;
#supersede domain-name "fugue.com home.vix.com";
prepend domain-name-servers 208.67.222.222 208.67.220.220;
request subnet-mask, broadcast-address, time-offset, routers,
        domain-name, domain-name-servers, domain-search, host-name,
        dhcp6.name-servers, dhcp6.domain-search,
        netbios-name-servers, netbios-scope, interface-mtu,
        rfc3442-classless-static-routes, ntp-servers,
        dhcp6.fqdn, dhcp6.sntp-servers;
#require subnet-mask, domain-name-servers;
#timeout 60;
#retry 60;
#reboot 10;
#select-timeout 5;
#initial-interval 2;
#script "/etc/dhcp3/dhclient-script";
#media "-link1-link2", "-link link1";
#reject 192.33.137.209;

alias {
```

**Figura 28-1.** Archivo dhclient.conf en el editor de texto nano

Lo he abierto en mi terminal de Ubuntu. Pero tienes que cambiarlo en tu

Máquina virtual Kali Linux. Te das cuenta de que hay muchas cosas escritas allí. Pero estamos interesados en esta línea intermedia:

```
prepend dominio-nombre-servidores 127.0.0.0.1;
```

Primero descomentaremos esta línea y luego la cambiaremos. Hay muchas direcciones IP OpenDNS disponibles en la web. Busca con el término "opendns" y se abrirán muchas opciones desde donde podrás copiar las direcciones OpenDNS. Una de ellas es "opendns.com". Copiemos dos direcciones de él y peguémolas en lugar de 127.0.0.0.1 así:

```
prepend dominio-nombre-servidores 208.67.222.222  
208.67.220.220;
```

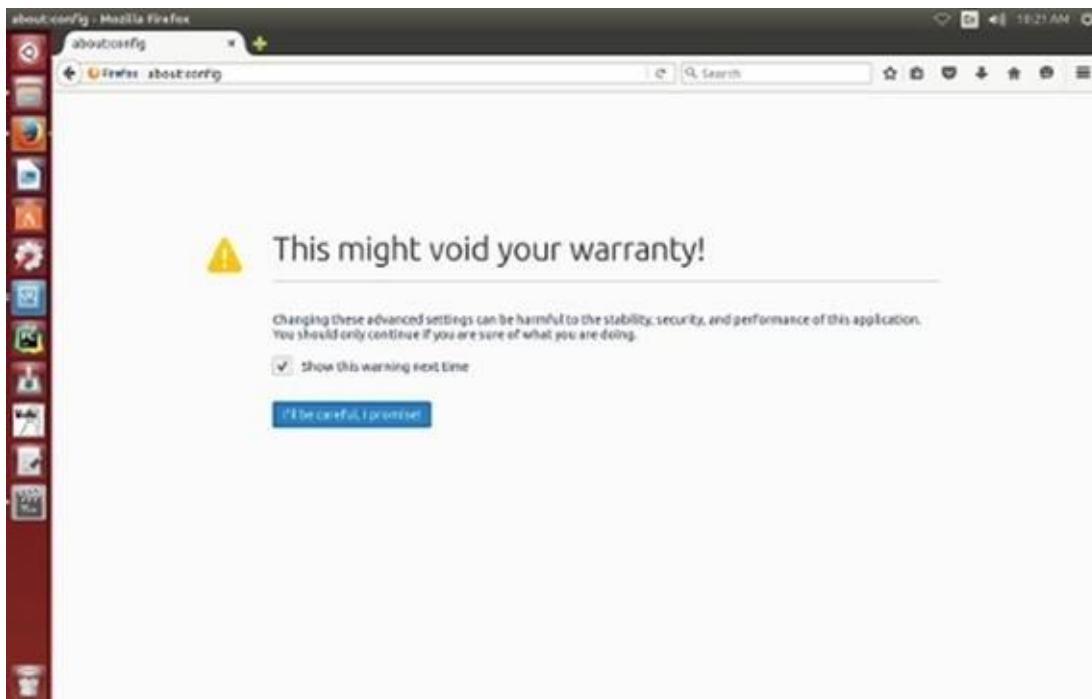
Ahora todo lo que tienes que hacer es una cosa. Debe reiniciar el administrador de red. Escriba este comando en su terminal Kali Linux:

**servicio red-manager reiniciar**

Ahora puede comprobar su servidor de nombres de nuevo. Mostrará dos nuevas direcciones.

Otra cosa es importante aquí. Debe comprobar si la conexión multimedia está activada o no. Abra su navegador Mozilla (en Kali Linux es "Iceweasel"). Lo encuentras en el panel superior izquierdo.

Abra el navegador y escriba "about:config". Se parece a esto:



**Figura 28-2.** about:config image en su navegador Mozilla

Si usas Chrome u Opera, esto mostrará algo más. Necesitas hacer clic y entrar en él. Entrando en él le asegurará un panel de búsqueda en la parte superior donde podrá introducir el término de búsqueda:

"medios.de.comunicación.con.la.posibilidad.de.conexión.entre.pares".

Veamos cómo se ve.

Preference Name	Status	Type	Value
media.peerconnection.capture_delay	default	integer	70
media.peerconnection.default_iceservers	default	string	[]
media.peerconnection.enabled	default	boolean	true
media.peerconnection.ice.default_address_only	default	boolean	false
media.peerconnection.ice.force_interface	default	string	
media.peerconnection.ice.link_local	default	boolean	false
media.peerconnection.ice.loopback	default	boolean	false
media.peerconnection.ice.relay_only	default	boolean	false
media.peerconnection.ice.stun_client_maximum_transmits	default	integer	7
media.peerconnection.ice.tcp	default	boolean	false
media.peerconnection.ice.tcp_so_sock_count	default	integer	0
media.peerconnection.ice.trickle_grace_period	default	integer	5000
media.peerconnection.identity.enabled	default	boolean	true
media.peerconnection.identity.timeout	default	integer	10000
media.peerconnection.turn.disable	default	boolean	false
media.peerconnection.use_document_iceservers	default	boolean	true
media.peerconnection.video.enabled	default	boolean	true
media.peerconnection.video.h264_enabled	default	boolean	false
media.peerconnection.video.max_bitrate	default	integer	2000
media.peerconnection.video.min_bitrate	default	integer	200
media.peerconnection.video.start_bitrate	default	integer	300

**Figura 28-3.** Compruebe "media.peerconnection.enabled" true or false

En la imagen de arriba, se muestra "true". Debe hacer doble clic y hacer que el valor booleano sea "false".

Ahora puede buscar la red privada virtual abierta y gratuita. Recuerde, la gente a menudo compra lo mismo y paga un alto precio por ello. Pero no están seguros todo el tiempo. ¿Por qué no son seguros? Es porque, a veces, cuando la seguridad nacional de un país está bajo ataque y quieren la información, las compañías de servidores tienen que dársela bajo presión. Así que todo este tiempo he tratado de enfatizar una cosa: nunca tratar de ir más allá de la ley. El hacking ético se trata de algo que mantiene estrictamente un único principio: permanecer dentro de la ley.

Aprendes todo para tu defensa personal, no para cualquier tipo de ataque por adelantado. De todos modos, en este capítulo nuestro objetivo principal es cómo podemos ocultar el servidor DNS de nuestro proveedor de ISP.

Hemos buscado sobre la VPN abierta y hemos encontrado "wwwvpnbook.com". Vamos a descargar desde este sitio. En el panel derecho, encontrará el nombre de los proveedores. Varía de vez en cuando. El país desde el que se descargue realmente no importa mientras funcione.

Durante la descarga notará que se proporciona una combinación de nombre de usuario y contraseña. Cópielos y guárdelos en algún lugar, ya que los necesitará cuando ejecute una red privada virtual en su máquina.

En la sección de descargas de tu Kali Linux tienes una versión comprimida de VPN. Descomprímelo primero y luego ejecútalo. ¿Cómo puedes hacer eso? Déjame abrir mi sección "Descargar" de Kali Linux y ver lo que veo.

```
sanjib@kali:~$ cd  
Downloads/ sanjib@kali:  
/Descargas$ ls vpnbook-  
euro1-tcp443.ovpn vpnbook-  
euro1-tcp80.ovpn vpnbook-  
euro1-udp25000.ovpn  
vpnbook-euro1-udp53.ovpn
```

Para obtener la misma salida, debe descomprimir la versión comprimida de su VPN. Ahora emita este comando:

```
openvpn vpnbook-euro1-tcp443.ovpn
```

Si la máquina dijera "openvpn command not found", tendría que instalarlo. Instalar cualquier cosa a través del terminal es bastante fácil en Linux. Busca en la web; hay toneladas de tutoriales que te guiarán al respecto. Normalmente se hace mediante el comando "apt-get".

Cuando intentes ejecutar "openvpn" te pedirá primero el nombre de usuario. Entonces te pedirá la contraseña. Una vez completado este proceso, intentará establecer la conexión. Tienes que esperar un tiempo. A menos que reciba un mensaje, "inicialización completa", no podrá abrir su navegador. Puede tomar varios minutos. Usualmente toma dos minutos como mínimo.

Si no tienes suerte, puede que pase algún tiempo, no siempre, por supuesto. Este mensaje no aparecerá. En ese caso, dice, "la conexión falló".

Una vez que recibas el mensaje "inicialización completa", puedes abrir el navegador y buscar en "www.duckduckgo.com". Este motor de búsqueda generalmente no rastrea el registro del usuario.

Su primer trabajo será comprobar la fuga de DNS. Vaya a por ello y definitivamente encontrará un cambio de dirección IP.

Esto significa que se ha conectado con éxito a través de la red privada virtual y su servidor DNS ISP original está completamente oculto.

## 29. Dirección MAC

Sanjib Sinha 

(1) Howrah, Bengala Occidental, India

---

Hasta ahora hemos aprendido muchos trucos, todos sobre el anonimato. Pero siempre intentaremos ir a un nivel superior. Cambiar la dirección MAC entra dentro de esa categoría.

De una manera simple, es su dirección de hardware. Básicamente, no es la dirección de hardware de su máquina, sino la dirección de hardware de su tarjeta de red a través de la cual se conecta con el mundo exterior.

Pongamos en marcha nuestra máquina virtual Kali Linux y abramos el terminal. Emite el comando: ipconfig.

Producirá algo como esto:

```
root@kali:~#  
ifconfig eth0:  
flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu  
10.0.2.15          máscara de red1500      inet  
10.0.2.255         inet6 e80::a00:27ff:fef4:16ec  
                      prefijo de255.255.255.0  
difusión64         scopeid 0x20<link>  
                      éter08:00:27:f4:16:ec      txqueuelen  
19      Bytes de paquetes de1000      (Ethernet) RX19  
1820 (1.7  
KiB) Los errores de recepción0      han dejado  
caer0              los excesos de0      fotogramas  
31      Bytes de paquetes0      TX31      2427  
(2.3 LosKiB)          errores de TX0      cayeron  
0 sobrepasan a la0      portadora 0 colisiones 0  
  
lo:  
flags=73<UP,LOOPBACK,RUNNING> mtu
```

```
127.0.0.1    máscara de red de red de red255.0.0.0
de red de
red de red de red de red de red de red de red de red
de red de red de red de red de red de red de red de
red de red de red de red de red de red de red de red
de red de red de red de red de red de red de red de
red de red de red de red de red de red de red de red
red de red de red de red
::1 prefijo del128 scopeid
0x10<host>  loop txqueuelen0      (Local Loopback) RX
packets36    bytes 2160 (2.1 Errores deKiB)
RX
0 rebasamiento de paquetes de0          transmisión
de0           tramas a la0            baja
36 bytes 2160 (2.1 Los errores deKiB)
transmisión0          cayeron 0 sobrepasan
las0           colisiones de0        la
portadora 0
```

En su caso, la salida podría ser diferente. Nos preocupa la dirección de hardware de nuestra red y queremos cambiarla.

En el medio, has visto la línea roja que dice: **éter 08:00:27:f4:16:ec**

Esta es la dirección MAC de la máquina virtual Kali Linux o la dirección de la tarjeta de red local . Ahora bien, en algunos casos podría ser así:

**HWaddr** 08:00:27:f4:16:ec

En algunos casos es diferente. Son tarjetas de red. Podrían ser tarjetas Ethernet, tarjetas inalámbricas, adaptadores inalámbricos, etcétera.

Pero esta dirección es extremadamente importante, ya que se utiliza para identificarle en el vasto mundo de la web. Los tres primeros dígitos son los símbolos que representan al fabricante.

Podemos comprobarlo aquí también emitiendo este comando:

```
root@kali:~# macchanger -s eth0
CorrienteMAC: 08:00:27:f4:16:ec (CADMUS
COMPUTER SYSTEMS)
MAC permanente: 08:00:27:f4:16:ec (CADMUS COMPUTER
SYSTEMS)
```

Como puede ver, muestra dos direcciones MAC: una es actual y la otra es permanente. Se preguntarán por qué estoy revisando esto. Lo he comprobado una vez emitiendo el comando "ifconfig". ¿No es suficiente?

Es porque el comando "ifconfig" sólo mostrará la dirección MAC actual. No mostrará la dirección MAC permanente. Esto significa que cuando ha cambiado la dirección MAC y ha emitido el comando "ifconfig", sólo

muestra la dirección cambiada, no la permanente.

Ahora nos gustaría cambiar nuestra dirección MAC. Emitamos este comando:

```
root@kali:~# macchanger -h
```

Y producirá una salida como esta:

## Cambiador de MAC de GNU

Uso: dispositivo macchanger [opciones]

-h, --ayuda	Imprimir esta ayuda
-V, Versión --salida	Imprimir versión y versión
-s, --espectáculo dirección y salida	Imprimir el MAC
-e, --ending los bytes del proveedor	No modifique
-a, --another	Establecer el MAC de un proveedor aleatorio del mismo
simpático	
-A	Fijar proveedor-A
aleatorio	
MAC de cualquier tipo	
-p, --permanent hardware original y permanente	Restablecer el MAC
-r, --random al azar	Fijar completamente
MAC	
-l, --list[=keyword] proveedores-l, conocidos	Imprimir --list[=keyword]
-b, --bia quemado en la dirección	Fingir ser un
-m, --mac=XX:XX:XX:XX:XX:XX	
--MAC XX:XX:XX:XX:XX:XX	

### Informar de errores a

<https://github.com/alobbs/macchanger/issues>

Las tres líneas rojas son importantes. Se define explícitamente lo que significan. La línea verde también es importante.

Las primeras dos líneas- -a, --otro Establecer MAC de vendedor aleatorio del mismo tipo

#### -Un MAC de vendedor aleatorio de cualquier tipo

-lo que significa que puedes cambiar la dirección MAC pero no puedes cambiar el proveedor. En este caso, existe la posibilidad de perder su anonimato. Los tres primeros conjuntos pertenecen al fabricante de la tarjeta de red y, puesto que no se ha modificado, se puede identificar.

La tercera línea de color rojo es bastante obvia y autoexplicativa en su significado. Dice: puedes volver a cambiar a la dirección MAC original.

Hasta ahora, la mejor opción disponible para nosotros es la línea de color verde **-r, -- random** **Set fully random MAC-** donde se dice claramente que se puede establecer un MAC completamente aleatorio. Es decir, los seis conjuntos son completamente aleatorios, lo que preferimos.

La más importante de ellas es la última línea azul. ¿Por qué es importante? Esto se debe a que puede cambiar la dirección MAC completamente.

Podemos tener una lista de todos los proveedores con un comando simple: 1. Si usted emite ese comando le dará una lista muy larga. Vamos a recoger a algunos de ellos.

```
root@kali:~# macchanger -  
  
1 Misc MACs:  
  
Num      MAC            Proveedor  
---      ---            -----  
  
0000 - 00:00:00 - XEROX CORPORACIÓN  
0001 - 00:00:01 - XEROX CORPORACIÓN  
0002 - 00:00:02 - XEROX CORPORACIÓN  
0003 - 00:00:03 - XEROX CORPORACIÓN  
0004 - 00:00:04 - XEROX CORPORACIÓN  
0005 - 00:00:05 - XEROX CORPORACIÓN  
0006 - 00:00:06 - XEROX CORPORACIÓN  
0007 - 00:00:07 - XEROX CORPORACIÓN  
0008 - 00:00:08 - XEROX CORPORACIÓN  
0009 - 00:00:09 - XEROX CORPORACIÓN  
0010 - 00:00:0a - OMRON TATEISI ELECTRÓNICA CO.  
0011 - 00:00:0b - MATRIX CORPORATION
```

**0012 - 00:00:0c - CISCO SYSTEMS,  
INC.**

**0013 - 00:00:0d - FIBRONICS LTD.**

**0014 - 00:00:0e - FUJITSU LIMITED**

**0015 - 00:00:0f - NEXT, INC.**

**0016 - 00:00:10 - SYTEK INC.**

**0017 - 00:00:11 - SISTEMAS  
NORMALIZADOS**

**0018 - 00: 00:12 - TECNOLOGÍA DE LA INFORMACIÓN  
LIMITADA**

**0019 - 00:00:13 - CAMEX**

Hemos tomado las primeras líneas - diecinueve en este momento. Pero la última es de **19010**.

- **fc:fe:77 - Hitachi Reftechno, Inc.** El número de color rojo muestra cuántos hay en total. La lista no está completa. Después de eso, hay direcciones MAC inalámbricas. En total hay alrededor de treinta y nueve.

Usted puede preguntar cuáles son en realidad. No son más que las partes de la dirección MAC de la empresa. Consideremos el último ejemplo: **0019 - 00:00:13 - CAMEX.**

El primero es el número de serie. La segunda es la dirección MAC. Puede cambiar su dirección de proveedor y utilizar esta y fingir que está utilizando esta empresa. Los hackers éticos a veces usan ese truco.

Teniendo todo en mente, me gustaría decir que la última opción -la de color azul- es la más importante.

En las universidades, los estudiantes a veces usan ese truco para engañar al profesor, junto con toda la clase. Alguien toma la dirección MAC del profesor y, fingiendo ser el PC del profesor, interfiere con la red. Una vez que la red se ha atascado, el profesor ya no puede tomar la clase.

Por lo general, hay un sistema de filtrado de red que descubre la dirección MAC falsa y bloquea esa dirección. Pero eso también es divertido. Cuando el sistema de filtrado de la red ha bloqueado la dirección MAC, se descubre que el PC del profesor ha sido bloqueado inadvertidamente.

Como un hacker ético usted necesita estudiar esta parte en particular, ya que los hackers maliciosos a menudo usan la dirección MAC de otra máquina y pretenden ser alguien mientras hacen las cosas equivocadas.

## # Epílogo - ¿Qué sigue?

Gracias por leer este volumen de *Ethical Hacking with Python 3*. Espero que, como principiante, hayas aprendido lo básico del hacking ético. Esto incluye los términos, el lado legal y el propósito; el trabajo en red, el medio ambiente y una introducción detallada sobre el anonimato. Además, espero que tengas un conocimiento práctico de Python 3.

Los próximos volúmenes de libros sobre Ethical Hacking tratarán conceptos más avanzados como "Nmap", "SQL Injection", "Denial of Service or DOS", "Brute Force Method", "Signal Jamming", "Password Cracking", "Footprinting with Nmap", "Attacking Wireless Networks", "WiFi Hacking, Breaking Encryptions", "SSL Strips" y muchos más.

Espero que nos encontremos en el próximo libro. Hasta entonces, la **mejor de las suertes**.

## Epílogo-¿Qué sigue?

Gracias por leer este volumen de *Ethical Hacking with Python 3*. Espero que, como principiante, hayas aprendido lo básico del hacking ético. Esto incluye los términos, el lado legal y el propósito; el trabajo en red, el medio ambiente y una introducción detallada sobre el anonimato. Además, espero que tengas un conocimiento práctico de Python 3.

Los próximos volúmenes de libros sobre Ethical Hacking tratarán conceptos más avanzados como "Nmap", "SQL Injection", "Denial of Service or DOS", "Brute Force Method", "Signal Jamming", "Password Cracking", "Footprinting with Nmap", "Attacking Wireless Networks", "WiFi Hacking, Breaking Encryptions", "SSL Strips" y muchos más.

Espero que nos encontremos en el próximo libro. Hasta entonces, **la mejor de las suertes.**

---

## Índice

### A, B

Métodos  
accesorios  
AMD64  
Protección del anonimato, Internet

### C

Formato de cadena  
de clase()  
función  
inmutable  
lower()  
ubicación  
reemplazar y  
encontrar() strip()  
upper()  
Comunicación  
Ejecuciones  
condicionales  
Valores/expresiones condicionales  
Configuración Módulo de análisis  
Contenedores

codificadora

objeto de lista de  
objetos del  
diccionario  
tuplas  
de  
salida  
Criminales de  
tarjetas de  
crédito/débito  
Crear, recuperar, actualizar y eliminar  
(CRUD) Ciberleyes

## D

Web oscura/red  
profunda  
Datagrama  
Decorador  
es de  
Debian

## E

Función enumerate()  
comunicación de extremo a  
extremo Hacking ético  
Excepciones

## F, G

Entrada y salida de archivos  
Función del sistema de  
comprobación del bastidor  
(FCS)

Otra codificación  
deFunction()  
DemarcationLine()  
genera listas de  
funciones de  
argumentos con  
nombres de  
argumentos que pasan  
los valores por

defecto.  
parámetros/argumentos de  
aprobación RangeFunctions()  
Retorno de los  
valores de  
reutilización  
TestFunction()

# H

## Hacking

ordenadores con métodos de ataque,  
explotación/penetración

delincuentes con  
número de tarjeta de  
crédito/débito  
sistemas operativos

anónimos del  
entorno  
lenguaje de programación  
máquina virtual

máquina virtual

## Wikipedia oculta

páginas wiki ocultas  
mercado interesante

Kali Linux

arquitectura del  
sistema terminal Tor  
navegador  
torproject.org sitio  
web

## Trata de personas

# I, J

## Herencia

Organización de Normas de Internet  
(ISO) Conexión a Internet

# K

## Kali Linux

# L

Distribución  
Linux Terminal  
Linux  
anonimato de  
comandos de  
adduser  
comando de gato

```
cat sources.list | grep src
```

cd command  
chmod command  
línea de comandos  
herramienta/terminal cp  
comando  
cp-help  
comando echo de  
directorios y carpetas  
hacking ético  
modo ejecutable  
archivo de  
permisos grep  
comando grupo  
de comandos  
comando  
Iceweasel  
Kali Linux vista de  
pantalla completa cerrar  
sesión  
ls command ls-  
la command  
mkdir command  
mv command  
Nano text editor  
output  
pwd comando  
pyfile.py  
Archivo  
Python  
rf command  
rm command  
root/super user  
r-x command  
sanjib  
sources.list  
sudo command  
Función de bucles de  
dirección de tarjeta de red  
local()

M

Dirección MAC

comando  
hackers éticos

ifconfig  
ipconfig  
tarjeta de red  
salida del sistema de  
filtrado de red  
Ataque malicioso  
Control de acceso a los medios (MAC)  
Módulo  
  programa de cálculo  
  Distribución de Linux  
  Debian  
  os/programa de módulos específicos del  
  sistema operativo  
  solicitud aleatoria y de  
  fecha y hora y respuesta  
  urllib de módulo específico  
  del sistema/sistema  
MySQL  
  conector del  
  archivo de  
  configuración de  
  codificación  
  Conexión a la base  
  de datos de la  
  aplicación CRUD  
  eliminar un registro  
  fetchall()  
  fetchmany()  
  archivo.ini  
  PHPMyAdmin  
  python-mysql  
  bases de datos  
  relacionales  
  recuperar registros  
  setup.py  
  pasos cortos  
  de BLOB  
  método de prueba y  
  error escribiendo  
  archivos  
Módulo conector MySQL

N

Características de  
la red  
interoperabilidad de las  
funciones de  
comunicación  
ingeniería modular  
que comparte el  
zócalo de recursos  
tipos

Tarjetas de  
red

O

Objeto

Programación orientada a objetos  
(OOP) que accede a los datos del  
objeto  
lucro  
codificación de  
argumentos de  
clases y objetos  
instancias del proceso  
de inicialización de la  
definición de  
argumentos por  
defecto  
metodologías  
MyMySQLConnection() y MySQLiteConnection()  
Salida de la base  
de datos MySQL

Decorador de mecanismo de  
herencia de clase  
descripción  
juego, buenos *vs.*  
malos generadores  
objetivo  
herencia  
interrelaciones e interacciones de  
datos objeto  
aplicaciones de  
software de  
polimorfismo de

*objeto vs. clase*

método  
open() de la  
página web  
Capa de aplicación de interconexión  
de sistemas abiertos (OSI)  
capa de enlace de  
datos capa de red  
capa de  
presentación capa  
física capa de  
sesión capa de  
transporte

## P, Q

Filtrado de  
paquetes  
Comutación de  
paquetes  
Selección de ruta  
Polimorfismo  
Proxies  
Archivo de  
configuración de  
cadenas proxy  
Prueba de fuga  
de DNS  
Solicitudes de  
DNS  
Documentación  
dynamic\_chain  
Dirección IP  
listar  
cadena\_al  
azar  
cadena\_estrict  
a tor  
tipos  
Python  
asignando  
valores  
codificando  
comentarios en

el diccionario de  
condicionales  
para bucles  
instaladores de hendiduras  
y espacios en blanco

valores lógicos  
números de  
función main()  
salida de  
objeto  
Función  
OutsideMainFunction()  
Función  
OutsideMainFunction  
print() Función print()  
ristra  
tuplas y listas  
de tipo e ID  
durante los  
bucles  
Comando de  
codificación  
Python 3  
página de  
documentación IDE  
IZQUIERDA  
intérprete de  
instrucciones  
Distribución de Linux  
salida de lenguaje de programación  
de código abierto  
Monitorización del sistema  
de configuración del  
sistema Pycharm  
Community Edition  
Plataforma  
Windows/Macintosh Python  
Standard Library

## R

range() Expresiones

regulares

reutilizar el

módulo

reutilizando la

búsqueda

Enrutadores

S

Segmento  
Efecto  
secundario  
Enchufe  
Instalación del  
software SQLite3  
Interruptores  
SyntaxError

## T

Protocolo de control de transmisión/Protocolo de Internet  
(TCP/IP) Bloque de prueba  
Tuplas  
TipoError

## U

Ubuntu  
Centro de software de  
Ubuntu uname-a  
Mensaje de error  
    del módulo de  
    prueba de  
    unidad  
    assertEqual()  
MyTest/BrainAndSoul  
PyCharm IDE  
saytimedate.py errores  
    sintácticos  
test\_PyVar() y test\_main()  
test\_Time() y test\_Version()  
TestUnitTest.py

## V

VBoxLinuxAdditions.run Caja  
Virtual Adiciones Caja Virtual  
de Invitados (VB)  
    lucro  
    apt-get update  
    apt-get upgrade  
Arquitectura de 32 bits/64 bits

sección de descargas, hosts Linux  
Archivos.exe  
tamaño de  
pantalla  
completa vista  
de pantalla  
completa  
pruebas de herramientas de hacking  
proceso de instalación, Kali  
Linux instala Windows 7  
Ultimate Internet connection  
Imagen ISO  
Kali Linux corriendo, Oracle  
VM tamaño de la memoria  
paquete de  
métodos del  
sistema  
operativo  
procedimiento de la  
herramienta de  
ataques de  
contraseña  
Red Hat/Fedora  
códigos de  
ejecución  
autoexplicativos  
en la sección de  
almacenamiento  
de su terminal de  
comandos y tipo  
de terminal  
imagen x86\_64  
Red privada virtual (VPN) sobre:config  
image, anonimato del navegador Mozilla  
apt-get  
command  
connection  
Archivo  
dhclient.conf  
Servidor DNS  
ISP proveedor  
media.peerconnection.enabled

nameserver  
el administrador  
de red aplica el  
principio  
openvpn  
versión comprimida

W, X, Y, Z

Windows XP