

Python para Hackers



Python para Hackers

Contenido del curso

Capítulo 1. Introducción

Capítulo 2. Primeros pasos

Capítulo 3. Python

Capítulo 4. Hands-On

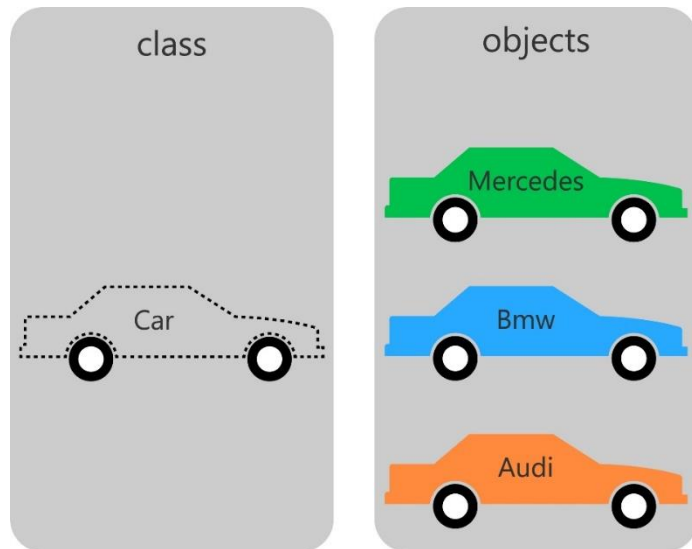
Tema de hoy

Ataques de diccionario | Web Server | Web Scraping | Banner Grabbing | Reconocimiento de máquinas | Servidor/Cliente TCP | Packet Sniffing con Scapy | Paramiko (Cliente SSH) | Nmap con Python | MacChanger | Fuerza bruta de Directorios Web | Fuerza bruta a formularios de autenticación (web)



Clases

Clases

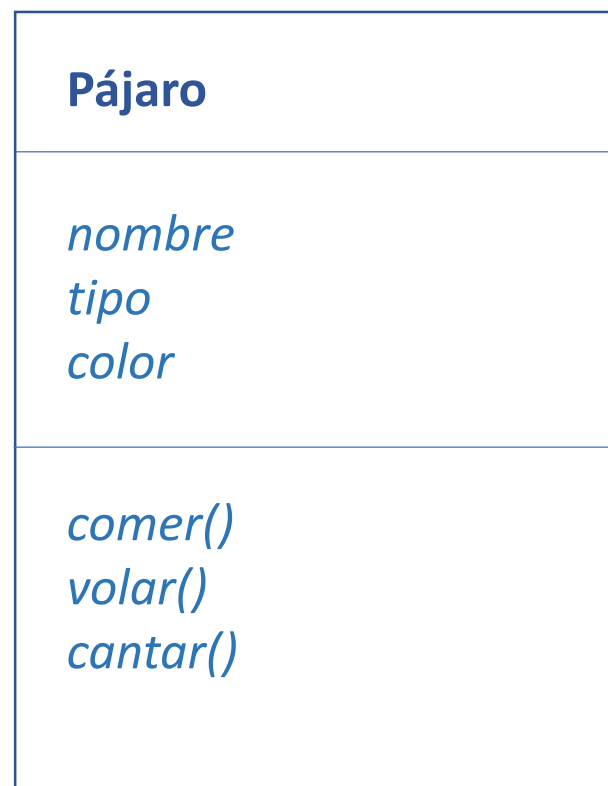
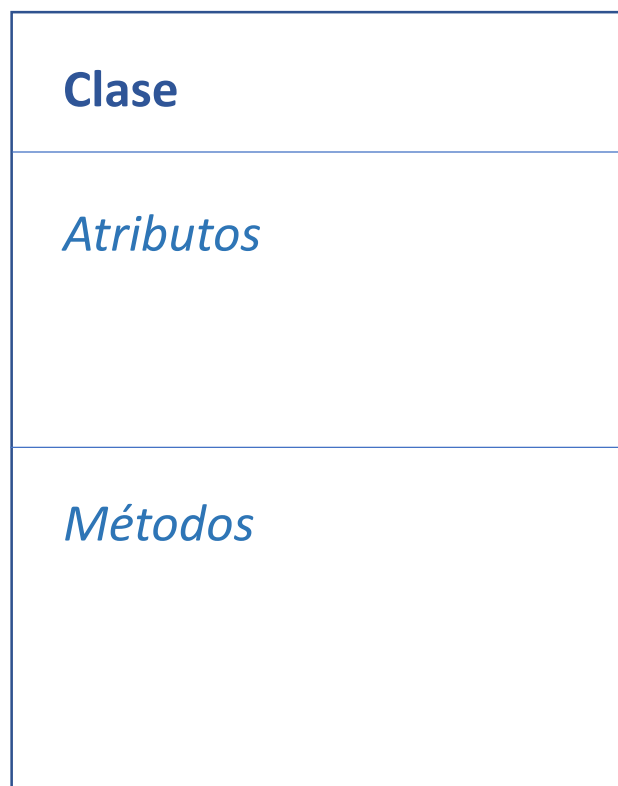


Una clase es un **plano** que describe como está formado un objeto.

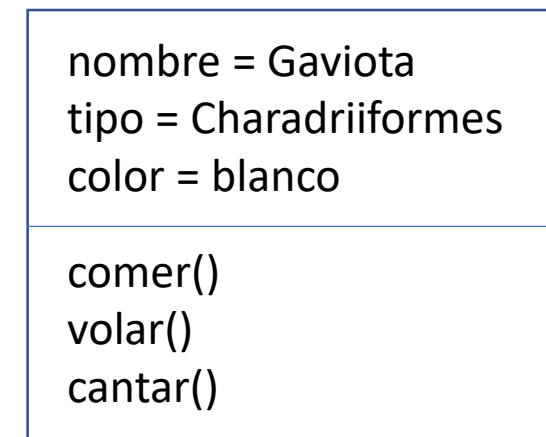
En Python todo es un objeto, por eso podemos utilizar métodos y atributos.

Clases

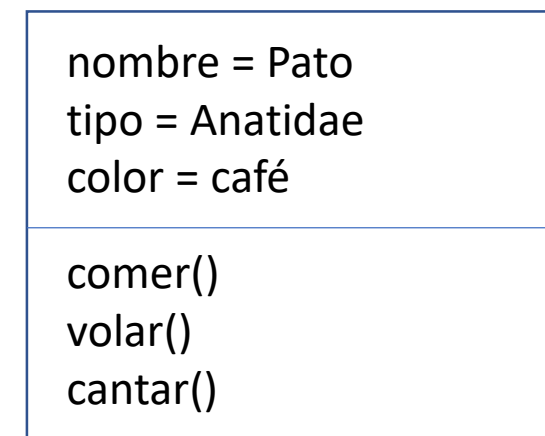
Estructura de una clase



Objeto Pájaro1



Objeto Pájaro2



Clases

Estructura de una clase

Palabra reservada.
Indica que una clase
va a comenzar.

Nombre de
la clase

`__init__` : Método especial de las clases también conocido como *constructor*. Se utiliza cuando se inicializa o construye un objeto basado en esa clase.

```
01. class Vehiculo:
02.
03.     def __init__(self, color, marca, anio):
04.         self.color = color
05.         self.marca = marca
06.         self.anio = anio
07.
08.     def conducir(self):
09.         print("Estoy conduciendo un " + str(self.marca) + " color " + str(self.color))
```

Constructor

Método

Clases

Estructura de una clase

Palabra reservada.
Indica que una clase
va a comenzar.

Nombre de
la clase

self: Palabra reservada que utilizan las clases para referirse a si mismas y para diferenciar una instancia de otra.

```
01. class Vehiculo:
02.
03.     def __init__(self, color, marca, anio):
04.         self.color = color
05.         self.marca = marca
06.         self.anio = anio
07.
08.     def conducir(self):
09.         print("Estoy conduciendo un " + str(self.marca) + " color " + str(self.color))
```

Constructor

Método

Clases

Estructura de una clase

Palabra reservada.
Indica que una clase
va a comenzar.

Nombre de
la clase

self.variable: Son los atributos que va a tener el objeto una vez inicializado. Deben de ir especificados en el constructor de la clase.

```
01. class Vehiculo:
02.
03.     def __init__(self, color, marca, anio):
04.         self.color = color
05.         self.marca = marca
06.         self.anio = anio
07.
08.     def conducir(self):
09.         print("Estoy conduciendo un " + str(self.marca) + " color " + str(self.color))
```

Constructor

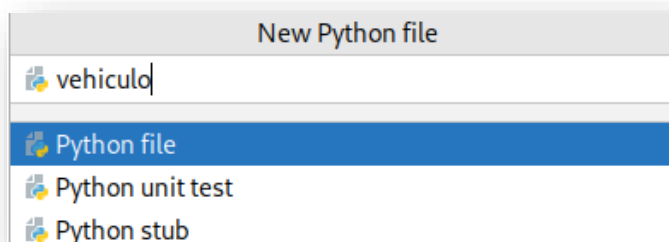
Método

Clases

Laboratorio



Crear un nuevo archivo en PyCharm que se llame “vehiculo”.



Programar la clase Vehiculo.

```
1 class Vehiculo:
2
3     def __init__(self, color, marca, anio):
4         self.color = color
5         self.marca = marca
6         self.anio = anio
7
8     def conducir(self):
9         print("Estoy conduciendo un " + str(self.marca) + " color " + str(self.color))
```

Clases

Laboratorio



Código:

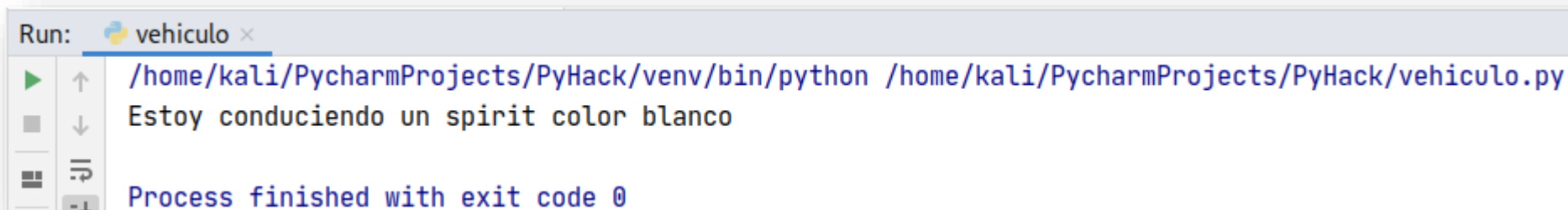
Fuera de la clase Vehiculo, escribir las siguientes líneas de código:

```
11 v = Vehiculo("blanco", "spirit", "2017")
12 v.conducir()
```

Se está creando una instancia de la clase Vehiculo.

Se llama al método conducir().

Ejecución del código



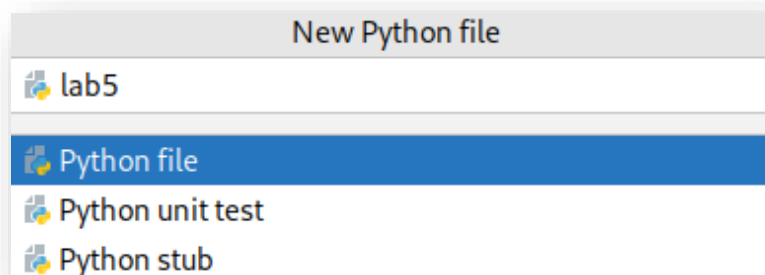
```
Run: vehiculo x
/home/kali/PycharmProjects/PyHack/venv/bin/python /home/kali/PycharmProjects/PyHack/vehiculo.py
Estoy conduciendo un spirit color blanco
Process finished with exit code 0
```

Clases

Laboratorio: Importar una clase



Crear un nuevo archivo en PyCharm que se llame “lab5”.



Escribir las siguientes líneas de código:

```
1 from vehiculo import Vehiculo
2
3 carro = Vehiculo("negro", "Toyota", "2016")
4 carro.conducir()
```

Clases

Laboratorio: Importar una clase



Ejecución del código

```
Run: lab5 x
/home/kali/PycharmProjects/PyHack/venv/bin/python /home/kali/PycharmProjects/PyHack/lab5.py
Estoy conduciendo un spirit color blanco
Estoy conduciendo un Toyota color negro
```

También se ejecuta el código que está fuera de la clase en el archivo vehiculo.py.

Para evitar que eso pase:

Modificar el código en el archivo “vehiculo.py” para que quede de la siguiente manera:

```
11 ▶ if __name__ == "__main__":
12     v = Vehiculo("blanco", "spirit", "2017")
13     v.conducir()
```

Indica a Python que el código dentro del “if” sólo se ejecutará cuando el archivo se ejecute individualmente.

Clases

Laboratorio: Importar una clase



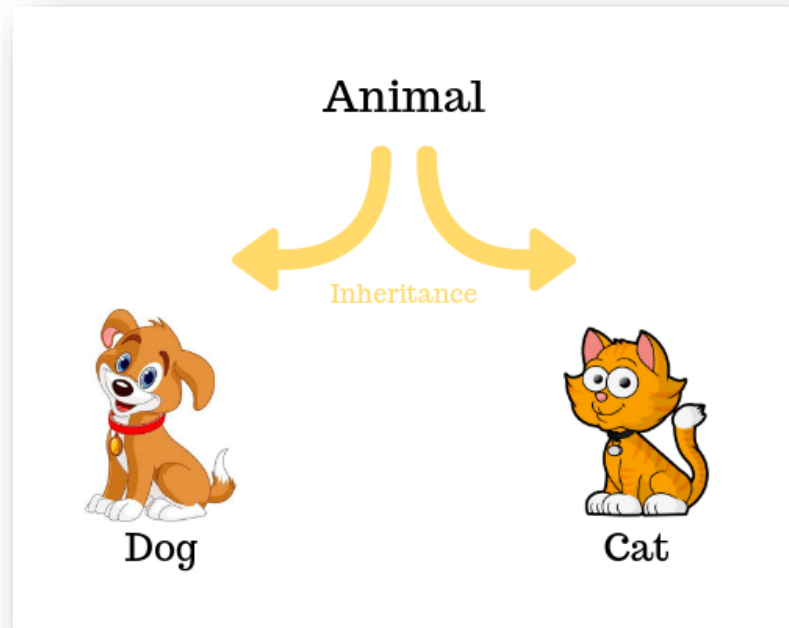
Ejecución de código

Ejecutar el archivo "lab5.py"

```
Run: lab5 x
▶ /home/kali/PycharmProjects/PyHack/venv/bin/python /home/kali/PycharmProjects/PyHack/lab5.py
■ Estoy conduciendo un Toyota color negro
```

Herencia

Herencia



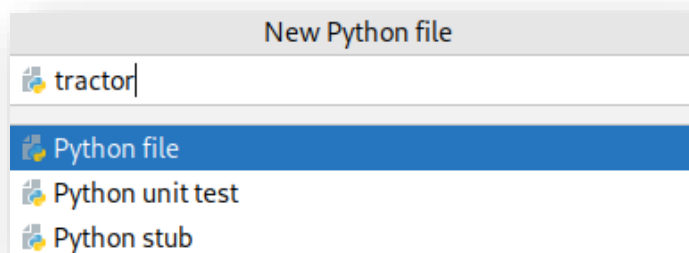
Permite a una clase **utilizar los métodos y atributos de otra clase** (clase padre).

Herencia

Laboratorio



Crear un nuevo archivo en PyCharm que se llame “tractor.py”.



Programar la clase Tractor.

```
1  from vehiculo import Vehiculo
2
3  class Tractor(Vehiculo):
4      pass
5
6  t = Tractor("verde", "John Deere", "2000")
7  t.conducir()
```

class ClaseHija(ClasePadre): Así se define que la clase hija va a heredar todos los métodos y atributos de la clase padre.

pass: Palabra reservada que se utiliza cuando no se quiere agregar métodos o atributos a la clase.

Herencia

Laboratorio



Ejecución del código

```
Run: tractor x
/home/kali/PycharmProjects/PyHack/venv/bin/python /home/kali/PycharmProjects/PyHack/tractor.py
Estoy conduciendo un John Deere color verde

Process finished with exit code 0
```

Servidor Web

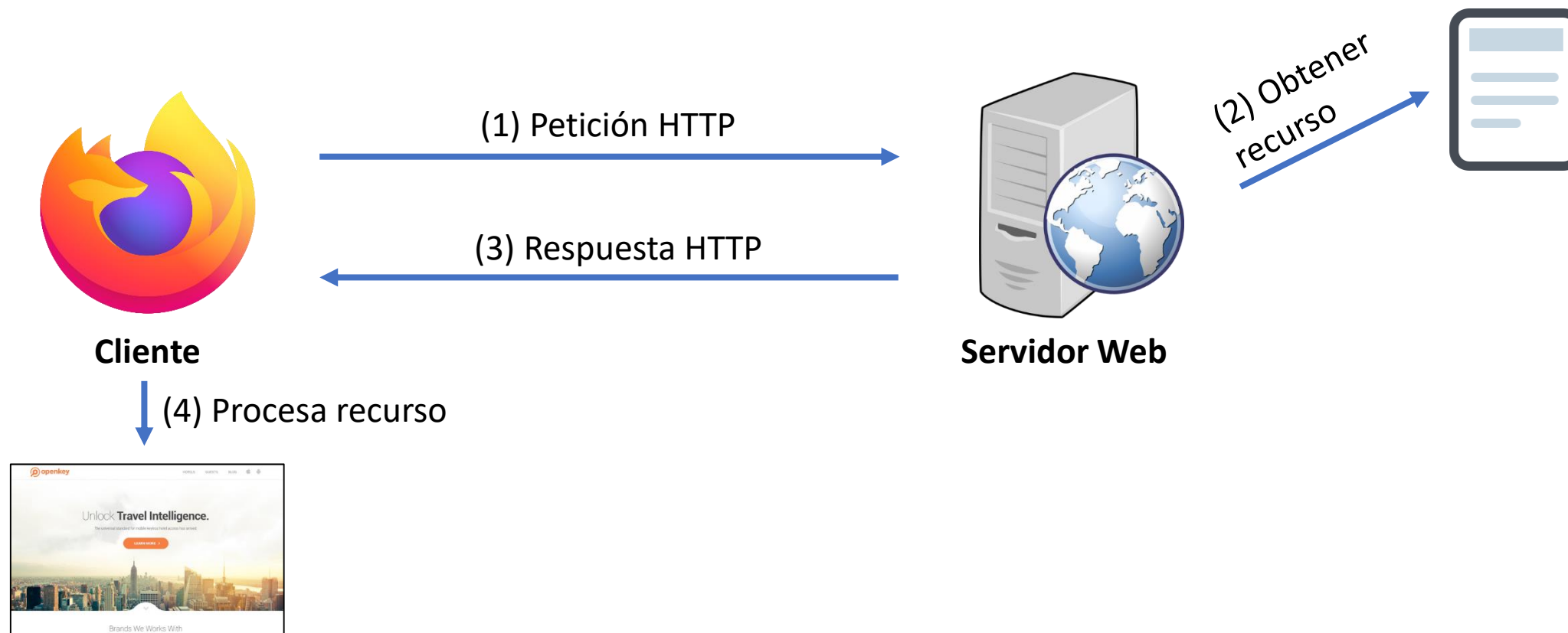
Servidor Web

Es un software que **procesa peticiones HTTP** (en una dirección IP y puerto en específico) enviadas por un cliente.



Servidor Web

Diagrama



Servidor Web

Penetration Testing y Python

Un servidor web es muy útil en el pentesting ya que ayuda al hacker a **descargar exploits o herramientas** en un cliente comprometido.

Python cuenta con **librerías** que permiten crear un servidor web sencillo donde cualquier directorio del sistema operativo se convierte en un servidor web.

Web Server Script

WebServer1.py

```
WebServer1.py x
1 import socketserver
2 import http.server
3
4 httpServer = socketserver.TCPServer(("", 20000), http.server.SimpleHTTPRequestHandler)
5 httpServer.serve_forever()
```

1 *socketserver* es una librería de Python que permite generar servidores de red. Define clases para manejar peticiones de red sobre TCP y UDP.

http.server es una librería de Python que permite generar servidores HTTP.

2 *TCPServer* es un método que genera un socket TCP/IP para la comunicación entre el cliente y el servidor.

`TCPServer(server_address, RequestHandlerClass)`

server_address: Tupla que contiene una cadena con la dirección IP del servidor y un entero que especifica el puerto a usar.

RequestHandlerClass: Clase que se utiliza para responder la petición HTTP.

WebServer1.py

```
WebServer1.py x
1 import socketserver
2 import http.server
3
4 httpServer = socketserver.TCPServer(("", 20000), http.server.SimpleHTTPRequestHandler)
5 httpServer.serve_forever()
```

3 *serve_forever()* método que comienza a procesar peticiones. Inicia el servidor.

WebServer2.py

```
WebServer2.py x
1 import socketserver
2 import http.server
3
4 class HttpRequestHandler(http.server.SimpleHTTPRequestHandler):
5
6     1 def do_GET(self):
7         2 if self.path == '/admin':
8             self.wfile.write(b'Esta pagina es solo para administradores!\n')
9             self.wfile.write(self.headers.as_bytes())
10        else:
11            3 http.server.SimpleHTTPRequestHandler.do_GET(self)
12
13
14 httpServer = socketserver.TCPServer(("", 20002), HttpRequestHandler)
15 httpServer.serve_forever()
```

1

Se sobrescribe el método do_GET()

2

Si el path contiene el directorio /admin enviará como respuesta un mensaje y las cabeceras HTTP del cliente.

3

De lo contrario llama al método do_GET() original.

Próxima clase...

- **Capítulo 4:** Hands-On (Parte III)
- **Evaluación** (Capítulo 3)



NEXT >>
Jue, 27 Ago



CloudLamb

**¡Muchas gracias por su
atención!**