# Hacking con Python

Hacking Ético

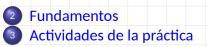
©Ismael Ripoll & Hector Marco

Universidad Politècnica de València

February 1, 2022

# Índice





# Qué vamos a trabajar

- "Repaso" de programación básica en Python. Asumo que ya sabes algo de Python.
  - CUIDADO: Esto no es un tutorial de Python.
  - Como informático que vas ha ser debes ser capaz de utilizar un lenguaje de programación en menos de 2 días.
  - Utilizar significa:
    - Poder leer, comprender y hacer pequeños cambios a código de terceros.
    - Poder escribir programas que resuelvan problemas simples.
- Búscate un tutorial de Python y estudia los conceptos seleccionados y los módulos necesarios para hacer la práctica.
- El objetivo es familiarizarte con Python para pode hacer las siguientes prácticas.

# Tipos de datos avanzados

Antes de nada: utiliza ipython para probar directamente la sintaxis de python.

- No es necesario declarar las variables.
- Hay cadenas, enteros, flotantes, imaginarios.
- Tipos de datos avanzados:

**Tupla:** lista elementos inmutable.

**Lista:** lista de elementos.

**Diccionario:** conjunto de parejas (key, valor).

**Conjunto:** Conjunto de elementos sin repeticiones y no

ordenado.

Cuidado con las funciones que manipulan estos objetos.
 Algunas crean una copia y otras los modifican.

## Cadenas

→ Hay muchas formas de introducir una cadena de caracteres:

```
Cadena1= "Tipico con doble comilla\n";
Cadena2= 'Con comillas simples\n';
Cadena3 = """Tres comillas dobles
  permite introdudir texto con retornos de carro
  fácilmente.""";
Cadena4 = r"Cadena 'raw' no interpreta los escapes.";
```

Fíjate que se pone una "r" pegada al inicio de la cadena para indicar que es raw. Se puede usar la r con cualquier sintaxis de cadena: doble comilla, simple comilla y triple doble comilla.

## Estructuras de control

#### **Bucles:**

```
for a in range(0,100):
    print a;
lista=[1,2,3,4,5,'a','4']
for i in lista:
    print(i);
counter=0
while counter < 10:
    print(counter);
counter += 1;</pre>
```

#### **Condicionales**

```
if counter > 10:
    print("si, es mayor");
else:
    print("no, no es mayor");
```

## **Funciones**

Funciones estandar:

```
def mi_funcion(a,b,c):
    print("los parametros son",a,b,c);
    mi_funcion(1,2,3);
```

→ lambda, map, filter. Se puede vivir sin estas funciones pero os las podéis encontrar en código de otros.

```
ascii=map(lambda ch: chr(ch), range(32,0x7f));
print(ascii);
```

La sintaxis en un tanto extraña pero hay que entenderla.

### Clases

- Junto con las drogas y los videojuegos, el abuso de la programación orientada a objetos es uno de los problemas sociales más acuciantes<sup>1</sup>.;-)
- → Los métodos y las atributos se pueden añadir dinámicamente.

```
class monedero:
   comun = [];  # Variable global de clase.
   def __init__(self, dinero): # Inicializador
      self.bolsa = dinero; # Variable local de objeto
   def ahorra(self, dinero):
      self.bolsa += dinero;
   cartera = monedero(100);
   cartera.ahorra(200);
   print (cartera.bolsa);
```

- No hay atributos y métodos privados.
- •> Se puede hacer herencia... y cosas peores!.

<sup>&</sup>lt;sup>1</sup>Hay que ser muy crítico con todo.

## Excepciones

- Aquí si que tenemos un problema.
- → En cualquier momento se puede emitir (raise) una excepción cuando se produce algún error.
- Se utiliza la construcción sintáctica try except

```
try:
   fd=open("no_existe.txt"); # Pueden emitir excepciones.
   lines= fd.readlines();
except IOError as bad: # except [class name] as [var name]
   print("Sorry, no existe no_existe.txt.'', IOError);
```

- → Las excepciones (lo que se raisea) tienen que ser una clase.
- → ¿Te has planteado alguna vez si las excepciones son una buena idea? Hazlo ahora. ¿Has pensado cómo el compilador o interprete las implementan?

## Módulos

- Son las librerías que contienen las funciones y las clases implementadas por terceros.
- Los tipos de datos avanzados y la gran cantidad de módulos disponibles es lo que hace de Python un lenguaje atractivo.

```
import sys;
print(sys.argv);
# Importar uno de las cosas del módulo a nuestro espacio.
from sys import argv;
print (argv);
# Importar TODO a nuestro espacio... peligroso.
from sys import *;
print (argv);
```

CUIDADO: Cuando importas un modulo con from sys import
 \*, todos los símbolos se reemplazan por los de nuevo módulo.

## Actividades a realizar (I)

- Uso básico de Python:
  - Escribe micro scripts donde se demuestre que sabes crear variables de todos los tipos y puedes operar con ellas con bucles y condicionales.
  - Aprende a depurar tu código usando "print".
- Utilización de la E/S:
  - Escribe un script que reciba un parámetro que sea el nombre de un fichero, e imprima por la salida estándar su contenido.
- Utilización de sockets (I):
  - Utilizando el modulo sockets crea un programa que obtenga e imprima por pantalla la página web principal de la UPV.
  - Tienes que crar un socket, conectarte a la upv y realizar una petición HTTP.
  - NO utilices otros módulos. Tienes que implementar el protocolo HTTP "a mano".

## Actividades a realizar (II)

- Utilización de sockets (II):
  - Construye un sevidor web que conteste con saludo y el nombre de la URL pedida, así como la identidad del browser (la cabecera "User-Agent:").
- Codificación base64:
  - Utiliza el programa base64 para codificar el fichero /etc/passwd,
     y copia el resultado en un fichero local.
  - Escribe un script, que haciendo uso del módulo base64 de python, liste en claro en fichero que acabas de crear.
- Otilización del módulo struct (función pack):
  - ► Tener un control preciso de los bytes que se transfieren nos va a permitir construir shellcodes como cadenas de caracteres.
  - Escribe un script que reciba un numero e imprima por la salida estándar ese número codificado como uno, dos y 4 bytes.
  - Para comprobar los resultados utiliza la orden hd.