

# Desarrollo de appliances y sistemas embebidos utilizando Python.

José A. Rocamonde

[jrocamonde@gmail.com](mailto:jrocamonde@gmail.com)



# ¿ Por qué Python ?

## Consola interactiva para pruebas

```
>>> a=EscuchaUSB()  
>>> pprint.pprint(a)
```

## Multiplataforma

```
root@pi:~# scp -r portatil:/home/jose/proyecto .  
root@pi:~# python controlppal.py
```

## Algunas 'perlas'

```
# Comprensión de listas/dicts  
>>> [{ 'S/N':d['datos']['SerialNumber'],  
      'dev':d['device name']} for d in lista_usb  
      if d['accion']=='CONECTADO']  
  
# Mapeo de dos listas en un diccionario  
>>> dict(zip(lista_nombres_campos,lista_valores))  
  
# Ejecución de strings de código fuente  
>>> exec('; '.join(['m.c_%02d=0' % i for i in range(1,100)]))
```

# ¿ Por qué Python ?

Módulos para casi todo lo que podamos necesitar

```
>>> import serial          # Acceso al puerto serie
>>> import smbus          # GPIO a traves de I2C
>>> import bitstring      # Manipulación de datos binarios
>>> import sqlite3        # Grabar los datos en la BD SQLite
>>> import pysqlcipher    # BD SQLite encriptados
>>> import pickle         # Serialización de objetos python
>>> import zlib           # Compresión de datos
>>> import Crypto.Cipher  # Algoritmos de cifrado
>>> import datetime       # Manejo fecha/hora
>>> import hashlib        # MD5, SHA-256 ...
>>> import subprocess     # Llamadas a comandos del S.O.
...

```

# Operaciones a muy bajo nivel

Acceder a un pendrive en RAW leyendo la información almacenada a partir del primer Megabyte.

```
>>> f=open('/dev/sdb','rb')
>>> f.seek(1024*1024)
>>> cuatro_bytes=f.read(4)
```

Conectarse a una máquina/dispositivo mediante puerto serie

```
>>> m = serial.Serial('/dev/ttyUSB1', timeout=1)
>>> nbytes_enviados = m.write(str_cmd)
>>> respuesta = m.read(nbytes_a_leer)

# Si no sabemos cuantos bytes debemos leer:
>>> respuesta = ''
>>> while dato<>'':
    dato = m.read(1)
    respuesta+=dato
```

# Operaciones a muy bajo nivel

Leer el estado de un sensor conectado por GPIO

*Ejemplo: botonera de Adafruit conectada a una Raspberry*

```
>>> resultado = smbus.SMBus(0).read_byte_data(0x20, 0x09) & 0b11111
```

Devuelve un byte cuyos 5 primeros bits indican que botones se están pulsando ahora mismo:

B2	B3	B4	B5	B1	: Int	: Botón pulsado
0	0	0	0	1	1	B1
1	0	0	0	0	16	B2
0	1	0	0	0	8	B3
0	0	1	0	0	4	B4
0	0	0	1	0	2	B5
...						
1	1	1	1	1	31	B1+B2+B3+B4+B5 simultaneamente
...						
1	0	0	0	1	17	B1+B2 simultaneamente
...						
0	0	0	0	0	0	No se está pulsando ningún botón

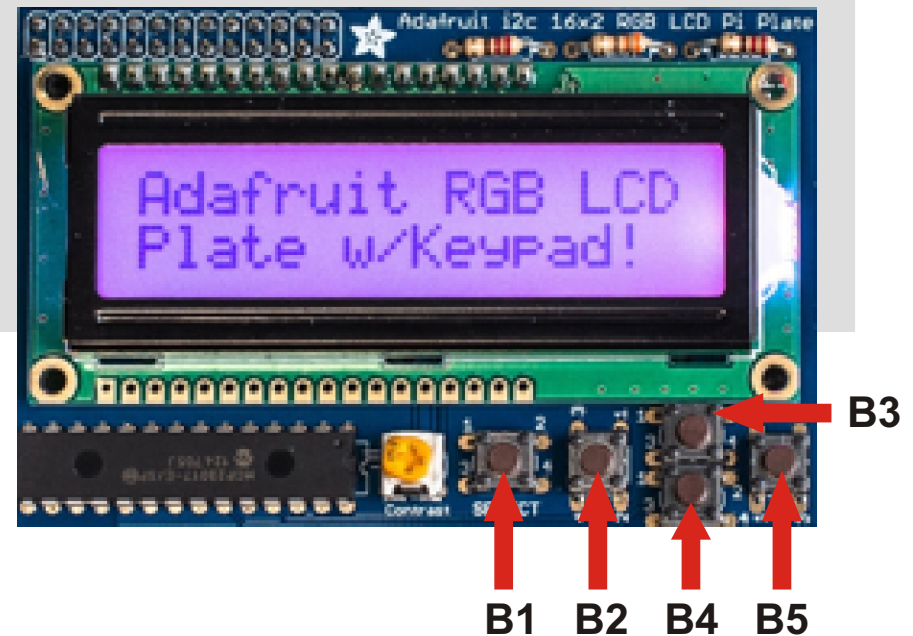
# Procesar un chorro de bytes, 'humanizando' su contenido.

Supongamos que se ha pulsado el Botón 3 (resultado==4)

```
>>> resultado = smbus.SMBus(0).read_byte_data(0x20, 0x09) & 0b111111
>>> tmp = '0000'+bin(resultado).replace('0b', '')
```

```
>>> nombres_botones = ['Pulsado boton %s' % i for i in range(1,6)]
>>> cinco_bits = [bool(int(b)) for b in tmp[-5:]]
>>> hum_result = dict(zip(nombres_botones, cinco_bits))
```

```
>>> pprint.pprint(hum_result)
{'Pulsado boton 1': False,
 'Pulsado boton 2': False,
 'Pulsado boton 3': True,
 'Pulsado boton 4': False,
 'Pulsado boton 5': False}
```



# Interfaz con el sistema operativo.

## *Interacción y control de sistemas Linux-based*

```
>>> import subprocess

>>> def tamaño_disco(device):
    """
    Devuelve el tamaño en bytes del dispositivo.
    Si no existe devuelve None.
    """
    fdisk_cmd='fdisk -l %s' % device
    try:
        p=subprocess.Popen(fdisk_cmd, stdout=subprocess.PIPE,
                            shell=True)
        (output, err)=p.communicate()
        tam_en_bytes=int(output.split('\n')[1].split( \
            ',') [1].replace('bytes','').strip())
    except:
        print "ERROR en %s" % fdisk_cmd
        tam_en_bytes=None
    return tam_en_bytes

>>> tamaño_disco('/dev/sda')
160041885696
```

# Detectar la conexión de un PenDrive utilizando /var/log/syslog

*/var/log/syslog* contiene estos datos:

```
... kernel: [10725.370295] usb 1-1.3.4: new high-speed USB device number 9 using dwc_otg
... kernel: [10725.471936] usb 1-1.3.4: New USB device found, idVendor=14cd, idProduct=121c
... kernel: [10725.471973] usb 1-1.3.4: New USB device strings: Mfr=1, Product=3, SerialNumber=2
... kernel: [10725.471993] usb 1-1.3.4: Product: Mass Storage Device
... kernel: [10725.472010] usb 1-1.3.4: Manufacturer: Generic
... kernel: [10725.472031] usb 1-1.3.4: SerialNumber: 812320090519
... kernel: [10725.480233] usb-storage 1-1.3.4:1.0: USB Mass Storage device detected
... mtp-probe: checking bus 1, device 9: "/sys/devices/platform/bcm2708_usb/usb1/1-1/1-1.3/1-1.3.4"
... kernel: [10725.490203] scsi1 : usb-storage 1-1.3.4:1.0
... mtp-probe: bus: 1, device: 9 was not an MTP device
... kernel: [10726.491087] scsi 1:0:0:0: Direct-Access USB Mass Storage Device PQ: 0 ANSI: 0 CCS
... kernel: [10726.493747] sd 1:0:0:0: [sda] 3911680 512-byte logical blocks: (2.00 GB/1.86 GiB)
... kernel: [10726.494342] sd 1:0:0:0: [sda] Write Protect is off
```

*Nosotros queremos estos:*

```
>>> pprint.pprint(EscuchaUSB.DevuelveSucesosUSB())
[{'accion': 'CONECTADO',
  'datos': {'Manufacturer': 'Generic',
            'Product': 'Mass Storage Device',
            'SerialNumber': '812320090519',
            'idProduct': '121c',
            'idVendor': '14cd',
            'id_completo': True,
            'size_bytes': 2002780160},
  'device name': '/dev/sda',
  'device number': '9',
  'hub_device_function': '1-1.3.4'}]
```



# Detectar la conexión de un PenDrive utilizando /var/log/syslog

```
>>> EscuchaUSB=DetectaUSB()
>>> l_sucesos_usb=EscuchaUSB.DevuelveSucesosUSB()

# Leemos la lista al revés, las últimas inserciones estarán primero
>>> for suceso in l_sucesos_usb[::-1]:
    if suceso['accion']=='CONECTADO':
        # Acciones cuando se conecta un nuevo USB
    elif suceso['accion']=='DESCONECTADO':
        # Acciones al desconectarlo
```

# Detectar la conexión de un PenDrive utilizando /var/log/syslog

```
class DetectaUSB:
    """
    Comprueba cada cierto tiempo /var/log/syslog
    para detectar si se ha conectado/desconectado
    algún dispositivo USB.
    """

    def __init__(self, fichero_log='/var/log/syslog'):
        self.d_usbs_conectados={}
        self.l_sucesos_usb=[]
        self.fichero_log=fichero_log
        f=open(fichero_log, 'r')
        f.seek(0,2)
        self.ultimo_seek=f.tell()
        f.close()

    def __lee_lineas_log(self):
        f=open(self.fichero_log, 'r')
        f.seek(self.ultimo_seek)
        l_log_lines=f.readlines()
        self.ultimo_seek=f.tell()
        f.close()
        return l_log_lines
```

# Detectar la conexión de un PenDrive utilizando /var/log/syslog

```
(continuación class DetectaUSB:)

def devuelve_datos_usb_device(self,line):
    d={}
    try:
        if line.find(' device number ')<>(-1):
            p=line.find('device number')
            if line.find('disconnect')<>(-1):
                d.update(dict([('DESCONECTADO',
                    line[p:p+16].replace('device number ','').strip())]))
            else:
                d.update(dict([('CONECTADO',
                    line[p:p+16].replace('device number ','').strip())]))
                d.update(dict([('hub_device_function',
                    line.split('] usb ')[1].split(':')[0].strip())]))
        elif (line.find('idVendor')<>(-1)) and (line.find('idProduct')<>(-1)):
            d.update(dict([tuple(line.split(':')[1][-14:].split('='))]))
            d.update(dict([tuple(line.split(':')[1][-29:-16].split('='))]))
        elif line.find(': Product: ')<>(-1):
            d.update(dict([('Product',line.split(':')[1].strip())]))
        elif line.find(': Manufacturer: ')<>(-1):
            d.update(dict([('Manufacturer',line.split(':')[1].strip())]))
        elif line.find(': SerialNumber: ')<>(-1):
            d.update(dict([('SerialNumber',line.split(':')[1].strip())]))
    except:
        d={'ERROR','parseando linea:'+line+''}
    return d
```

# Creación de un daemon que inicia automáticamente un display

Paso 1: Crear `/root/python-ta/panel_lcd.py`

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
#
import adafruit.myLCD
lcd=adafruit.myLCD.MyLCDPlate()
lcd.message('Iniciando..\nSistema')

while True:
    if lcd.buttonPressed(lcd.B1):
        lcd.backlight(lcd.RED)
        lcd.message('B1 Pulsado')
        # Acciones del botón B1
    if lcd.buttonPressed(lcd.B2):
        lcd.backlight(lcd.GREEN)
        lcd.message('B2 Pulsado')
        # Acciones del botón B2
    if lcd.buttonPressed(lcd.B1) and lcd.buttonPressed(lcd.B2):
        lcd.backlight(lcd.ON)
        lcd.message('Pulsados\n B1 y B2')
        # Acciones del botón B1+B2
```

# Creación de un daemon que inicia automáticamente un display

## Paso 2: Crear /etc/init.d/panel\_lcd

```
#!/bin/sh
### BEGIN INIT INFO
# Provides: panel_lcd
# Required-Start: $remote_fs $syslog
# Required-Stop: $remote_fs $syslog
# Default-Start: 2 3 4 5
# Default-Stop: 0 1 6
# Short-Description: Botonera y display LCD
# Description: Botonera y display LCD para controlar
#               funciones basicas del appliance
### END INIT INFO
#
export HOME
case "$1" in
  start)
    echo "Starting LCD"
    /root/python-ta/panel_lcd.py 2>&1 &
    ;;
  stop)
    echo "Stopping LCD"
    LCD_PID=`ps auxwww | grep panel_lcd.py | head -1 | awk '{print $2}'`
    kill -9 $LCD_PID
    ;;
  *)
    echo "Usage: /etc/init.d/panel_lcd {start|stop}"
    exit 1
    ;;
esac
exit 0
```

# Creación de un daemon que inicia automáticamente un display

*Paso 3: Dar permisos de ejecución:*

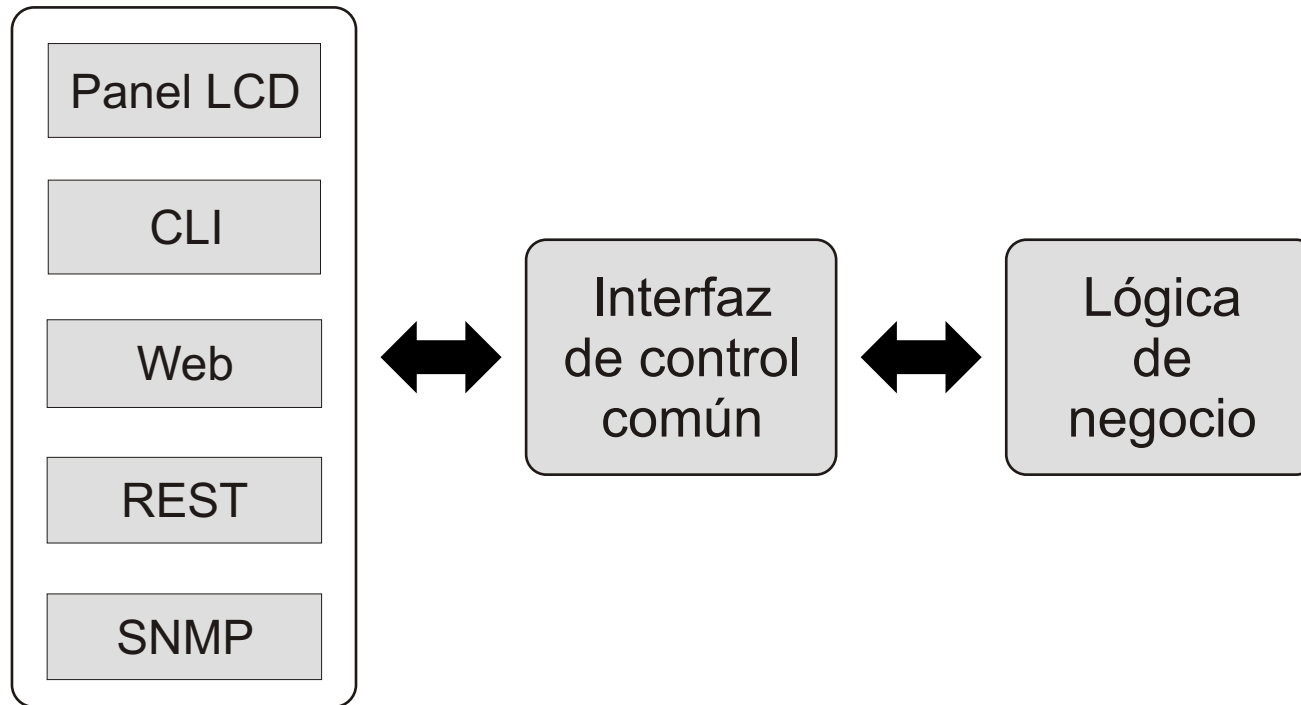
```
root@pi:~# chmod +x /root/python-ta/panel_lcd.py
root@pi:~# chmod +x /etc/init.d/panel_lcd
```

*Paso 4: Instalar el script en el sistema de inicio*

```
root@pi:~# update-rc.d panel_lcd defaults
```

*Reiniciar y listo ...*

# Construcción de interfaces



*Interfaz web / RESTful: django, Flask, TurboGears ...*

*SNMP: pysmnp*

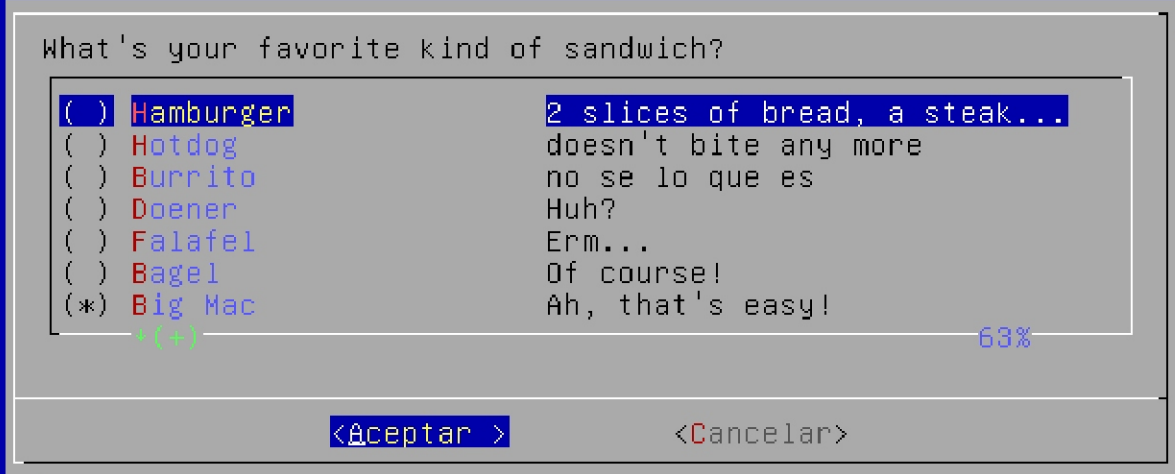
***PySNMP is a cross-platform, pure-Python SNMP engine implementation. It features fully-functional SNMP engine capable to act in Agent/Manager/Proxy roles, talking SNMP v1/v2c/v3 protocol versions over IPv4/IPv6 and other network transports.***

# Construcción de interfaces

## Texto plano: interfaz CLI utilizando python-dialog

*pythondialog is a Python wrapper for the dialog utility originally written by Savio Lam, and later rewritten by Thomas E. Dickey. Its purpose is to provide an easy to use, pythonic and comprehensive Python interface to dialog. This allows one to make simple text-mode user interfaces on Unix-like systems (including Linux).*

```
pythondialog demo
```



```
What's your favorite kind of sandwich?
```

<input checked="" type="checkbox"/> Hamburger	2 slices of bread, a steak...
<input type="checkbox"/> Hotdog	doesn't bite any more
<input type="checkbox"/> Burrito	no se lo que es
<input type="checkbox"/> Doener	Huh?
<input type="checkbox"/> Falafel	Erm...
<input type="checkbox"/> Bagel	Of course!
<input checked="" type="checkbox"/> Big Mac	Ah, that's easy!

63%

<Aceptar >      <Cancelar >



# Construcción de interfaces

## *Ventajas interfaz CLI:*

*Manipulación por técnicos y operadores:*

- Conexiones al appliance mediante puerto serie, conexión telefónica, teclado-pantalla, ...*
- Configuración y realización de tareas básicas para las que fue diseñado (envío de datos, manipulación de un dispositivo ...)*

*Estableciéndolo como shell por defecto*

```
root@pi:~# vi /etc/passwd  
  
(...)  
operador:x:1001:1001::/home/operador:/opt/python-ta/interfaz.py  
(...)
```

# Almacenamiento y procesamiento de datos

## **Ficheros binarios:**

*Python pickle (compatible entre arquitecturas)*

```
>>> import pickle

# Volcado a disco:
>>> f_out=open('/home/jose/objeto.pkl','wb')
>>> pickle.dump(objeto,f_out,2)
>>> f_out.close()

# Lectura desde disco
>>> f_in=open('/home/jose/objeto.pkl','rb')
>>> objeto=pickle.load(f_in)

# Volcado a un string:
>>> str = pickle.dumps(objeto,2)

# Lectura
>>> objeto = pickle.loads(str)
```

# Almacenamiento y procesamiento de datos

## *Aplicación de pickle:*

Envío de una lista de datos a través de una petición HTTP,  
o bien guardar en un campo varchar() de la BD:

serializa ↔ comprime ↔ [encripta] ↔ base64 ↔ (urlquote)

## *Binarios HDF5 y PyTables*

The h5py package is a Pythonic interface to the HDF5 binary data format. It lets you store huge amounts of numerical data, and easily manipulate that data from NumPy. For example, you can slice into multi-terabyte datasets stored on disk, as if they were real NumPy arrays. Thousands of datasets can be stored in a single file, categorized and tagged however you want.

```
>>> import h5py
```

PyTables is a package for managing hierarchical datasets designed to efficiently cope with extremely large amounts of data. It is built on top of the HDF5 library, the Python language and the NumPy package.

```
>>> import tables
```

# Almacenamiento y procesado de datos

## *Binarios XLS*

```
# Reading data and formatting information from Excel files.
>>> import xlrd

# Writing data and formatting information to Excel files.
>>> import xlwt
```

## *Ficheros CSV*

```
# Supongamos que la primera linea contiene nombres de campos
>>> datos=open(fich_csv).readlines()
>>> nombres_campos=datos[0].replace('"','').strip().split(';')
>>> l_art=[]
>>> for linea in datos[1:]:
    v=dict(zip(nombres_campos,
              linea.replace('"','').strip().split(';')))
    l_art.append(v)

# Accedemos al dato por el nombre del campo
>>> l_art[0]['nombre']

# Utilizando el módulo csv
>>> import csv
```

# Almacenamiento y procesado de datos

## **Bases de datos.**

*Todos los datos en un sólo fichero: SQLite y Firebird*

```
>>> import sqlite3  
  
>>> import kinterbasdb
```

*Servidor de BD completo: MySQL y PostgreSQL*

```
>>> import MySQLdb  
  
# PostgreSQL Python DB API 2.0 libpq wrapper  
>>> import psycopg2  
  
# PyGreSQL Python DB-API 2.0 compliant interface  
>>> import pgdb  
  
# Classic PyGreSQL interface  
>>> import pg
```

# Encriptación y transmisiones seguras

## *Función de encriptación basada en AES*

```
>>> import pickle
>>> import Crypto.Cipher.AES
>>> import zlib
>>> import base64
>>> def encripta(objeto, contraseña='Contraseña de 16',
                comprimido=True, b64=True):
    """
    NOTA: La contraseña AES debe ser de 16, 24,
          o 32 bytes de tamaño
    """
    cadena=pickle.dumps(objeto,2)
    aes_obj = Crypto.Cipher.AES.new(contraseña,
                                    Crypto.Cipher.AES.MODE_ECB)
    padding = ''
    if (len(cadena) % 16 ) <> 0 :
        padding = 'X' * (16-(len(cadena) % 16))
    cadena=aes_obj.encrypt(cadena+padding)
    cadena=zlib.compress(cadena) if comprimido else cadena
    cadena=base64.b64encode(cadena) if b64 else cadena
    return cadena

>>> encripta('jose')
'eJyrPmd7kPEca463+vOoYvaGewA8LQcT'
```

# Encriptación y transmisiones seguras

*Encriptando los datos en la BD:*

```
>>> cur.exec("""
CREATE TABLE datos (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    contenido text,
    creado TIMESTAMP NOT NULL DEFAULT current_timestamp,
    sincronizado TIMESTAMP DEFAULT NULL) """)

>>> cur.exec("""
INSERT INTO datos (contenido) VALUES '%s'
                "" " % encripta(objeto))
```

*Problema:*

*No se pueden utilizar funciones SQL sobre los datos de manera nativa.  
MAX, MIN, COUNT(\*), ORDER BY, GROUP BY, ...*

# Encriptación y transmisiones seguras

*BD con soporte para encriptación* (Python interface to SQLCipher):

```
>>> import pysqlcipher
```

pysqlcipher is an interface to the SQLite 3.x embedded relational database engine. It is almost fully compliant with the Python database API version 2.0. At the same time, it also exposes the unique features of SQLCipher.

*Ejemplo:*

```
root@pi:~# sqlcipher
sqlite> .open datosmaquinas.db
sqlite> PRAGMA key='password';
sqlite> .quit
```

```
>>> from pysqlcipher import dbapi2 as sqlcipher
>>> db = sqlcipher.connect('datosmaquinas.db')
>>> db.executescript('pragma key="password"; pragma kdf_iter=64000;')
>>> db.execute('select * from maquinas;').fetchall()
```



# Encriptación y transmisiones seguras

*Envío de datos a través de HTTPS a un servidor que requiera usuario y contraseña.*

```
>>> import ssl, urllib, urllib2
# Guarda cookies
>>> cookie_h = urllib2.HTTPCookieProcessor()
>>> opener = urllib2.build_opener(cookie_h)
>>> urllib2.install_opener(opener)
# Autentifica
>>> url = urllib2.urlopen('https://192.168.1.1/login',
                          urllib.urlencode({"username": "jose",
                                             "password": "1234"}))

# Envía datos
>>> url = urllib2.urlopen('https://192.168.1.1/envia_datos',
                          urllib.urlencode({"id": "5",
                                             "datos": encripta(datos_a_enviar)}))

# Recibe datos
>>> url = urllib2.urlopen('https://192.168.1.1/recibe_datos?id=5')
>>> datos_recibidos = f_url.read()
```

# Procesando datos con Pandas

pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language

```
# Creamos un Pandas Data Frame
>>> df=calcula_jugadas(desde='2014-06-20',
                       hasta='2014-07-20',
                       intervalo='1 dia',
                       pandas_output=True)

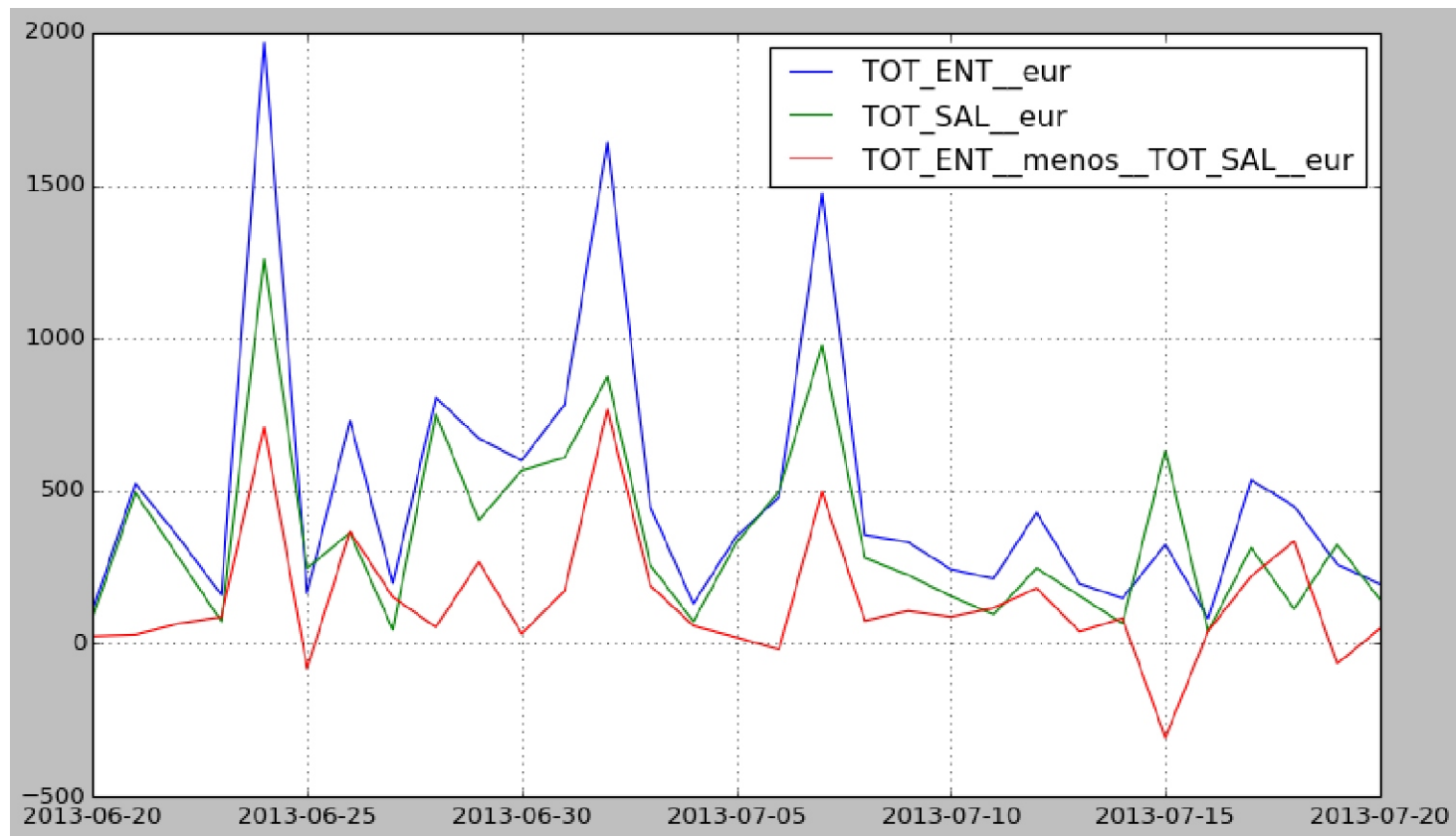
# Calculamos el dinero que tiene la máquina en cada instante
# a partir de un estado inicial
>>> recalcula_cash(df,{'1-euro':200,'2-euros':200})

# Cuanto dinero en monedas de 1 Euro hay?
>>> df.loc['2014-06-20','CASH_1-euro__eur']
Decimal('400.00')
>>> df.loc['2014-06-21','CASH_1-euro__eur']
Decimal('426.00')

# Cuanto dinero se 'generó' de un día para otro?
>>> df.loc['2014-06-21','CASH_TOTAL__eur'] - \
      df.loc['2014-06-20','CASH_TOTAL__eur']
Decimal('30.20')
```

# Procesando datos con Pandas

```
# Mostramos un gráfico  
>>> df[['TOT_ENT_eur', 'TOT_SAL_eur',  
        'TOT_ENT_menos__TOT_SAL_eur']].astype(float).plot()
```



# Protegiendo la ejecución del código

*Verificando hash de los módulos antes de cargarlos.*

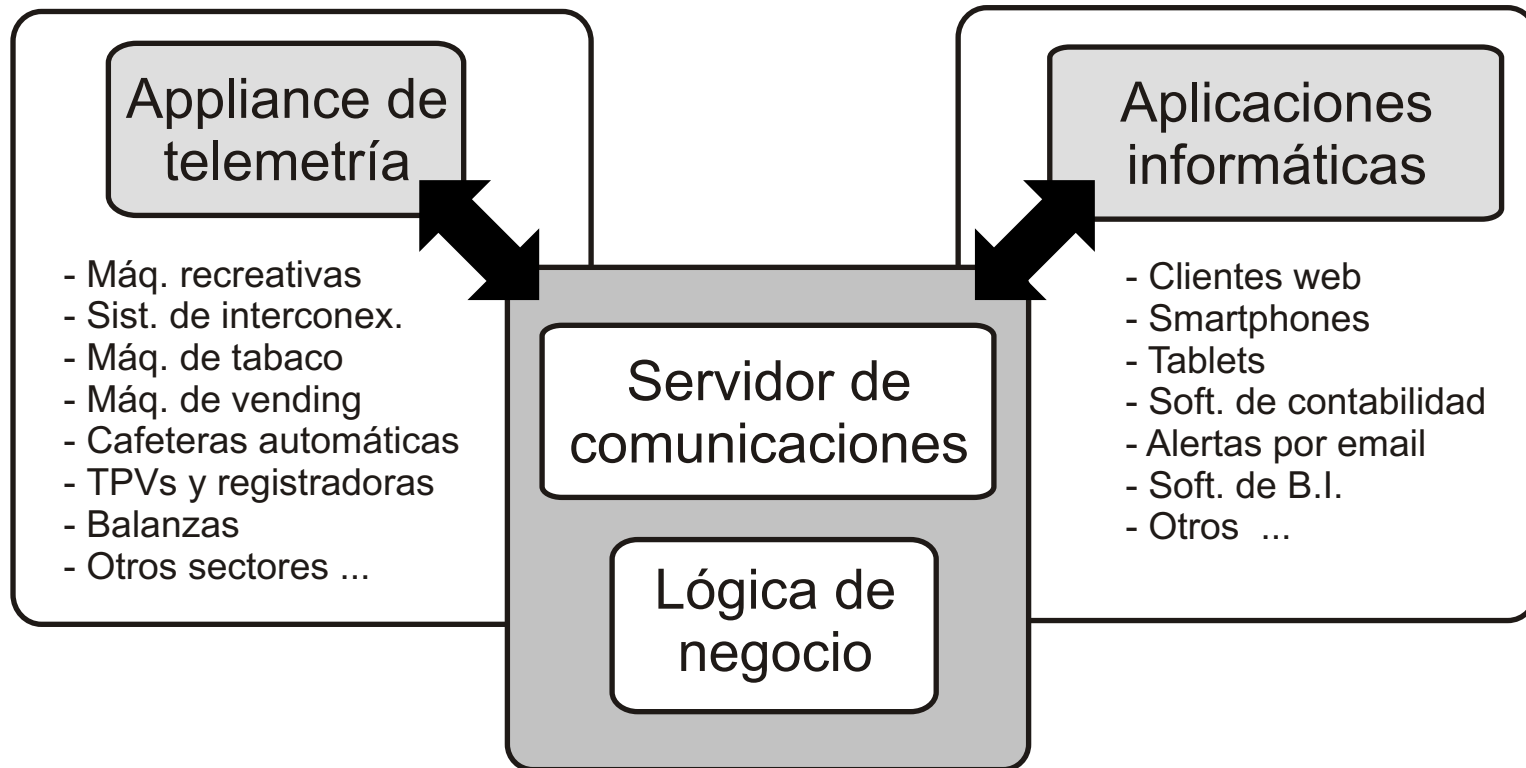
```
>>> import hashlib
>>> hashlib.algorithms
('md5', 'sha1', 'sha224', 'sha256', 'sha384', 'sha512')

>>> if hashfile(open('datasaver.py', 'r')) ==
    '674c4287a2782705a33c22df08511784':
    import datasaver
else:
    raise Exception('Hash no válido')

>>> import compileall
>>> compileall.compile_dir('/opt/python-ta/', force=True)

>>> import os
>>> pyc_hashes = {f:hashfile(open(os.path.join(basedir, f), 'rb'))
    for basedir, dirs, fichs in os.walk('/opt/python-ta/')
    for f in fichs if f.endswith('.pyc')}
```

# Esquema general de un sistema de telemetría



# Sistema de ayuda en la toma de decisiones

*Ejemplo: appliance de telemetría para una empresa operadora de máquinas automáticas (recreativas, vending, ...)*

## **¿Qué tipo de máquina debo ubicar en este establecimiento?**

El sistema me permite comparar con establecimientos similares, y realizar una estimación de ingresos.

## **¿Debería incorporar más máquinas a mi explotación?**

Mediante el sistema de telemetría puedo testear máquinas y ubicaciones, obteniendo datos reales que posteriormente analizo.

## **¿Cuanta mercancía cargo en la furgoneta para atender mi ruta?**

Consulto el sistema de telemetría para obtener exactamente las unidades que necesito recargar en cada máquina.

## **¿Cuanto cambio (monedas) será el adecuado para la ruta de hoy?**

El sistema de telemetría me informa de la cantidad exacta de monedas que tiene cada máquina

---

# ¿Preguntas?

---

José A. Rocamonde  
jrocamonde@gmail.com  
@jrocamonde

