Python

Pequeño manual de Python

1. Introducción a Python
2. Estructura léxica de Python
3. Tipos de datos
4. Control de flujo
5. Entrada/Salida y Ficheros
6. Funciones definidas por el usuario
7. Clases y programación orientada a objetos
8. Módulos
9. Metodología, Herramientas y Entornos de desarrollo
10. Administración de sistemas con Python
11. Expresiones regulares con Python
12. Acceso a Bases de Datos MySQL con Python
13. Python - Raspberry Pi

Sumario

- 1 Programación Orientada a Objetos en Python
- 2 Aplicaciones de Escritorio con TKinter
- 3 Intercambio de datos con jSON
- 4 Ejemplos Python + Raspberry Pi 2
    - 4.1 LED y PIR
    - 4.2 Encendido y Apagado de LED por Web
    - 4.3 Sensor de Temperatura y Humedad AM2302
- 5 Arrays
- 6 Enlaces

Programación Orientada a Objetos en Python

Python es un lenguaje de programación orientado a objetos. Pero, a diferencia de otros lenguajes de este tipo, Python no obliga a utilizar esta "orientación a objetos".

---------- Seguir en página 81 del libro Python in a nutshell ----------

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

class Persona:
    nBrazos=0
```

```
    nPiernas=0
    cabello=True
    cCabello="Defecto"
    hambre=0 #hambre de 0-10

    def __init__(self):
        self.nBrazos=2
        self.nPiernas=2

    def dormir(self):
        pass
    def comer(self):
        self.hambre=5

class Hombre(Persona):
    nombre="Defecto"
    sexo="M"

    def cambiarNombre(self,nombre):
        self.nombre=nombre

class Mujer(Persona):
    nombre="Defecto"
    sexo="F"



jose=Hombre()  # Definimos el objeto jose=Hombre()
jose.comer()    # LLamamos al método comer
print jose.hambre   # Comprobamos si el parámetro hambre se cambió o no
```

Aplicaciones de Escritorio con TKinter

Seguir la web: effbot.org

Intercambio de datos con jSON

A continuación vemos un ejemplo en el que adquirimos información con jSON, también utilizamos el módulo requests que nos provee de múltiples métodos para obtener todo tipo de información de una url, además de utilidades para normalizar nuestros datos obtenidos.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import requests
import json

addr_str = raw_input("Dime tu dirección: ")

maps_url = "https://maps.googleapis.com/maps/api/geocode/json"
tiene_sensor = "false"

payload = {'address': addr_str, 'sensor': tiene_sensor}

r = requests.get(maps_url,params=payload)

maps_output = r.json()

lista_resultado = maps_output['results']
estado_resultado = maps_output['status']

direccion = lista_resultado[0]['formatted_address']
result_geo_lat = lista_resultado[0]['geometry']['location']['lat']
result_geo_lng = lista_resultado[0]['geometry']['location']['lng']

print("%s is at\nlat: %f\nlng: %f" % (direccion, result_geo_lat, result_geo_lng))
```

Ejemplos Python + Raspberry Pi 2

LED y PIR

.- Programa con detector PIR en GPIO PIN 11 y LED en GPIO PIN 3. Si detecta presencia el LED se enciende y si no detecta presencia el LED permanece apagado.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
# Blinking
import RPi.GPIO as GPIO ## Import GPIO library
import time ## Import 'time' library. Allows us to use 'sleep'

# El LED estará en el pin GPIO 3
GPIO_LED = 3
# El PIR estará en el pin GPIO 11
GPIO_PIR = 11

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(GPIO_LED, GPIO.OUT)
GPIO.setup(GPIO_PIR, GPIO.IN)

while True:
    i = GPIO.input(GPIO_PIR)
    if i == 0:          #Cuando la salida del PIR es LOW
        print 'No hay intrusos'
        GPIO.output(GPIO_LED,0) # LED apagado
        time.sleep(3) # Espera 3 segundos
    elif i == 1:
        print 'Intruso cerca'
        GPIO.output(GPIO_LED,1) # LED encendido
        time.sleep(3) # Espera de 3 segundos
```

Encendido y Apagado de LED por Web

.- Encender y apagar LED por interface web.

- Instalamos servidor web Apache en la Raspberry Pi con soporte para PHP.

```
# apt-get update
# apt-get upgrade
# apt-get dist-upgrade
# reboot
# apt-get install apache2
# apt-get install php5 libapache2-mod-php5
```

El directorio donde se guardan las páginas web por defecto es */var/www*.

- Ahora añadimos el usuario **www-data** como *sudoer* para que no tenga problemas para interactuar con GPIO:

Escribimos **visudo** en el terminal y añadimos una nueva linea en el archivo */etc/sudoers*:

```
# See sudoers(5) for more information on "#include" directives:

#includedir /etc/sudoers.d
pi ALL=(ALL) NOPASSWD: ALL
www-data ALL=(ALL) NOPASSWD: ALL
```

- Contenido del archivo **enciende.py**

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
# enciende.py
import RPi.GPIO as GPIO ## Import GPIO library

# El LED estará en el pin GPIO 3
GPIO_LED = 3
```

```
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(GPIO_LED, GPIO.OUT)

# Encendemos el LED
GPIO.output(GPIO_LED, GPIO.HIGH)
```

- Contenido del archivo **apaga.py**

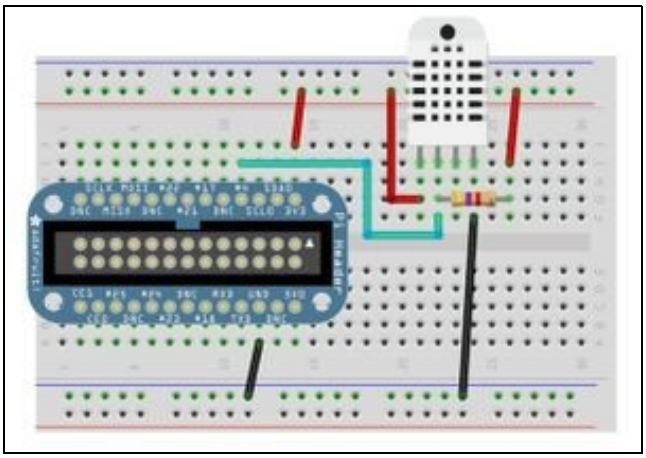
```
#!/usr/bin/python
# -*- coding: utf-8 -*-
# apaga.py
import RPi.GPIO as GPIO ## Import GPIO library

# El LED estará en el pin GPIO 3
GPIO_LED = 3

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(GPIO_LED, GPIO.OUT)

# Apagamos el LED
GPIO.output(GPIO_LED,0)

# Liberamos los GPIO
# GPIO.cleanup()
```

- Por último, programamos el archivo **index.php** que nos va a permitir encender y apagar el LED:

```
<html>
 <head>

 </head>
 <body>


  <form action="" method="post">
   GPIO 03 <input type="submit" name="encender03" value="Encender">
   <input type="submit" name="apagar03" value="Apagar">

 <br></br>
 </body>
</html>

<?php

// Funciones PHP del pin GPIO 03

  if ($_POST[encender03]) {
   $a- exec("sudo python /var/www/enciende.py");
   echo $a;
  }

  if ($_POST[apagar03]) {
   $a- exec("sudo python /var/www/apaga.py");
   echo $a;
  }

// Fin de las funciones del pin GPIO 03
?>
```

Sensor de Temperatura y Humedad AM2302

El sensor de Temperatura y Humedad AM2302 es una versión del sensor DHT22 y trabaja con las librerías de Adafruit para los sensores DHT.



AM2302

- Tiene tres pines de conexión: uno rojo (VDD, *power*), uno amarillo (*Data signal*) y uno negro (GND).

  Realizaremos las siguientes conexiones:

  - ◊ Conectar el cable rojo al GPIO pin 1 (3,3V).
  - ◊ Conectar el cable negro al GPIO pin 6 (Tierra).
  - ◊ Conectar el cable amarillo al GPIO pin 7 (GPIO04).

  En la imagen de la derecha podemos verlo (¡ojo que NO es la Raspberry pi 2!).

- Instalamos la librería Python necesaria:

  - ◊ Descargamos el código de Github:

```
# cd /tmp
# git clone https://github.com/adafruit/Adafruit_Python_DHT.git
# cd Adafruit_Python_DHT
```

  - ◊ Instalamos las dependencias necesarias:

```
# sudo apt-get update
# sudo apt-get install build-essential python-dev python-openssl
```

  - ◊ Instalamos la librería:

```
# sudo python setup.py install
```

  - ◊ Hacemos un test de que todo va bien con alguno de los programas que existen en la carpeta ***examples***. Podemos utilizar el llamado *AdafruitDHT.py* al que le tenemos que pasar dos parámetros, el primero para indicar cual es el sensor utilizado (11, 22, or 2302) y el segundo el pin GPIO al que está conectada la salida del sensor. Así:

```
# cd examples
# sudo ./AdafruitDHT.py 2302 4
```

  Siendo la salida algo como esto:

```
root@pi1:/home/pi/scripts/Adafruit_Python_DHT/examples# ./AdafruitDHT.py 2302 4
Temp=24.1*C  Humidity=50.6%
root@pi1:/home/pi/scripts/Adafruit_Python_DHT/examples#
```

Realizaremos un sencillo programa que calcule la Temperatura y Humedad actual y encienda un Led cuando la temperatura llegue a una pasada como parámetro.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

########### Definimos las funciones de Encender y Apagar el LED ################
##
### enciende(pin) ###
def enciende(pinLED):
    import RPi.GPIO as GPIO ## Import GPIO library

    # El LED estará en el pin pasado como parámetro
    GPIO_LED = pinLED

    GPIO.setwarnings(False)
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(GPIO_LED, GPIO.OUT)

    # Encendemos el LED
    GPIO.output(GPIO_LED, GPIO.HIGH)
########################################

## apaga(pin) ###
def apaga(pinLED):
    import RPi.GPIO as GPIO ## Import GPIO library

    # El LED estará en el pin GPIO 3
    GPIO_LED = pinLED

    GPIO.setwarnings(False)
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(GPIO_LED, GPIO.OUT)

    # Apagamos el LED
    GPIO.output(GPIO_LED,0)

    # Liberamos los GPIO
    # GPIO.cleanup()

########################################

import sys
import Adafruit_DHT

# Parámetros
if len(sys.argv) == 2:
    TempLED = float(sys.argv[1])
    sensor = Adafruit_DHT.AM2302
    pin = 4
else:
    print 'Debes introducir un parámetro '
    print 'que será la Temperatura desde la que se encienda el LED'
    sys.exit(1)

# Esta función lee la temperatura y la humedad cada 2 segundos
humedad, temperatura = Adafruit_DHT.read_retry(sensor, pin)

# Mostramos los datos leídos siempre si estos se producen
# Si la Temperatura supera la dada como parámetro encendemos el LED
if humedad is not None and temperatura is not None:
    print 'Temp={0:0.1f}*C  Humedad={1:0.1f}%'.format(temperatura, humedad)
    ## El LED está conectado en el pin 3 de la Raspberry Pi
    if float(temperatura) >= TempLED:
        enciende(3)
    else:
        apaga(3)
else:
    print 'Fallo leyendo! Inténtalo de nuevo!'
```

Leer Temperatura y Humedad del sensor AM2302.

Arrays

Un array es un conjunto de valores con el mismo tipo, esto es, todos sus elementos son enteros, reales en doble precisión, complejos en simple precisión...
Python ya dispone de un tipo array que sirve para almacenar elementos de igual tipo pero no proporciona toda la artillería matemática necesaria como para hacer operaciones de manera rápida y eficiente. De este modo, siempre que nos refiramos a la clase array siempre nos referiremos a la que viene con el módulo **numpy**.

Si consultamos la documentación de numpy nos cuenta que proporciona lo siguiente:

- ◊ Un objeto tipo array para datos homogéneos de tipo arbitrario
- ◊ Operaciones matemáticas rápidas para dichos arrays
- ◊ Rutinas para álgebra lineal, transformadas de Fourier y generación de números pseudoaleatorios.

Es, en sentido estricto, una parte mínima que permite convertir Python en un lenguaje apto para Cálculo Numérico.
Instalación de **numpy** en Debian:

```
# apt-get install python-numpy
```

Enlaces

- Curso Python Google
- Libros de Python
- Librerías Python