PROGRAMACIÓN ANDROID CON PYTHON

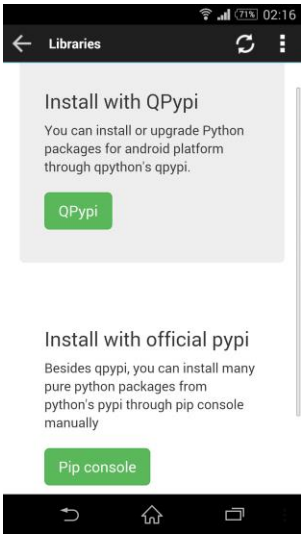
Una opción muy viable para la programación Android desde el mismo terminal es QPython, una herramienta muy poderosa, la oportunidad de **programar en Python desde nuestro teléfono móvil Android**, y más importante que eso, poder **acceder a los sensores del propio móvil** entre otras cosas.

QPython te permite **ejecutar código Python desde vuestras terminales Android**, además de **instalar librerías externas** con un solo comando, tanto librerías preparadas para el acceso a las funciones del móvil como instalar otras librerías disponibles para Python con cientos de funcionalidades directamente de la red. Estas aplicaciones las podemos dejar **ejecutándose en segundo plano** en nuestro móvil.

La aplicación de QPython la puedes descargar gratuitamente desde la **Play Store** y está basada en Python 2.x, pero también existe la versión para **Python 3.x**. Puedes encontrar documentación en http://qpython.com/ con ejemplos y explicaciones, o enstackoverflow.

Para la instalación de librerías externas consta de dos métodos:

- **QPypi**: librerías ya preparadas por la aplicación. Podemos encontrar *androidhelper* (acceso a sensores del móvil, cuadros de texto, etc), **kivy** (nos permite crear aplicaciones rápidas en nuestro android: botones, pantallas con mensajes, etc), **numpy**, **sqlite3**, etc.
- **pypi**: nos da acceso al directorio de librerías Python en la red, permitiendo la descarga de la librería que queramos tan solo introduciendo su nombre.

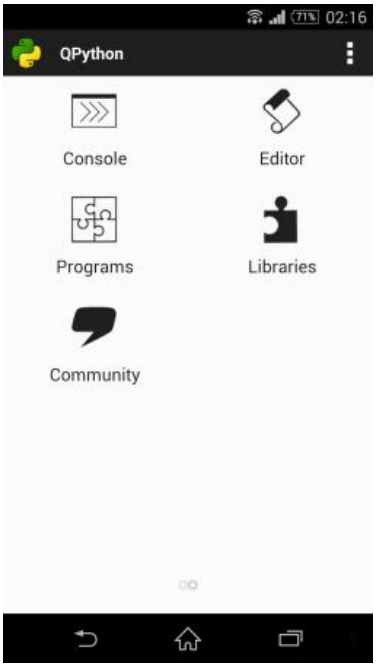


Noé Jiménez Peñaloza
Programación Paralela y Distribuida
Facultad de Ingeniería UAEMex

Una de la librerías que hemos probado a instalar es la librería de Geopy, entrando en el apartado de librerías de la aplicación y escribiendo un: **`pip install geopy`** se nos instalará esta librería de Python para la geocodificación de coordenadas en direcciones. La precisión de los GPS en los móviles sabemos que no es una maravilla, pero para muchos casos puede ser suficiente, y lo mismo de sus sensores de orientación, etc.

Una vez instalada, podemos abrir la aplicación y acceder a un pequeño tutorial **"Quick Start".** Aquí ya podemos echar un vistazo de la apariencia que tiene el código en estas aplicaciones. En cualquiera de ellas podemos hacer click sobre **"Copy to editor to run"**, y una vez estamos en el editor hacemos click en el botón de **"Play"** para ejecutar el script. En el segundo podemos ver un ejemplo de una aplicación que cambia de color el fondo de pantalla al hacer click sobre la pantalla (tenerla en ojo para hacer captura al presionar de un punto GPS por ejemplo).

Si en vez de entrar en el tutorial, deslizamos la pantalla hacia la izquierda, aparecerán el resto de botones de la aplicación, entre los que podemos ver la consola para ejecutar código, el *Editor* y el *Programs* con nuestros scripts, el apartado de *Libraries* para la instalación de librerías externas y el de *Community* con acceso a la web de qpython.org



Otra cosa que estarás pensando es lo complicado que sería escribir código con el teclado del móvil, o el envío a través de otros medios. Desde QPython lo han solucionado con un**lector QR en su web que carga directamente tu código en la app** (actualmente no disponible por reforma de la web). Por ahora lo más factible sería cargar tu fichero en el móvil utilizando algún métodos de los comunes para envío de datos, ya sea usando la red o mediante cable de datos.

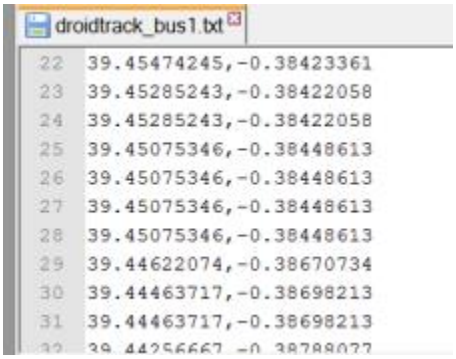
Un ejemplo sencillo es el **acceso al sensor gps**, mostrando por consola sus coordenadas. Aquí faltaría ver con más calma el significado de los datos que nos da, creo que si le pedimos al sensor las **coordenadas de tipo "network" son las relacionadas con tu red wifi o torre móvil** pero no están calculadas mediante gps, para ello son las de **tipo "gps"** que vemos en el script gps_track de más abajo.
(gps_basics.py)


```
1  import androidhelper
2  dr =androidhelper.Android()
3  dr.startLocating()
4  l =dr.readLocation()
5  rloc =l.result
6  rnet =rloc["network"]
7  print rnet["latitude"], rnet["longitude"]
```

Otro pequeño ejemplo es leer la orientación del sensor aunque no he podido comprobar su exactitud o funcionamiento, pero nos sirve como prueba:

```
1  import androidhelper
2  dr =androidhelper.Android()
3  dr.startSensingTimed(1, 250)
4  sensor =dr.sensorReadOrientation().result
5  l =dr.readSensors()
6  print l.result
7  lr =l.result
8  azi=lr['azimuth']
9  print azi
```

El siguiente script de prueba que he realizado está basado en ejemplos vistos en la wiki de Qpython y también en este antiguo post, **Droidtrack**, sobre la **captura de datos GPS**. Lo que conseguirás con el siguiente script es simplemente ir capturando los **datos del sensor gps de tipo "gps" cada 30 segundos** con un total de 50 puntos, siendo el resultado un fichero de texto como el de la imagen que aparecerá en nuestra tarjeta de memoria.
(gps_track.py)




```
1  #-*-coding:utf8;-*-
2  #qpy:2
3  #qpy:console
4
5  print "This is console module"
6
7  import androidhelper, time
8  droid =androidhelper.Android()
9  droid.startLocating()
10 time.sleep(10)
11 c =0
12 while c<50:
13
14   l =droid.readLocation()
15   R =l.result
```

```
16      try:
17        R =R["gps"]
18        Lat= R["latitude"]
19        Lon=R["longitude"]
20        #R1 =R["network"]
21        #Lat1=R1["latitude"]
22        #Lon1=R1["longitude"]
23        outstr = str(Lat) +","+ str(Lon)
24        droidfile = '/sdcard/droidtrack_bus1.txt'
25        print outstr#, str(Lat1) , str(Lon1)
26        fh = open(droidfile,'a')
27        res= fh.write(outstr +'\n')
28        res = fh.close()
29        time.sleep(30)
30        c+=1
31        print c
32      except:
33        time.sleep(10)
34        print c
35        print R
36        c+=1
```

Para este caso he querido realizar una representación rápida de los datos utilizando**CartoDB** quedando algo así:

Pensar la **cantidad de opciones que nos abre tener un dispositivo programable y móvil bajo un lenguaje como Python**, con el que poder realizar por ejemplo una interfaz con un formulario realizado con la librería *kivy* para la captura de coordenadas gps mediante *androidhelper*, teniendo la posibilidad de convertirlos al momento a direcciones mediante *geopy* y analizar las estadísticas

de la zona mediante *numpy*, todo esto mediante vuestro móvil en el propio lugar donde estés y de forma automatizada.