



Contents

- [1 Foreword](#)
- [2 Introduction](#)
- [3 Installing Python for S60](#)
 - ◆ [3.1 Installing the SDK / Emulator](#)
 - ◆ [3.2 Installing the phone](#)
 - ◆ [3.3 Testing programs PyS60](#)
- [4 The modules in PyS60](#)
 - ◆ [4.1 Events and callbacks](#)
 - ◆ [4.2 Functions](#)
 - ◆ [4.3 Specific modules of PyS60](#)
- [5 Developing](#)
 - ◆ [5.1 Connecting the camera](#)
- [6 See Also](#)

Foreword

This article is a translation of a Portuguese article - [Introdução ao Python para S60](#).

Introduction

Before getting started with [PyS60](#), we will familiarize with the actors involved in this tutorial.

Python

Python is a dynamic programming language that implements several paradigms of development. It is easy to learn and use, created in 1991 by Guido van Rossum. Its name is derived from the humorous British group Monty Python. More information about Python on [Python.org](#)

Series 60 (S60)

Series 60 (S60) is a platform for smartphones of Nokia phones that come equipped with the Symbian operating system. Currently there are 3 "editions" of the Series 60 and in each one a different version of Symbian is used. Within each edition there is still a subdivision of feature packs to distinguish between devices that have certain features (eg Wi-Fi, Bluetooth, camera, etc.).. A complete list of issues, FPs and devices can be found at http://www.forum.nokia.com/devices/matrix_all_1.html More information about the Series 60 in <http://www.s60.com>

Symbian

Symbian is proprietary operating system for mobile devices and charged by a consortium formed by companies like Nokia, Sony / Ericsson and Samsung. Currently Symbian belongs entirely to Nokia and there are rumors that the source code it will be available as Open Source. More information on <http://www.symbian.com>

Forum Nokia

Forum Nokia is portal for developers of mobile software for Nokia devices. Here you can find the SDK for developing C++ and Java for Series 60 and a series of articles, tutorials, examples, tools and resources for developers.

Python for S60

Python for S60 (PyS60) is a open Source project that is officially supported by Nokia. Python for S60 uses the 2.2 version of Python (which is a version considerably older since the former Python is going to version 2.6). The central point of information on the PyS60 is the site of the project in <http://pys60.sf.net>

Installing Python for S60

We will describe in this tutorial how to setup the working environment for development in Python using Windows but at the site of [PyS60](#) you can find information on how to develop PyS60 using both Linux and the MacOS X (which is considerably easier than using Windows) .

Although, there is an emulator for S60 mobile phones in the SDK C++ from Nokia, it does not implement all the features available in a real device (such as camera, communication via GSM or Bluetooth) so it is strongly recommended to have a real S60 device to accompany this tutorial. Any edits can be used but I recommend any "3rd Edition" device. The phone used to test code in this article is a Nokia N95 device that is a "3rd Edition Feature Pack 1", so if something does not work as described here on your device, make sure that this is not happening because of different version of device/SDK.

Installing the SDK / Emulator

If you already have a S60 device then installing the SDK is optional. The S60 C++ SDK depends on the installation of Perl on your computer. I recommend the installation of [ActiveState ActivePerl](#) because to install it simply run the installer).

Install the C++ SDK once you have downloaded it from [Forum Nokia](#). To do this simply decompress the file. From the zip file you just downloaded and run the installer (setup.exe), and follow the instructions. If you did not change the default location of installation, it is likely that you have a directory C:\Symbian on your computer.

Now that you've installed the C++ SDK and the emulator on your computer, it's time to download and install PyS60 to the emulator. For this is just down the PythonForS60_1_4_4_SDK_XXX.zip (replacing the XXX version of the SDK you have installed).

Introduction_to_PyS60

After you unpack the zip file, just copy its contents in C:\Symbian\XX\S60_3rd_FP1\Epoc32 (or the equivalent directory depending on the version of the SDK you have installed)

Now, we already have Python installed for running on the emulator. To run it simply enter the menu option

"Start" -> Programs-> S60 Developer Tools-> SDK-> Emulator

(be patient because the emulator will it take to get started and running) go on "Installed" and choose "Python".

Installing the phone

To install PyS60 on your cellphone you need to download the appropriate version of PyS60 for your cellphone in http://sourceforge.net/project/showfiles.php?group_id=154155 You'll need to download the two files with the extension SIS:

PythonForS60_1_4_4_VERSION.SIS : This package is the interpreter of Python itself.

PythonScriptShell_1_4_4_VERSION.SIS : This package provides the "Python" menu and some sample python scripts.

Now using the Nokia PC Suite, install these packages on your mobile phone so that the "Python" will appear on your menu of applications:



Application Menu

Python's start screen

Testing programs PyS60

Now is the last step that needs to be done before we are ready for the development of an application for S60: send a Python script to the device to test.

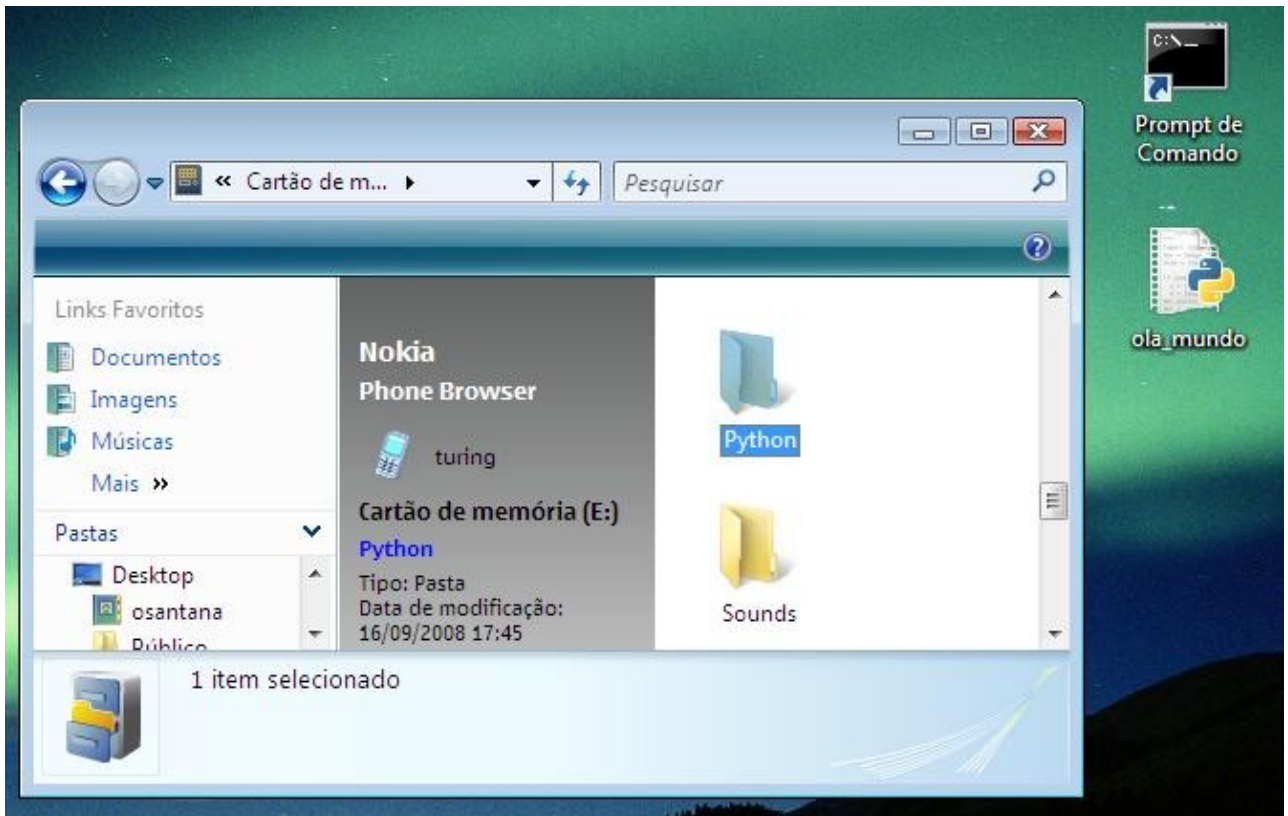
Introduction_to_PyS60

To do this we will do a small program called "ola_mundo.py" (translation note: "Olá mundo!" is the portuguese version of "Hello World!") with the following contents:

```
import appuifw
appuifw.note(u"Hello world!", "info")
```

To do this script you can use a text editor such as Notepad for Windows (I usually use Notepad++ for this).

After you create your small script, you must send it to the C:\Python or E:\Python (in case your phone has a memory card) so we can run it. In the example below I am connecting my phone via USB but that communication can also be done via Bluetooth connection. Note also that you must have the Nokia PC Suite installed so that everything works correctly:



Copying "ola_mundo.py" to E:\Python

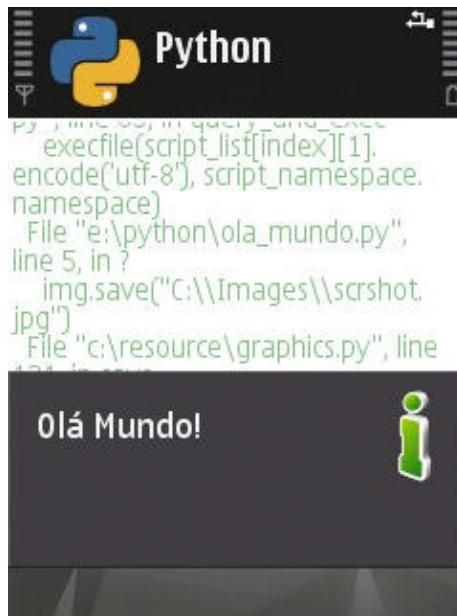
Now it is time to Run our first application for S60 Python:



Choosing "Run Script" option

Selecting our file

Once we begin our program to run it will display a dialog with our text:



Greetings!

Now that we know how to do a small PyS60 script, send it to your phone and run it- its time to present a more detailed overview of the features available for the development of Python applications for mobile devices.

The modules in PyS60

The limitations of memory and processing power of mobile devices require the mobile developer to take more care with the development of its applications. This is not different when you carry the Python interpreter for the Symbian environment.

Small changes in the interpreter make the garbage collector of Python does not make the garbage collection of objects that contain circular references (The reference B that reference A), so it is important to be careful that this type of situation does not occur.

Beyond this type of care the developer has to be careful to "turn off" the resources that are not being used and, where possible, for its program "to sleep" because doing this will reduce the processor's clock to cause a considerable saving of battery.

Events and callbacks

To facilitate the life of the programmer to develop applications that consume few resources modules Python S60 platform for the specific use and abuse of the concept of events and callbacks.

This concept is simple: the application is "sleeping" until an event occurs (button pressed, the clock signal, etc.) and shoot a light pre-determined (callback).

Functions

In addition to the concept of events and callbacks above the modules that come with PyS60 also have several functions that behave in a manner asynchronous, that is, they return before the end of performing his task.

In such cases it is quite common for these functions perform a callback function to 'warn' that the task was completed.

Specific modules of PyS60

We will talk briefly about the modules of PyS60 from now but for the sake of space we can not get into details about each of them, or even on their functions.

For a detailed view of each one of them I recommend a visit to the official documentation of Python for S60 can be found at <http://www.pythonbrasil.com.br/moin.cgi/DocumentacaoPython>.

The specific modules of PyS60 are:

E32

This module provides access to specific functions of the Symbian operating system that have no direct relationship with the interface with the user. In this module you will find functions that return the version of PyS60, if your program is running in the emulator, a list of all available drives and functions and objects that deal with locks, threads, etc..

Sysinfo

This module provides functions that return data from the device such as what the profile (general, meeting, silent, ...), the state of battery power, size of screen, free disk space, IMEI, signal strength of the network telephone, type of touch, information on the memory of the device, the firmware version, and so on.

Appuifw

In this module you will find everything that is related to the graphical interface with the user (GUI). It is one of the most important modules of PyS60.

Graphics

Module with functions for manipulating graphic images, drawing from primitive, printing of texts in images, functions for taking screenshots, etc.. This module has full interoperability with the camera modules and appuifw.

Camera

One of the most interesting of PyS60 modules for its ease of use. This module provides functions to manipulate (s) camera (s) of the cell allowing it to take pictures or videos with them is serious.

GL ES

Library that provides an API compatible with OpenGL/ES for drawing 3D graphics acceleration with (some of the S60 devices have a chip for 3D graphics acceleration).

Sensor

This module gives access to sensors, acceleration, rotation and tapping (hitting with a finger on the screen of your cell triggers the sensor). Remember that only some models of phones S60 have these sensors.

Audio

This module allows the manipulation of the total system's audio device. You can manipulate both the self-external speaker (playing an MP3, for example) as the audio of a phone call (play sound in the middle of a conversation or even write it).

Telephone

Telephony features for making a call or answering to a call are in this module.

Messaging

This module is responsible for the functions of sending SMS and MMS. inbox, contacts, calendar, inbox, contacts, calendar respectively handle the inbox of messages (SMS/MMS), contacts, calendar and the calendar of events. These modules are extremely powerful.

location , positioning

Location and Positioning are respectively for obtaining location using data from the GSM network and data from the GPS (internal/external) of the appliance.

e32db

Mini relational database that allows manipulation using SQL (will be replaced by SQLite in future versions of PyS60).

Socket

Module that is bundled with Python and received additions to support connections via Bluetooth.

Developing

The philosophy of Python says that the language has "batteries included" (batteries included) and that means that the language should always be accompanied by a standard library rather being complete and powerful.

Python for S60 is not different from standard Python. However in PyS60 we have some modules of the Python standard library (only a portion of the modules patterns) and some more specific libraries for development for S60.

Obviously, for reasons of space, we will not describe or use all modules in this tutorial but if you want detailed information about what is available for this platform, I recommend an enormously read in the official documents of PyS60 that can be downloaded at the project site listed on the first part of this tutorial (available in PDF format).

To illustrate the development an implementation of Python for S60 will use a simple example of an application that takes a picture and send via MMS to a phone number listed.

All applications PyS60 can use the skeleton below to be developed (for those who do not yet know Python I recommend reading the tutorial available in the language <http://www.pythonbrasil.com.br/moin.cgi/DocumentacaoPython> and www.python.org):

```
import e32
import appuifw

class MinhaApp(object): # define a class MinhaApp
    def __init__ (self):
        # Create an object "lock" will "hold"
        # Our application running
        self.lock = e32.Ao_lock()

    def run (self):

        # Here's our Application
        # ===== # =====

        # Assign a so-called "callback" to
        # The "exit ()" when the user
        # To choose "Exit" in cell
        # Note: note that the "exit()" is
        # Does not have the brackets because the function
        # Is not implemented immediately
        appuifw.app.exit_key_handler = self.leaves()

        # Awaiting safe until the execution
        # "Hangs" receives a signal
        self.lock.wait()
```

```
def leaves (self):
    # Sends the signal to "lock"
    self.lock.signal()

if __name__ == '__main__':
    application = MinhaApp()
    application.execute()
```

This application does nothing interesting yet it presents the basic skeleton. The only thing it does is organize a class "MinhaApp" which implements the method. "Run()" where program the option of "Exit" the application using a "ActiveObject Lock".

The "brake" is needed because the applications are developed in the universe Symbian to work in asynchronous model, i.e. the functions return immediately after being called even before they have completed their tasks. Many features of PyS60 are also implemented using the callback model, i.e. the developer combines the functions and events when these events occur the appropriate function is called. In our case the method "Exit()" runs whenever an event is triggered `exit_key` and this event is triggered when the right soft key is pressed.

Connecting the camera

One of the most interesting modules that accompanies the PyS60 is the "camera". PyS60 can easily trigger the camera from of the device, take pictures, manipulate the picture, record it on the memory card, send it to other phones and so on.

To take a picture with the camera just run the commands:

```
import camera
photo = camera.take_photo()
```

The function "Take_photo()" module of the "camera" will return an object of type "Image" containing the image to be captured/photographed. To record the image on the memory card simply call the method. "Save()" this object (note that the character "\" needs to be doubled in the path):

```
photo.save ( "E:\\Images\\minha_foto.jpg")
```

As you can see, it is very simple to capture an image in Python but it has an inconvenience: the photo is taken so that the function `camera.take_photo ()` is called but what is being photographed does not appear on the screen of your cell phone, then, you do not see what is being photographed. To see what the camera will shoot you need to trigger the camera's view finder (and close it immediately before taking the picture). Let us return to our "skeleton" of application and add a little more code to it:

```
:
import appuifw
import camera
:

class MinhaApp(object):
    def __init__(self):
        :
            # Let's create a Canvas object.
            # Canvas object allows the display
```

Introduction_to_PyS60

```
# Of images.
self.canvas = appuifw.Canvas()

def desenha_tela(self, imagem):
    self.canvas.blit(imagem)

def run(self):
    # Define the title of the application
    # The "u" before the string says
    # She is in Unicode format
    appuifw.app.title = u"PyFoto"

# Let's define that the body of the application
# Canvas is the object created above.
appuifw.app.body = self.canvas

# Started the "view finder" that will
# The image captured by the finder in Canvas
camera.start_finder(self.desenha_tela)
:
def leaves(self):
    # Off the view finder
    camera.stop_finder()
:
```

Running the script, we get the following screen:



Now let's add an option "Take picture" to our menu options. This is extremely simple to do ... let's go back to our code and add the following sentence:

```
:
class MinhaApp(object):
:
    def take_pic(self):
        # Turn off the view finder
        camera.stop_finder()

        # Take a picture and write in
```

Introduction_to_PyS60

```
# E:\Images\foto.jpg
foto = camera.take_photo()
foto.save("E:\\Images\\foto.jpg")

#start the view finder
camera.start_finder(self.function)

def execute(self):
    :
    #... after the call camera.start_finder ()

# Create a "Take picture" on the menu
# Options of cellular calls that the method
# self.take_pic() when triggered
appuifw.app.menu = [(u "Take picture," self.take_pic)]
:
```

Now you can take a picture that will be recorded in the file E:\Images\foto.jpg (for future send it via MMS):



Python

Do not be alarmed if the application to get a white screen for a few seconds because it is the time required for the serious Python the picture just taken.

```
:
import messaging

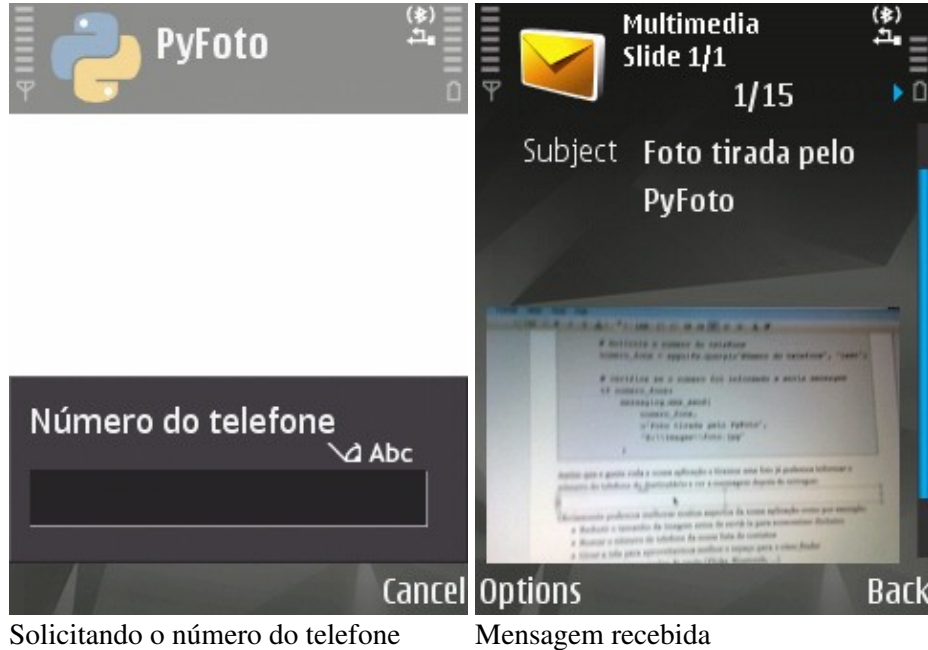
class MinhaApp(object):
    :
    def take_pic(self):
    :
        # ... Soon after photo.save (...)

# The number of phone calls
Phone_number = appuifw.query(u"number of phone", "text")
```

Introduction_to_PyS60

```
# Check if the number was informed and sends message
if Phone_number:
    messaging.mms_send(messaging.mms_send(Phone_number,u"Photo taken by PyFoto", "E:\\I
```

As soon as we run our application and take a picture we can select the recipient's phone number and see the message delivered after (careful to test the sending of MMS since this will incur charge):



Obviously we can improve many aspects of our application for example:

- Reducing the size of the image before sending it to save money
- Search the phone number of our list of contacts
- Rotate the screen to make better use of space for the view finder
- Add other options for sending (Flickr, Bluetooth, ...)

But it is as an exercise for the reader.

See Also

- [Introdução ao Python para S60](#) - Portuguese version of this article
- [PyS60 Code Examples](#)
- [PyS60 for 5th Edition + Sample applications](#)