

✓ This tick symbolizes the featured content on the Forum Nokia Wiki. Featured articles are considered to be good quality articles in the Forum Nokia Wiki, as determined by the [Forum Nokia Wiki administrators](#). Before being listed here, these articles are reviewed for accuracy, neutrality, completeness, and style.

A small tick (✓) in the top-right corner of an article indicates that the article is a featured article.

View by category: [All](#) | [Symbian C++](#) | [Open C/C++](#) | [Java](#) | [Qt](#) | **[Python](#)** | [Flash Lite](#) | [WRT Widget](#) | [Maemo](#) | [Other](#)

Configuring Eric IDE for PyS60 development

Week 38 - September 13th 2009

This week's featured article, [Configuring Eric IDE for PyS60 development](#), is created by [Sam bakki](#) and enhanced by enthusiastic PyS60 developers over the past two months.



Are you a [PyS60](#) developer? If yes, which IDE do you use for developing your applications?

It is true that even a simple text editor viz. Notepad is capable of developing full fledged [PyS60 applications](#). But how wonderful would it be to have some comprehensive facilities like debugging, auto-completion, build automation tools, etc. while writing PyS60 code? This week's featured article addresses configuring an open source IDE for PyS60 development along with its code completion feature.

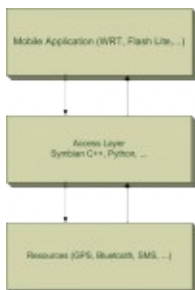
[Eric IDE](#) is a full featured Python and Ruby editor and IDE, written in python. It is based on the cross platform Qt gui toolkit, integrating the highly flexible Scintilla editor control. The article mentions clear instructions for installing and configuring the IDE for PyS60 on Windows and Mac OS.

[Read the article](#) to develop [PyS60 applications](#) with Eric IDE which has many advanced features.

How to access S60 resources in WRT or FlashLite using PyS60

Week 36 - August 30th 2009

This week's featured article, [How to access S60 resources in WRT or FlashLite using PyS60](#), is written by [Ivo Calado](#) and [Marcos Fábio](#).



The programming languages such as [Flash Lite](#) and [WRT](#), do not have full access to the device native resources such as GPS, accelerometer, etc. due to the fact that they usually run in a sandbox. An approach that

Features_Articles:_Python

fixes this problem is the development of an access layer between the device and the mobile application (written in [Flash Lite](#) and [WRT](#)). This access layer can be developed in any programming language that has native access to the device.

This article explains how to develop a mobile web server, written in [Python](#) language, whose main goal is to allow the retrieval of available native services in a Web Service REST-based approach. The web server is developed in an extensible manner that allows an easy addition of new services, decoupling the parameters handling of the service implementation.

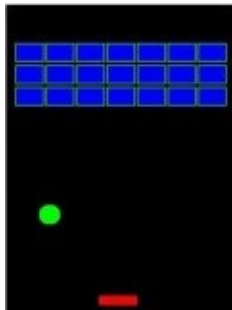
The article is a step-by-step installation guide for the required setup. It also contains a working example with the full source code and executable files.

[Read the article](#) and start accessing native [S60](#) functionalities in your [Flash Lite](#) or [WRT](#) applications.

How to develop brick-breaker game in Python

Week 32 - August 2nd 2009

This week's featured article, [How to develop brick-breaker game in Python](#), is written by [Nirpsis](#).



The gaming industry has no plans to leave your tiny mobile screen anytime soon and today's mobile games yield more realism and fun than ever. Being an easy to use tool and rightly described as a pseudo-code that runs, [Python for S60](#) allows you to develop [fascinating games and appealing fun-loving applications](#) rapidly.

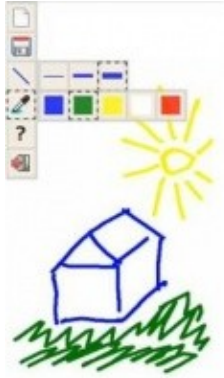
If you simply cannot wait to go through the steep learning curve of game development, have a look at our this week's [featured article](#), which illustrates the basics of developing a fully working game in [PyS60](#). The featured article series has three parts, which demonstrates, step by step, making of a simple brick-breaker game.

[Read the article](#) and start making your own mobile games. Don't forget to post them on the Wiki, we might pick it as the next [featured article](#)!

Toolbar on canvas for touch and non touch S60 devices

Week 26 - June 21st 2009

This week's featured article, [Toolbar on canvas for touch and non touch S60 devices](#), is authored by [Marcelobarrosalmeida](#).



A toolbar is a panel on which onscreen buttons, icons, menus or other input or output elements are placed. They add functionality and facilities for users, making their user experience (UX) more convenient and enjoyable.

In this article, a toolbar implementation for PyS60 applications called CanvasToolbar is presented. The toolbar can be used with [S60 3rd Edition](#) and [S60 5th Edition](#) devices, with support for moving and enabling/disabling functions. Moreover, it is also possible to create toolbars with transparency and to choose their orientation (vertical or horizontal).

The toolbar illustrated in the article has been implemented on a sample Python script, `scribble.py` for touch enabled devices, which is available [here](#).

[Read the article](#) to add toolbars to your [PyS60 applications](#).

How to display transparent PNG on canvas with masks

Week 24 - June 7th 2009

This week's featured article, [How to display transparent PNG on canvas with masks](#), is written by [Javsmo](#).



Portable Network Graphics (PNG) is a bitmapped image format that employs lossless data compression. Although, [PyS60](#) is not able to deal with images with alpha information (32-bit or RGBA), this article shows an improvised way to display transparent GIF and PNG images using the `mask` parameter in the `Image` class.

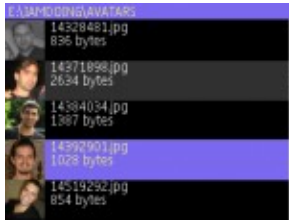
The article initially explains the problem statement and how other potential solutions like `blit` and `automask` methods are inefficient to address the problem. Finally, the article presents an efficient code snippet using the extracted the alpha channel (exported as grey-scale) to produce an image perfectly merged with its gradient background.

[Read the article](#) and start playing with transparent PNG images using [PyS60](#).

Customized listbox on canvas with images support

Week 16 - April 12th 2009

This week's featured article, [Customized listbox on canvas with images support](#), is authored by [Marcelobarrosalmeida](#).



Normally, [PyS60](#) developers use the [basic single-line item Listbox](#) or the [double -item Listbox with icons](#), offered by the `appuifw` module. However, special software requires customization. This article provides a new variety of Listbox in PyS60, which have variable number of lines per row.

The Listbox illustrated in the article is accomplished by drawing it on the Canvas and hence called a *CanvasListBox*. A *CanvasListBox* is similar to original [PyS60 Listbox](#), with methods `set_list` and `current`, generating a callback when an item is selected. The Listbox template code is provided in the article and can be used in regular PyS60 applications. A *File Explorer* application is presented in the article, to illustrate *CanvasListBox* with images support.

[Read the article](#) and customize your [PyS60 applications](#) with *CanvasListBox*.

How to use touch events with PyS60

Week 14 - March 29th 2009

This week's featured article, [How to use touch events with PyS60](#), is authored by [Jouni Miettunen](#).



Last week, [PyS60](#) 1.9.3, one of the dot releases in the 1.9.x series, was released. PyS60 1.9.3, which is updated to core Python 2.5.4, now supports touch events with the enhanced `appuifw` module. The newer version also has a new module called `sciptext` and SSL support for socket. See the release announcement [here](#). Download PyS60 1.9.3 [here](#).

This week's featured article contains source code for a small application called *Watch Me - Light Touch*, which demonstrates the use of touch events with Python. The application demonstrates the key aspects like defining active areas on-screen, capturing touch events, interpreting raw data and user actions based on them.

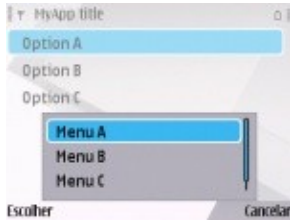
Basic knowledge of [Python for S60](#) is pre-requisite to study this article.

[Read the article](#) and add touch support to your [Python applications](#). Do not miss submitting articles about your touchy applications to the wiki - it might be our next featured article.

Basic framework for creating user interface

Week 9 - February 22nd 2009

This week's featured article, [Basic framework for creating user interface](#) authored by [Marcelobarrosalmeida](#), is one of the best articles in the [Python](#) section.



The design of an application's user interface affects the amount of effort the end-user must expend to provide input for the system & to interpret the output of the system, and how much effort it takes to learn how to do this.

This article describes how well a [PyS60](#) application can be designed to be used for its intended purpose by its target users with efficiency, effectiveness, and satisfaction, also taking into account the requirements from its context of use. The template framework presented in this article, although simple, is powerful and easy to use, and allows rapid prototyping of applications with multiple dialogs.

[Read the article](#) and learn how to implement effective UI in your [PyS60 applications](#).

Localization Example for PyS60

Week 6 - February 1st 2009

This week's featured article, [Localization Example for PyS60](#), is authored by [Marcelobarrosalmeida](#).



Localization is the process of adapting a product, in our context a mobile application, to a specific locale, i.e., to its language, standards and cultural norms as well as to the needs and expectations of a specific target market.

International mobile users expect their applications to 'talk' to them in their own language. This is not only a matter of convenience or of national pride, but a matter of productivity. Users who understand an application fully will be more skilled in handling it and avoid mistakes. So they will prefer applications in their language and adapted to their cultural environment.

This article presents an indispensable component suite for adding multilingual support to your [PyS60 applications](#). Since the default language is defined, additional translations may be added at any time. Moreover, missing translations are replaced by the default translation, allowing incremental translations without breaking the code. Users can interact with a successfully localized application in their own language which feels natural to them.

[Read the article](#) and enable your PyS60 applications to support multiple languages. This article is also available in Portuguese - [Adicionando suporte a várias linguagens em PyS60](#).

Creating C Python extensions using Carbide.c++

Week 4 - January 18th 2009

This week's featured article is [Creating C Python extensions using Carbide.c++](#) and is authored by [Diegodobelo](#).



[PyS60](#) is Nokia's port of the [Python](#) runtime to the [S60 platform](#). Although it has evolved into a very powerful programming environment, it still does not have as much functionality as the native environment, Symbian C++.

[Python extensions](#) are native modules which provide Python bindings to its native environment. Written in a programming language which directly compiles into native binary format, the extensions offer a standardized way to wrap low level machine calls and byte structures to high level programming language constructs. This article explains how to get started with creating [C extensions](#) for PyS60 using the [Carbide.c++](#) IDE.

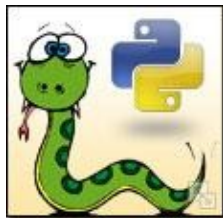
This article proves to be a step by step tutorial for building a Python extension using Carbide.c++. Taking the illustration of the uikludges extension, the article analyzes its contents and explains how to build it. On following the instructions successfully, one can result in learning basic steps required to build an extension.

[Read the article](#) and start making cool Python extensions.

Basic Python Elements

Week 42 - October 12th 2008

This week's featured article is [Basic Python Elements](#) created by [Bogdan Galiceanu](#).



[Python](#) is an easy to learn but powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms. [Python for S60](#), used in mobile phones, has been ported from the original language. It allows programmers to control features such as camera, messaging, and Bluetooth through the scripts they write.

[Basic Python Elements](#) is an article that aims to provide information about the most basic [Python](#) and [PyS60](#) operations. It is intended for beginners in the Python programming language. The article explains the use of Python elements such as variables, IDE, program syntax, arithmetic and logical operators, conditional and loops, functions, modules, etc. It also illustrates developing a simple application and running it directly on the mobile device.

[Read more in the article...](#)

How to develop a Geo-scheduler application - Part 1

Week 33 - August 10th 2008

Python is a high-level programming language used in both computers and mobile phones. Python for S60, used in mobile phones, has been ported from the original language. It allows programmers to control features such as camera, messaging, and Bluetooth through the scripts they write.

Location-based services are widely adopted and implemented in the world of mobile applications. Location awareness can be used, for example, for navigation and mapping, workforce tracking, finding points of interest, and getting weather information.

This week's featured article is about implementing location-based services in Python. How to develop a Geo-scheduler application - Part 1 is the first in a series that approaches all the basics of developing location-based applications for mobile phones in Python.

Find out more about it in the article How to develop a Geo-scheduler application - Part 1 submitted by Croozeus.
