Python Extensions in Symbian C++

- Reasons for writing extensions
  - Missing functionality
    - Accessing phone features
    - Using existing C/C++ code libraries
- Improve efficiency
  - Heavy data processing
  - Event driven instead of timer based

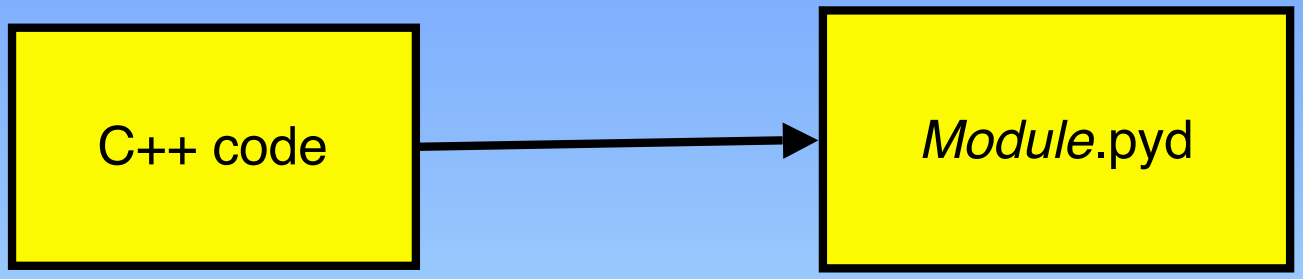
Writing a Python Extensions

- In regular Python, extensions are C programs.

- In Symbian Python **everything is the same**, but for each extension module
  - add an initialization function that is called by the Python extension loader framework.
  - add an initialization function that is called by Symbian OS DLL loader.

- Also…
  - Some Symbian C++ knowledge is needed for calling Symbian API or S60 API.

Python Extension Modules



A pyd module is a “Polymorphic DLL”.

In Symbian “Polymorphic” describes a DLL that is:

1. Loaded a runtime
2. Has one EXPORTED function.
3. Has a UID2 that identifies the *type* of interface being implemented. All Python extension modules have UID2 set to *0x1000008d.*

Extensions in regular Python

1. Include python header file: `#include <Python.h>`
2. Write the C function implementation

```
static PyObject* play_tone(PyObject* /*self*/, PyObject* args)
{  …  }
```

3. Make a table of python function names to their C function implementations.

```
static const PyMethodDef tone_methods[] = {
    {"tone", (PyCFunction)play_tone, METH_VARARGS, "play a tone."},
    {0, 0}
};
```

4. Call My_InitModule to register the *module name* with the Python runtime.

```
Py_InitModule("music", tone_methods);
```

Add some stuff for Symbian

1. The "Polymorphic DLL" entry point.
   When Python starts up it finds all extension modules and executes this initialization function from each. The module name is added to Python. The Python <u>import</u> statement adds the method table to the namespace.

```
DL_EXPORT(void) MODULE_INIT_FUNC()
{
  Py_InitModule("music", tone_methods);
}
```

2. A mandatory DLL initialization function.
   Called by the Symbian OS DLL loader framework.

```
GLDEF_C TInt E32Dll(TDllReason)
{  return KErrNone;}
```

Extensions that don't use Symbian OS API or S60 API

- Usually limited to just doing data manipulation
- Study extension_example\elemlist.cpp *[Supplied with Python SDK]*

Interfacing with Symbian OS API or S60 API

- Use C++
- `#include "symbian_python_ext_util.h"`
  - Helps with error handling

- Study examples *[Supplied at this workshop]*
  - Tone.cpp : illustrates asynchronous method calls.

  - Vibra.cpp : illustrates synchronous behavior.
    WARNING: Vibra API introduced in S60 2nd edition feature pack 2.

Prototype for Symbian C function Implementation

```
1. static PyObject* play_tone(PyObject* /*self*/, PyObject* args)
2. {
3.   TInt freqInt, durationInt;
4.   if (!PyArg_ParseTuple(args, "ii", &freqInt, &durationInt))
5.   {
6.     return NULL;
7.   }
8.   TInt64 durationInt64 = durationInt;
9.   TTimeIntervalMicroSeconds duration(durationInt64);

10.   TUint16 freq = static_cast<TUint16>(freqInt);

11.   TInt error;
12.   Py_BEGIN_ALLOW_THREADS;
13.   TRAP(error, DoPlayToneL(freq, duration));
14.   Py_END_ALLOW_THREADS;
15.   if (error)
16.       return SPyErr_SetFromSymbianOSErr(error);
17.   return Py_BuildValue("i", 1);
18.}
```

1. Parsing the arguments

2. Wrap call with ALLOW_THREADS

3. Use TRAP macro to catch exceptions.

4. Returning error status to Python.

Build Environment

- To build for phone you need
  - Platform SDK from Forum.nokia.com

- To build for WINS Emulator you need
  - Platform SDK from Forum.nokia.com
  - C++ Compiler such as Visual C++.

Installing Development SDK's on Windows PC

1. Install Active Perl 5.6.1 build 531 for SDK build tools http://activeperl.com
2. Install Java Runtime version 1.4.1_02 or later is required
3. Install the platform SDK for your phone from http://org.csail.mit.edu/mode or http://forum.nokia.com or Python Course '06 CDROM.

S60 2nd Edition, Feature Pack 2 S60_2nd_fp2_msb.zip PythonForSeries60_1_2_for_2ndEd_FP2_SDK.zip	6630, 6680, 6681	
S60 2nd Edition, Feature Pack 1 S60_sdk_2_1_NET.zip PythonForSeries60_1_2_for_2ndEd_FP1_SDK.zip	7610, 6620, 3230	
S60 2nd Edition S60_sdk_v2_0.zip	6600, 3650	

Build Environment Setup

- Make a Subst drive as the root of your platform directory.

C:\WINNT\system32\cmd.exe

```
C:\Symbian\8.0a\S60_2nd_FP2>subst p: C:\Symbian\8.0a\S60_2nd_FP2

C:\Symbian\8.0a\S60_2nd_FP2>p:

P:\>
```

Build Step 1

- Recommend copying pyext folder to platform SDK root directory – directory contains epoc32 folder.

C:\WINNT\system32\cmd.exe

```
P:\>cd pyext\music

P:\pyext\music>dir
 Volume in drive P has no label.
 Volume Serial Number is 7C73-52CF

 Directory of P:\pyext\music

01/25/2006  11:21 PM    <DIR>          .
01/25/2006  11:21 PM    <DIR>          ..
01/25/2006  08:59 PM                56 bld.inf
01/25/2006  02:09 PM               489 music.mmp
01/25/2006  09:04 PM               230 music.pkg
01/25/2006  09:05 PM             1,295 music.SIS
01/25/2006  08:51 PM             3,859 tone.cpp
               5 File(s)          5,929 bytes
               2 Dir(s)   7,652,335,616 bytes free

P:\pyext\music>
```

Build Step 2

C:\WINNT\system32\cmd.exe

```
P:\>cd pyext\music

P:\pyext\music>dir
 Volume in drive P has no label.
 Volume Serial Number is 7C73-52CF

 Directory of P:\pyext\music

01/25/2006  11:21 PM    <DIR>          .
01/25/2006  11:21 PM    <DIR>          ..
01/25/2006  08:59 PM                56 bl[illegible]f
01/25/2006  02:09 PM               489 [illegible]c.mmp
01/25/2006  09:04 PM               230 [illegible]sic.pkg
01/25/2006  09:05 PM             1,2[illegible] music.SIS
01/25/2006  08:51 PM             3,[illegible]9 tone.cpp
               5 File(s)          [illegible],929 bytes
               2 Dir(s)   7,652,[illegible]35,616 bytes free

P:\pyext\music>bldmake bldfiles

P:\pyext\music>_
```

"bldmake bldfiles"

Do this after you change bld.inf or the mmp file.

Build Step 3

ARM processor target.

Unicode Release mode build

- Abld build armi urel

Missing DEF file means you need to “FREEZE” the entry points

C:\WINNT\system32\cmd.exe

```
               2 Dir(s)   7,652,335,616 bytes free

P:\pyext\music>bldmake bldfiles

P:\pyext\music>abld build armi urel
  make -r  -f "\EPOC32\BUILD\PYEXT\MUSIC\EXPORT.make" EXPORT VERBOSE=-s
Nothing to do
  make -r  -f "\EPOC32\BUILD\PYEXT\MUSIC\ARMI.make" MAKEFILE VERBOSE=-s
WARNING: Frozen .DEF file \PYEXT\BMARM\MUSICU.DEF not found - project not froz

perl -S makmake.pl  -D \PYEXT\MUSIC\MUSIC   ARMI
  make -r  -f "\EPOC32\BUILD\PYEXT\MUSIC\ARMI.make" LIBRARY VERBOSE=-s
make -s  -r -f "\EPOC32\BUILD\PYEXT\MUSIC\MUSIC\ARMI\MUSIC.ARMI" LIBRARY
WARNING: Not attempting to create any import libraries.
When exports are frozen in "\PYEXT\BMARM\MUSICU.DEF", regenerate Makefile.
  make -r  -f "\EPOC32\BUILD\PYEXT\MUSIC\ARMI.make" RESOURCE CFG=UREL VERBOSE=

make -s  -r -f "\EPOC32\BUILD\PYEXT\MUSIC\MUSIC\ARMI\MUSIC.ARMI" RESOURCEUREL
  make -r  -f "\EPOC32\BUILD\PYEXT\MUSIC\ARMI.make" TARGET CFG=UREL VERBOSE=-s
make -s  -r -f "\EPOC32\BUILD\PYEXT\MUSIC\MUSIC\ARMI\MUSIC.ARMI" UREL
WARNING: Not attempting to create any import libraries.
When exports are frozen in "\PYEXT\BMARM\MUSICU.DEF", regenerate Makefile.
  make -r  -f "\EPOC32\BUILD\PYEXT\MUSIC\ARMI.make" FINAL CFG=UREL VERBOSE=-s
```

Build Step 4

- Freeze, build again. Only done 1st time and after EXPORTS change.

C:\WINNT\system32\cmd.exe

Freeze

Build…

```
  make -r  -f "\EPOC32\BUILD\PYEXT\MUSIC\ARMI.make"

P:\pyext\music>abld freeze armi
  make -r  -f "\EPOC32\BUILD\PYEXT\MUSIC\ARMI.make" FREEZE VERBOSE=-s
WARNING: \PYEXT\BMARM\MUSICU.DEF: File not found - OK if freezing for first time

make -s  -r -f "\EPOC32\BUILD\PYEXT\MUSIC\MUSIC\ARMI\MUSIC.ARM
EFREEZE: Appending 1 New Export(s) to \PYEXT\BMARM\MUSICU.DEF
  MODULE_INIT_FUNC__Fv @ 1 NONAME R3UNUSED ; MODULE_INIT_FUNC(void)

P:\pyext\music>abld build armi urel
  make -r  -f "\EPOC32\BUILD\PYEXT\MUSIC\EXPORT.make" EXPORT VERBOSE=-s
Nothing to do
  make -r  -f "\EPOC32\BUILD\PYEXT\MUSIC\ARMI.make" MAKEFILE VERBOSE=-s
perl -S makmake.pl  -D \PYEXT\MUSIC\MUSIC   ARMI
  make -r  -f "\EPOC32\BUILD\PYEXT\MUSIC\ARMI.make" LIBRARY VERBOSE=-s
make -s  -r -f "\EPOC32\BUILD\PYEXT\MUSIC\MUSIC\ARMI\MUSIC.ARMI" LIBRARY
  make -r  -f "\EPOC32\BUILD\PYEXT\MUSIC\ARMI.make" RESOURCE CFG=UREL VERBOSE=-s

make -s  -r -f "\EPOC32\BUILD\PYEXT\MUSIC\MUSIC\ARMI\MUSIC.ARMI" RESOURCEUREL
  make -r  -f "\EPOC32\BUILD\PYEXT\MUSIC\ARMI.make" TARGET CFG=UREL VERBOSE=-s
make -s  -r -f "\EPOC32\BUILD\PYEXT\MUSIC\MUSIC\ARMI\MUSIC.ARMI" UREL

PETRAN - PE file preprocessor V01.00 (Build 191)
Copyright (c) 1996-2004 Symbian Ltd.

  make -r  -f "\EPOC32\BUILD\PYEXT\MUSIC\ARMI.make" FINAL CFG=UREL VERBOSE=-s

P:\pyext\music>_
```

You've built the PYD! Where is it?

- The build tool chain puts the pyd under P:\epoc32\release\armi\urel
- So, in this case it's at P:\epoc32\release\armi\urel\music.pyd
- You can Bluetooth this file to your phone… or build a SIS "application installer" file.

Building an SIS

- Use the makesis command with the music.pkg file in the example folder.
- Music.pkg is a text file listing the objects to be installed on the phone.
- Send SIS file to your phone using Bluetooth or Infrared.

```
C:\WINNT\system32\cmd.exe
  make -r  -f "\EPOC32\BUILD\PYEXT\MUSIC\EXPORT.make" EXPORT VERBOSE=-s
Nothing to do
  make -r  -f "\EPOC32\BUILD\PYEXT\MUSIC\ARMI.make" MAKEFILE VERBOSE=-s
perl -S makmake.pl  -D \PYEXT\MUSIC\MUSIC   ARMI
  make -r  -f "\EPOC32\BUILD\PYEXT\MUSIC\ARMI.make" LIBRARY VERBOSE=-s
make -s  -r -f "\EPOC32\BUILD\PYEXT\MUSIC\MUSIC\ARMI\MUSIC.ARMI" LIBRARY
  make -r  -f "\EPOC32\BUILD\PYEXT\MUSIC\ARMI.make" RESOURCE CFG=UREL VERBOSE=-s

make -s  -r -f "\EPOC32\BUILD\PYEXT\MUSIC\MUSIC\ARMI\MUSIC.ARMI" RESOURCEUREL
  make -r  -f "\EPOC32\BUILD\PYEXT\MUSIC\ARMI.make" TARGET CFG=UREL VERBOSE=-s
make -s  -r -f "\EPOC32\BUILD\PYEXT\MUSIC\MUSIC\ARMI\MUSIC.ARMI" UREL

PETRAN - PE file preprocessor V01.00 (Build 191)
Copyright (c) 1996-2004 Symbian Ltd.

  make -r  -f "\EPOC32\BUILD\PYEXT\MUSIC\ARMI.make" FINAL CFG=UREL VERBOSE=-s

P:\pyext\music>makesis music.pkg
Processing music.pkg...
Created music.SIS

P:\pyext\music>_
```

Installing an Extension Module

- There are TWO ways
  - Alternative #1: Send the PYD file to the Phone
    - Build process puts the .PYD in P:\epoc32\release\armi\urel
      - Right click on the .PYD file
      - 'Send To'→Bluetooth→Phone
  - Alternative #2: Build and Send an SIS
    - You need to write a .PKG file
    - Use the makesis command.
      - "makesis music.pkg"
      - Right click on the SIS.
      - 'Send To'→Bluetooth→Phone

The quick & easy way

You can include other files in the package.

Can be uninstalled with the 'App Manager'

Using an Extension Module in Python

```
from music import *
tone(440, 250)
```

Install Python on phone

S60 2nd Edition, Feature Pack 2 PythonForSeries60_for_2ndEd_SIS.zip	6630, 6680, 6681	 
S60 2nd Edition, Feature Pack 1	7610, 6620, 3230	  
S60 2nd Edition	6600, 3650	 

Online Resources -Programming Python Extensions

- Resources
  - Python.org
    http://www.python.org
  - Python S60 SDK
    http://forum.nokia.com
  - MISO Open Source Python Extension Library
    http://pdis.hiit.fi/pdis/download/miso/
  - Documentation for writing Python extensions
    http://python.org/doc/2.2.3/ext/ext.html
  - Pyrex
    "Pyrex lets you write code that mixes Python and C data types any way you want, and compiles it into a C extension for Python."
    http://www.cosc.canterbury.ac.nz/~greg/python/Pyrex/

Summary of Key Points

- Symbian Python extensions are similar to normal python extensions.
  1. C function implementations
  2. Method table
  3. Module initialization call
- Two initialization methods must be added for Symbian.
  1. Python extension loader initialization function
  2. Symbain OS DLL loader initialization function
- Prototype C function implementation.
- Platform SDK is enough to build for phone target.
- Visual C++ or CodeWarrior needed to build for WINS emulator.
- Build steps for ARMI (phone target).
- Packaging step for making SIS file.

Workshop Exercise Pt 1

Compile and Use and extension module

- Download PYEXT.ZIP from
  http://org.csail.mit.edu/mode → IAP 2006 Class
- Extract to C:\Symbian\OS_*version**S60_version*
  for example: C:\Symbian\8.0a\S60_2nd_FP2
- Open a DOS shell
  1. C:\> **subst P: C:\Symbian\8.0a\S60_2nd_FP2**
  2. C:\> **P:**
  3. P:\> **cd pyext\music**
  4. P:\pyext\music\> **bldmake bldfiles**
  5. P:\pyext\music\> **abld build armi urel** ← *expect some errors*
  6. P:\pyext\music\> **abld freeze armi**
  7. P:\pyext\music\> **abld build armi urel**
  8. P:\pyext\music\> **makesis music.pkg**
- In File Explorer
  - left click on MUSIC.SIS and 'Send To'→ Bluetooth → Phone
  - Left click on P:\pyext\ball_tone.py and Send To→Bluetooth→Phone
- On Phone: Install both (any order) and Try Script from Python

Workshop Exercise Pt 2

Create or Modify a Python Extension Method

- In tone.cpp there is a method CToneAdapter::SetVolume(TInt aVolume) for changing the volume.

- Exercise steps:
  1. Modify tone.cpp so that you can control the volume.
     - Suggestions:
       - Make another method similar to 'tone' that takes a volume argument.
       - Or add a third argument to the existing 'tone' method.
  2. Modify a Python script to use your new method.
     - Suggestions:
       - Add sound to the snake.py
       - Modify ball_tone.py

(solution is provided in pyext\music\sol\)