CS 219 - Lecture 2
Python S60 Tutorial and Homework

Sasank Reddy

Overview

Getting Started

- Go over programming languages, install python on your phone, etc..

Programming via PyS60

- We will write a series of short scripts to start you off.

User Interface Framework

- We will explain the user interface widgets that can be used for interaction.

Additional Libraries

- We will talk about some additional libraries that might be useful.

Resources

- We will overview some resources that might be useful to continue your learning.

Programming Languages on Mobile Phones

Java (MIDP)
- Major Sandboxing
- Functionality growing with MIDP2, but still not quite there.

C++ (Symbian)
- Very steep learning curve
- Frustrating features
- Fast

Python
- Cross platform, Open Source, Scripting Language
- Good standard libraries (full access to phone functionality)
- Easy

Before you get started....

Upgrade your phone's firmware!
Upgrade your phone's firmware!
Upgrade your phone's firmware!

This is very important, otherwise many of the features you need will not be present. You should download the Nokia Software Updater for the Nokia n95 (google it).

- Upgrading the phone will clear all stored files on your phone (both on the external storage and on the main disk).

- All applications will also be deleted.

- Lots of benefits including being able to use the "sensor" module and enabling A-GPS.

Installing PyS60

Installing Python for S60 consists of downloading a runtime package and a script shell.

Python runtime package contains a Python interpreter DLL, standard and proprietary Python library module, S60 UI application framework.

Python script shell package contains an application written in Python and visible in the application menu of the device that provides an execution environment for Python scripts.

Both can be downloaded from Forom Nokia Website:
http://discussion.forum.nokia.com/forum/showthread.php?t=125168

A plugin for the S60 C++ SDK is also available which makes it possible to run Python scripts in the S60 emulator environment and to compile Python extension modules (PYDs) for the emulator and the device.

Installing PyS60

The latest version is Python for S60 1.4.2. For location services to work, you will need to sign the Python interpreter and script shell before installing it. So download the unsigned versions, sign it with your key, and then install it.

1. Signup for SymbianSigned (Symbian Developer Certificate Request Process Link)
http://developer.symbian.com/main/tools/devtools/critical/index.jsp#devcert

2. Download DevCert to generate key from (generates a cert):
http://developer.symbian.com/main/tools/devtools/critical/dev_cert_request/index.jsp

3. Request DevCert at (generates a key):
https://www.symbiansigned.com/app/barclayhtml/devcert/requestupload.jsp

4. Download Ensymble or SignSIS.exe to Sign Module
http://www.nbl.fi/~nbl928/ensymble.html
http://www.forum.nokia.com/main/resources/tools_and_sdks/index.html

- ensymble signsis --cert=mycert.cer --privkey=mykey.key --passphrase=mypassword unsigned.sis signed.sisx
- signsis unsigned.sis signed.sisx mycert.cer mykey.key mypassword

First Script - “Hello World”

Use a text editor on Mac or PC to write your python script.   Basically, create a file with the following lines:

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

Save your script as “helloworld.py”

Test your script:

- By pushing the python file to the E:\Python\ directory.
- Then, start up Series 60 Python interpreter application.
- Press “options”, select “Run Script” and select “helloworld.py”.

First Script - “Hello World”

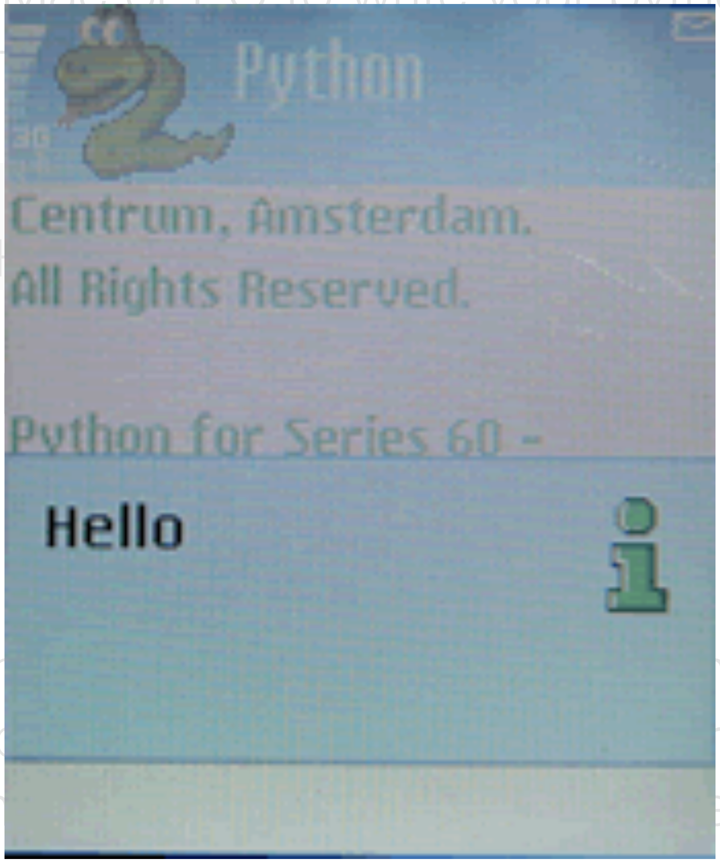
Use a text editor on Mac or PC to write your python script.   Basically, create a file with the

```
import appuifw
appuifw.note(u"H
```

Save your script as “

Test your script:

- By pushing th n\ directory.
- Then, start u r application.
- Press “optior elect “helloworld.py”.



Text Input with PyS60

Program an application that does the following:

- Display a text input field on the screen to input a text string.

- Display a pop up note stating the result from the previous input.

Program looks like this:

```
# 1. import the application user interface framework module
import appuifw

# 2.  create a text input field:  appuifw.query(label, type) and variable
data = appuifw.query(u"Type a word:", "text")

# 3.  create a pop-up note: appuifw.note(label, type)
appuifw.note(u"The typed word was: " + data, "info")
```

Text Input with PyS60

Program an application that does the following:

n the sc

ting the

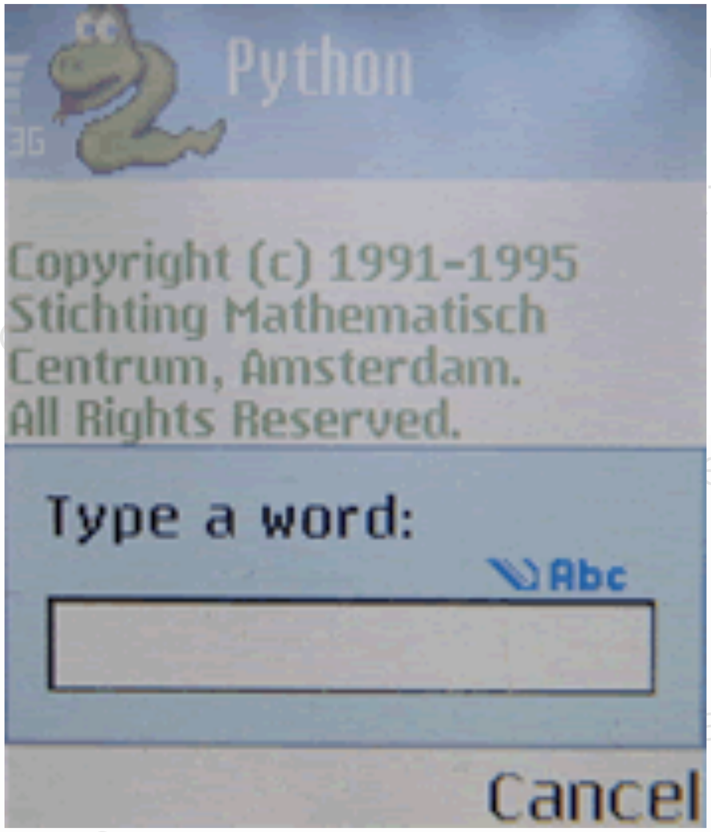
t.

Pr

er interfa

appuifw

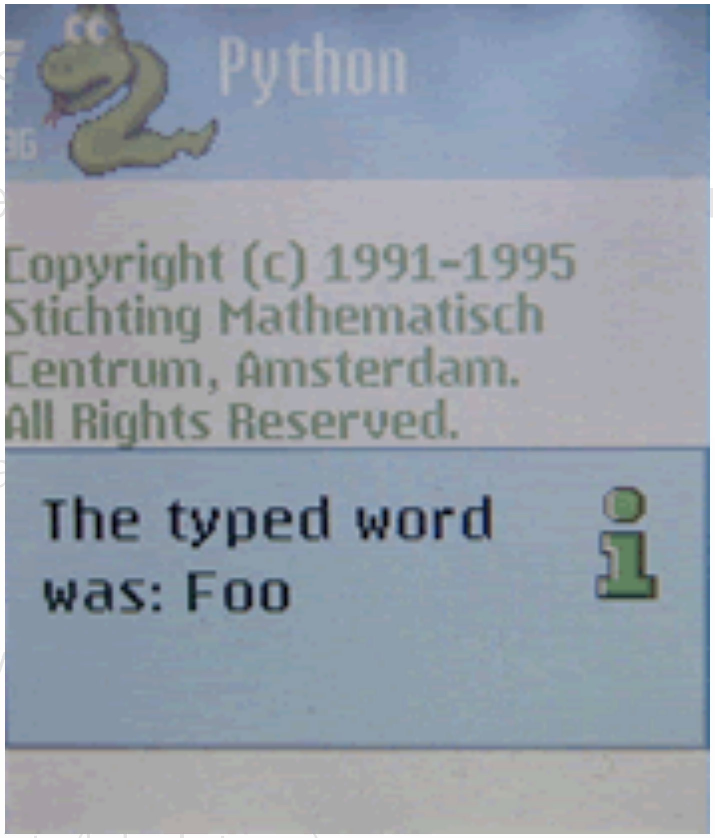
a word:"



Python

Copyright (c) 1991-1995
Stichting Mathematisch
Centrum, Amsterdam.
All Rights Reserved.

Type a word:



Python

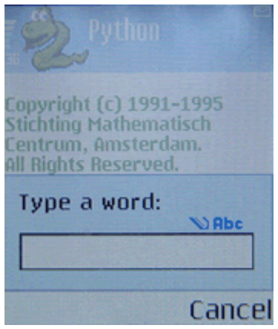
Copyright (c) 1991-1995
Stichting Mathematisch
Centrum, Amsterdam.
All Rights Reserved.

The typed word was: Foo

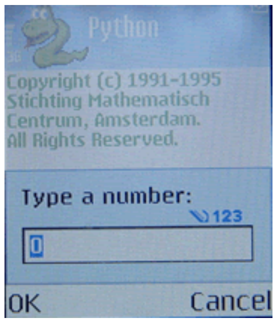
```
# 3.  create a pop-up note: appuifw.note(label, type)
appuifw.note(u"The typed word was: " + data, "info")
```

Different Types of Inputs Available

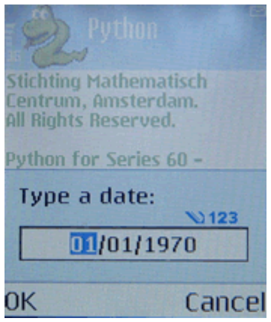
```
appuifw.query(u"Type a word:", "text")
appuifw.query(u"Type a number:", "number")
appuifw.query(u"Type a date:", "date")
appuifw.query(u"Type a time:", "time")
appuifw.query(u"Type a code:", "code")
appuifw.query(u"Are you ok:", "query")
```


Python

Copyright (c) 1991-1995
Stichting Mathematisch
Centrum, Amsterdam.
All Rights Reserved.

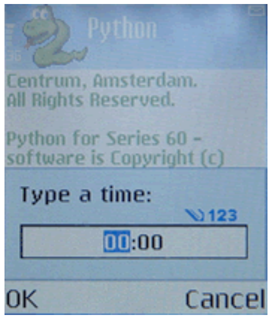

Python

Copyright (c) 1991-1995
Stichting Mathematisch
Centrum, Amsterdam.
All Rights Reserved.


Python

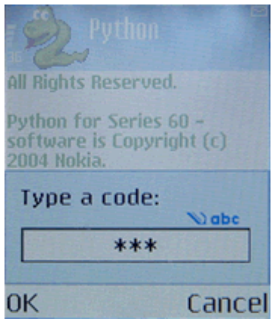
Stichting Mathematisch
Centrum, Amsterdam.
All Rights Reserved.

Python for Series 60 -


Python

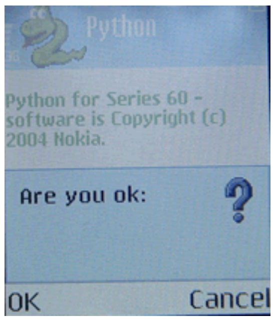
Centrum, Amsterdam.
All Rights Reserved.

Python for Series 60 -
software is Copyright (c)


Python

All Rights Reserved.

Python for Series 60 -
software is Copyright (c)
2004 Nokia.


Python

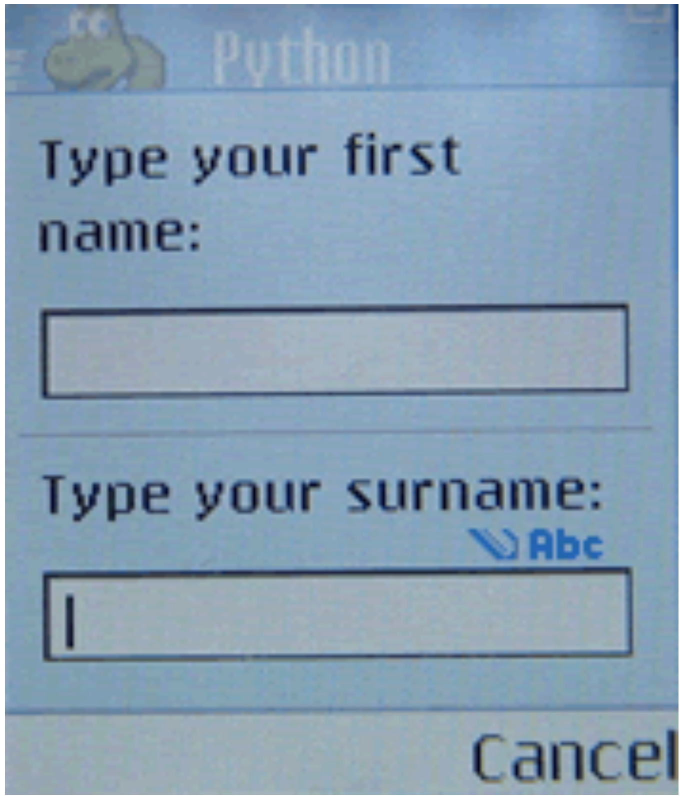
Python for Series 60 -
software is Copyright (c)
2004 Nokia.

Different Types of Inputs Available

Wouldn’t it be cool to do multiple queries on one screen?

Well you can! Can you believe it!

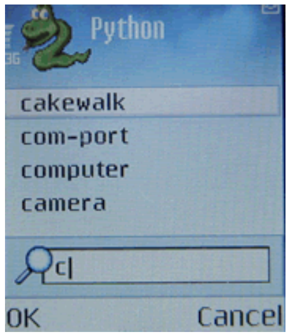
```
import appuifw

data1,data2 =
appuifw.multi_query(u"Type your
first name:",u"Type your
surname:")
```

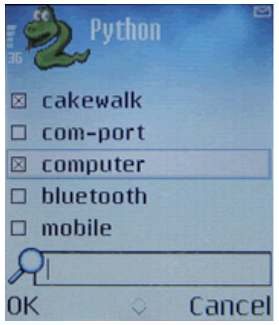


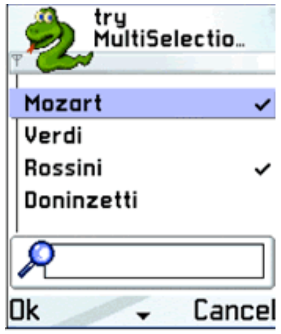
Different Types of Inputs Available

What about multiple choices for one question? Yup, that can be done as well.

```
import appuifw
# define the list of items (put a u in front - must be written in unicode!)
L = [u'cakewalk', u'com-port', u'computer', u'mobile', u'screen'',
u'keys']
# create the selection list
index = appuifw.selection_list(choices=L , search_field=1)

# or chose multiple using checkbox
index = appuifw.multi_selection_list(L , style='checkbox',
search_field=1)

# or choose multiple using checkmark
index = appuifw.multi_selection_list(L , style='checkmark',
search_field=1)
```







Tabs for Organization

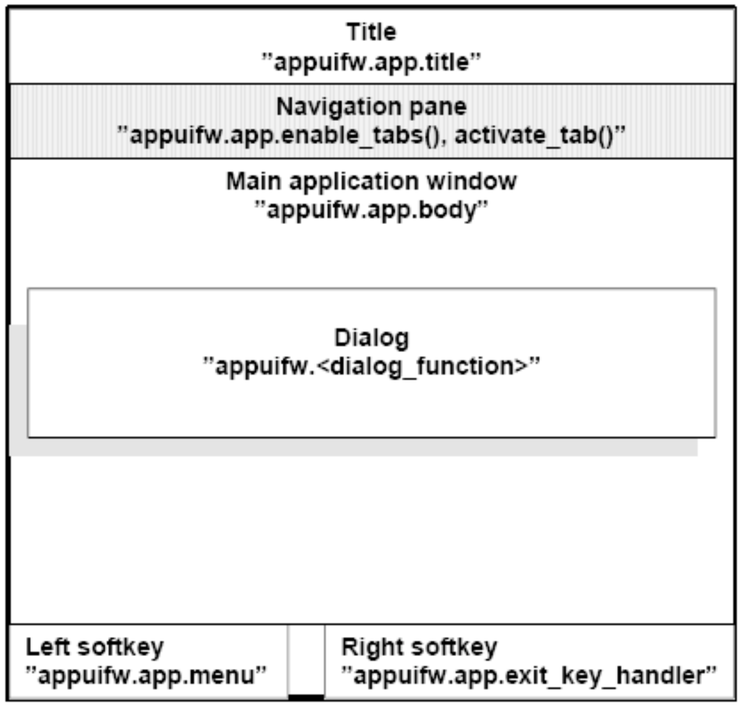
```
appuifw.app.set_tabs(tabnames, callback)

# define application 1: text app
app1 = appuifw.Text(u'Appliation o-n-e is on')
# define application 2: text app
app2 = appuifw.Text(u'Appliation t-w-o is on')

def exit_key_handler():
    app_lock.signal()

# create a tab handler that switches the application based on what tab is selected
def handle_tab(index):
     if index == 0:
        appuifw.app.body = app1 # switch to application 1
    if index == 1:
        appuifw.app.body = app2 # switch to application 2

# create the tabs with its names in unicode as a list, include the tab handler
appuifw.app.set_tabs([u"One", u"Two"],handle_tab)
```

Tabs can be used to switch application pages.

Formalizing Building of an Application

Building an application is a nine step process.

- import all modules needed
- set the screen size (normal, large, full)
- create your application logic
- create an application menu
- set an exit key handler
- set the application title
- deal with active objects
- set the application body
- create a main loop if suitable



Application Creation: Import Modules

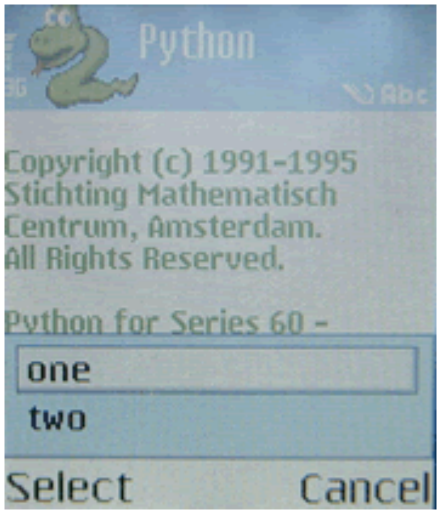
Syntax consists of import module_name, Several different modules that can be imported.

appuifw - interface to the S60 GUI framework
e32 - OS related functional package
graphics - graphics related services package
camera - interface for taking photographs
audio - an audio related service package
telephone - telephone service for making / receiving calls
messaging - messaging services package.
e32db - interface to the Symbian native DB.
contacts - access to the contacts (phone book)
calendar - access to calendar related services
location - GSM location information
positioning - GPS location information
sensor - accelerometer access

Application Creation: Set Screen Size

Three different screen size options

```
normal - screen with title pane and softkeys
appuifw.app.screen='normal'

large - only softkeys visible
appuifw.app.screen='large'

full screen
appuifw.app.screen='full'
```



Copyright (c) 1991-1995
Stichting Mathematisch
Centrum, Amsterdam.
All Rights Reserved.

Python for Series 60 -

one
two

Application Creation: Application Logic

Well, this is up to you.

Examples of programs that we implemented thus far in Python include:

- pyCampaignr: Enables us to capture many different sensing values automatically or with user intervention.

- audioSampler: A program that samples a snippet of audio and then transfers the information to SensorBase.org.

- ecoPDA: A data capturing tool for Conservational International that gathers information about butterflies in traps. This information is initially saved on the phone and later uploaded to SensorBase.org.

Application Creation: Application Menu

Application menu uses the left softkey and can always be accessed while your application is running.

Furthermore, the application menu can contain a submenu.

```
def item1():
    print "item one"
def subitem1():
    print "subitem one"
def subitem2():
    print "subitem two"

appuifw.app.menu = [(u"item 1", item1), (u"Submenu 1", ((u"sub item 1",
subitem1), (u"sub item 2", subitem2)))]
```

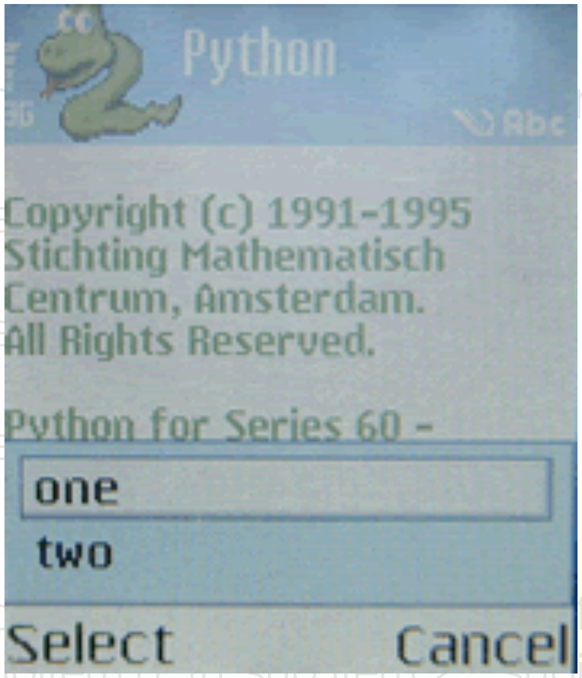
We are basically using appuifw.app.menu where the first entry is the title and the second is the callback function.

Application Creation: Application Menu

Application menu uses the left softkey and can always be accessed while your application is running.

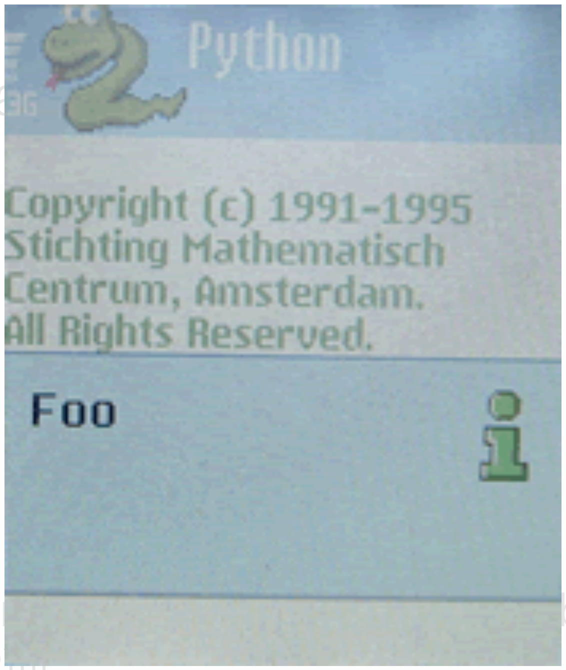
Furtherm nu ca

```
de

de

de

ap                    1", ite                                  o item 1",
subitem1), (u"sub item 2", subitem2)))]
```



Copyright (c) 1991-1995
Stichting Mathematisch
Centrum, Amsterdam.
All Rights Reserved.

Python for Series 60 -



Copyright (c) 1991-1995
Stichting Mathematisch
Centrum, Amsterdam.
All Rights Reserved.

We are basically using appuifw.app.menu where the first entry is the title and the second is the callback function.

Application Creation : Exit Key and Title

The exit key handler gets activated when you press the right (exit) softkey.

This can be used to perform clean up before the application is terminated or ask for other information from the user.

```
def quit():
    appuifw.app.set_exit()
app.exit_key_handler=quit
```

You can also set the application title for the script.

```
appuifw.app.title = u"SMS sending"
```

Application Creation : Active Objects

A facility called “active object” is used extensively on the Symbian OS to implement cooperative non-preemptive scheduling within operating system threads.

It preserves the responsiveness of the UI and enables other applications to still run.

```
import e32

# create an instance of the active object
app_lock = e32.Ao_lock()

# starts a scheduler, the script processes events until lock.signal() is called
app_lock.wait()

# stops the scheduler
app_lock.signal()
```

Application Creation : Body

The designer has the ability to add different elements to the application body.

```
Listbox, text, or a canvas for instance.

# body as Listbox:
appuifw.app.body = appuifw.Listbox(entries,shout)

# body as Text:
appuifw.app.body = appuifw.Text(u'hello')

# body as Canvas:
appuifw.app.body=appuifw.Canvas(event_callback=None,
redraw_callback=handle_redraw)
```

Application Creation : Main Loop

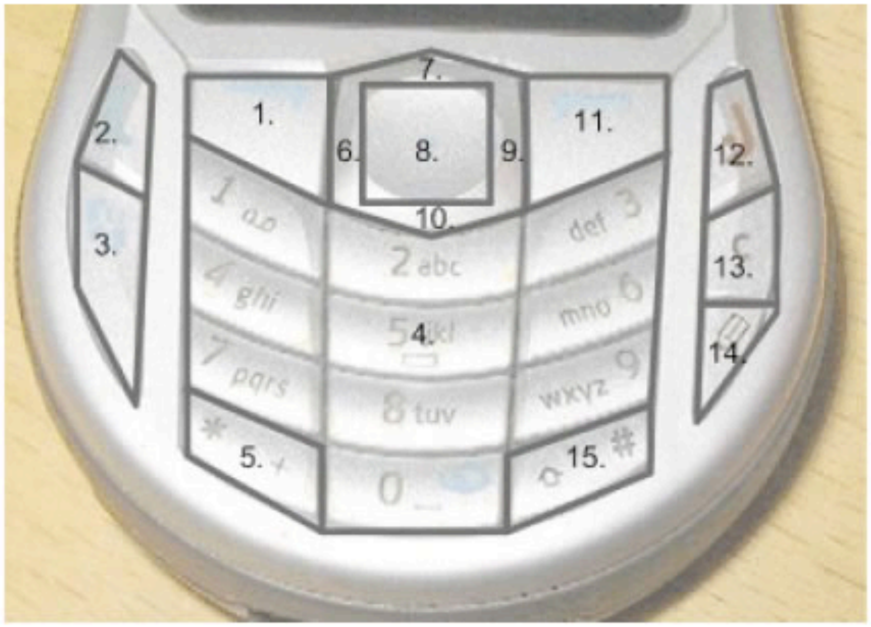
You might employ a main loop functions that need to be run again and again in the script.

This is useful for UI type elements or doing periodic updates.

```
# create a main loop (e.g. redraw the the screen again and again)

while running:
    # put things that need to be run through again and again
    # e.g. redraw the screen:
    handle_redraw(())
```

Miscellaneous : Keyboard Keys

1. EKeyLeftSoftkey
   EScancodeLeftSoftkey
2. EKeyYes
   EScancodeYes
3. EKeyMenu
   EScancodeMenu
4. EKey1...9,0
   EScancode1...9,0
5. EKeyStar
   EScancodeStar
6. EKeyLeftArrow
   EScancodeLeftArrow
7. EKeyUpArrow
   EScancodeUpArrow
8. EKeySelect
   EScancodeSelect



9. EKeyRightArrow
   EScancodeRightArrow
10. EKeyDownArrow
    EScancodeDownArrow
11. EKeyRightSoftkey
    EScancodeRightSoftkey
12. EKeyNo
    EScancodeNo
13. EKeyBackspace
    EScancodeBackspace
14. EKeyEdit
    EScancodeEdit
15. EKeyHash
    EScancodeHash

Figure Keycodes and scancodes for phone keys usable from Python applications

Miscellaneous : Networking

Can use FTP....

```
from ftplib import FTP
picselection = 'c:/testimg.gif'

ftp = FTP('www.exampleserver.com') # connect to host
ftp.set_pasv('true')
ftp.login('username','password') # login anonymous
ftp.cwd('public_html/examplefolder') # change directory where to store the
image
F=open(picselection,'r')
ftp.storbinary('STOR image.gif',F,1024) # store the image
ftp.quit()
F.close()
```

Miscellaneous : Networking

Can use HTTP....

```
import httplib
import urllib

content = "hello"
params = urllib.urlencode({'data': content, 'eggs': 0, 'bacon': 0})
headers = {"Content-type": "application/x-www-form-urlencoded",
"Accept": "text/plain"}

conn = httplib.HTTPConnection("www.exampleserver.com")
conn.request("POST", "/examplefolder/example.php", params, headers)

response = conn.getresponse()
conn.close()
```

Miscellaneous : Accelerometer

Can query the Accelerometer

```
import sensor

def data(self, stuff):

   x = stuff["data_1"]
   y = stuff["data_2"]
   z = stuff["data_3"]

# Fetch data from accelerator
sensors = sensor.sensors()
self.acc = sensor.Sensor(sensors["AccSensor"]["id"],
                         sensors["AccSensor",["category"])
self.acc.connect(self.data)
```

Additional Resources

Python for S60 Book

MobileNin - Lots of examples, tutorials, etc...
- http://www.mobilenin.com/pys60/

Nokia Python for S60
- http://forum.nokia.com/python
- http://sourceforge.net/projects/pys60

Google
- Lots of examples all over the internet just search for them.

Homework #1 : Capture - Due Wednesday 4/9

Objective

Write a **program using PyS60** on the Nokia n95 to **automatically gather a location trace** (GPS and GSM) every **1 second**. Furthermore, when a user **presses a button**, you should enable the user to capture a photo, record a second of audio, and annotate the media files. The location trace along with the media file with annotation and timestamps should be **uploaded to SensorBase**. Then you need to create a **webpage that summarizes the location trace** - can include, the total distance traveled, the image/audio/annotations in a table by themselves, etc...

Submission

You should create a webpage to post all your homework. For this particular homework, you need to **provide a link to the source code** (zipped up), and also provide a link to the **webpage where we can view the current summary for the trace you generated**.

Homework #1 : Capture - Due Wednesday 4/9

Objective

Your task is to write a pro matically
gather a location trace (G when a
user presses a button, yo record
a second of audio, and p . This
information should be uplo reate a
webpage that summarize n can
include, the total distance ble by
themselves, etc...

Python script for phone should be easy.

Challenge comes from interacting with SensorBase. You need to log data via the phone and retrieve data via a webpage.

Examples code exists on ...
http://sensorbase.org

Submission

You should create a webpage to post all your homework. For this particular homework, you should provide a link to the source code (zipped up), and also provide a link to the webpage where we can view the current summary for the trace you generated.

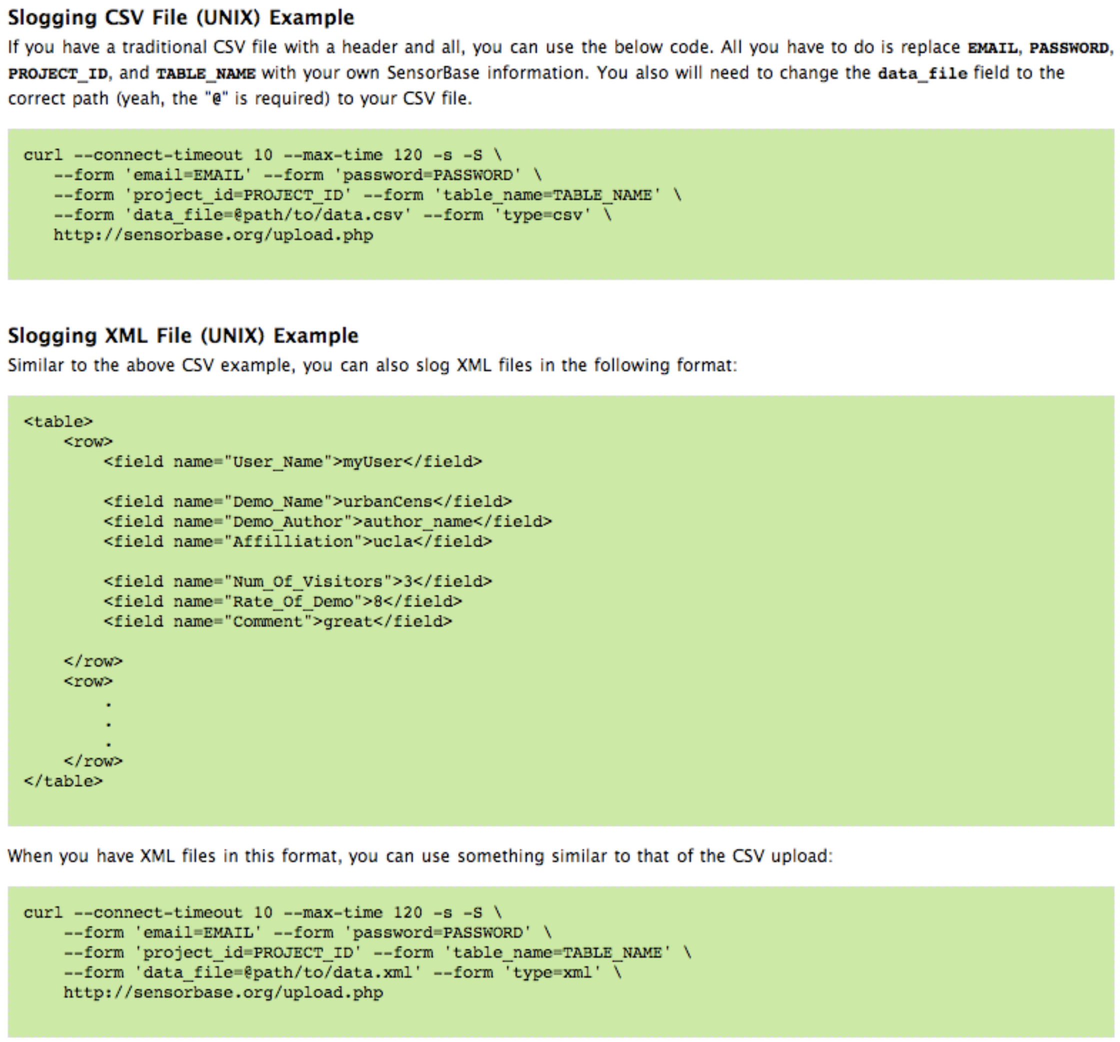
Homework #1 : Capture - Due Wednesday 4/9

Object

Your tas … to automatically
gather a … ermore, when a
user pre … a photo, record
a secor … edia files. This
informat … eed to create a
webpag … nformation can
include, … ns in a table by
themsel

Submi

You shc … r this particular
homewo … d up), and also
provide … ummary for the
trace yc


Slogging CSV File (UNIX) Example

If you have a traditional CSV file with a header and all, you can use the below code. All you have to do is replace **`EMAIL`**, **`PASSWORD`**, **`PROJECT_ID`**, and **`TABLE_NAME`** with your own SensorBase information. You also will need to change the **`data_file`** field to the correct path (yeah, the "`@`" is required) to your CSV file.

```
curl --connect-timeout 10 --max-time 120 -s -S \
    --form 'email=EMAIL' --form 'password=PASSWORD' \
    --form 'project_id=PROJECT_ID' --form 'table_name=TABLE_NAME' \
    --form 'data_file=@path/to/data.csv' --form 'type=csv' \
    http://sensorbase.org/upload.php
```

Slogging XML File (UNIX) Example

Similar to the above CSV example, you can also slog XML files in the following format:

```
<table>
    <row>
        <field name="User_Name">myUser</field>

        <field name="Demo_Name">urbanCens</field>
        <field name="Demo_Author">author_name</field>
        <field name="Affilliation">ucla</field>

        <field name="Num_Of_Visitors">3</field>
        <field name="Rate_Of_Demo">8</field>
        <field name="Comment">great</field>

    </row>
    <row>
        .
        .
        .
    </row>
</table>
```

When you have XML files in this format, you can use something similar to that of the CSV upload:

```
curl --connect-timeout 10 --max-time 120 -s -S \
    --form 'email=EMAIL' --form 'password=PASSWORD' \
    --form 'project_id=PROJECT_ID' --form 'table_name=TABLE_NAME' \
    --form 'data_file=@path/to/data.xml' --form 'type=xml' \
    http://sensorbase.org/upload.php
```

Things you need to do

- Check out a Nokia n95 phone from the CENS Main Office (you have until 4:45 p.m. today).
- Create a SensorBase account. Visit http://sensorbase.org for details (do this today).
- Make sure you have access to a server that you can host a webpage (your CS, EE, or SEAS account needs to be activated).
- Sign up for the mailing list (do this today). http://www.cens.ucla.edu/mailman/listinfo/cs219sp08