

Mobi Vision '10

OS Ankur Shrivastava
Appan Anand

Why/What/How?

- We will learn about mobile application development
- Basically focusing on Symbian S60
- We will use Python (Python for S60)
- At the end of the workshop you will be able to understand/identify problem(s) related to cell phones and solve them

Who are we?



- Linux User's Group Manipal
- Life, Universe and FOSS!!
- Believers of knowledge sharing
- Most technologically focused "group" in University
- This workshop is not an official LUG workshop, and is being organized under Techtatva '10
- <http://lugmanipal.org>

Whats the plan??

- Getting started with Python
- Learning about basic of mobile phones and their (diminishing) limitations
- Understanding Symbian S60
- Starting with Python for S60
- Making your first app
- Finishing with many working prototype apps

Important Points

- If you have problem(s) don't hesitate to ask
- Slides are based on Documentation so discussions are really important, slides are for later reference!!
- Please don't consider sessions as Class (i hate classes)
- Speaker is just like any person sitting next to you
- Documentation is really important
- Google is your friend
- If you have questions after this workshop mail me or come to LUG Manipal's forums

<http://forums.lugmanipal.org>

Why So Serious??



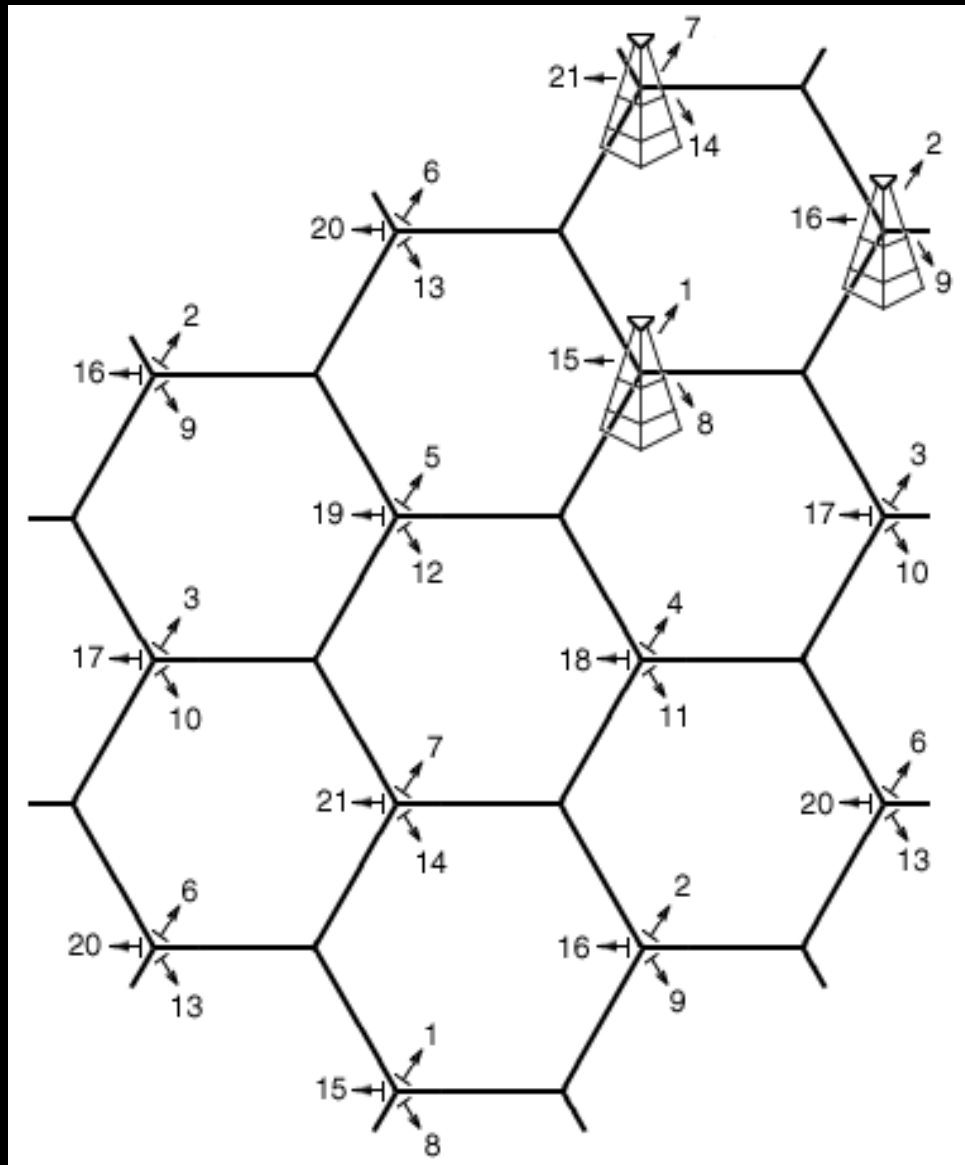
What all do we need ?

- See MobiVision\install files\
- Install Perl
- Install python
- Run vista patch if you have vista/win 7
<drive>:\S60\devices\S60_3rd_FP2_SDK_v1.1\Plugins\vistapatch
- install python for s60 setup on windows
- install python on cell

Python slides

==>

How do phones work?



- Each cell unit is hexagonal in shape
- Cellular tower at end of each cell
- GSM is FDMA(!)
- what is handover(!)
- GSM vs CDMA
- What is UMTS??
- 3G in India ?

What is SYMBIAN ?



<http://symbian.org>

- Symbian OS a very popular mobile OS by Symbian Ltd.
- Descendant of Psion's EPOC
- Nokia acquired Symbian Ltd fully in June 2008 and announced Symbian Foundation, making Symbian OS Open Source

Importance of Phones

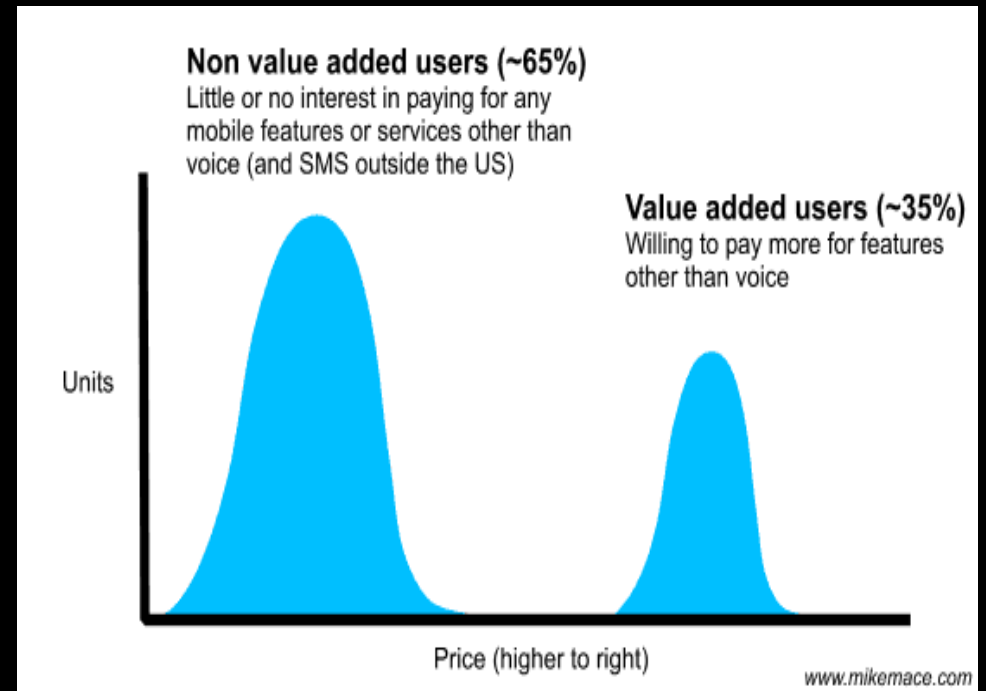


Because you cannot always carry everything!!!

- Small, cheap and efficient means of communication
- Almost always connected
- Not just communication device anymore
- Music/Video Player, Web Browser, Games, E Mail, IM, etc...
- Unique ways of usage being found everyday!!!

Data is Important

- Phones have come a long way from just analog communication device to always connected high speed digital device
- Importance of Data is increasing day by day, 1G → 2G → 2.5G → 3G → 3.5G → 4G



So Lets Begin...

Simple GUI

appuifw module

- APPlication User Interface FrameWork
- Used for interacting with user in GUI
- We will cover everything one by one
- Dialog ->
 - note
 - query
 - multi_query
 - selection_list
 - multi_selection_list
 - popup_menu

note

- Function \rightarrow `note(text[,type[,global]])`
- Displays a note dialog to user with text
- Type \rightarrow info, error, conf (default info)
- Global \rightarrow boolean (int) use 0 or 1
- Type selects the type of dialog
- If global is set, dialog is displayed even if application in background

query

- Function → `query(label,type[,initial_value])`
- Displays a single field query dialog, with prompt set to label, of specified type
- Supported types → text, code, number, date, time, query, float
- `initial_value` → sets the initial value of the query

multi_query

- Function → `multi_query(label1, label2)`
- Displays a 2 field query with label1 and label2
- Returns a tuple with len 2 in unicode
- Returns None if canceled

popup_menu

- Function \rightarrow `popup_menu(list[,label])`
- Displays a popup type dialog
- list is a list of unicode strings or unicode string pairs as tuples
- Returns index of when selected
- Returns None when user cancels

selection_list

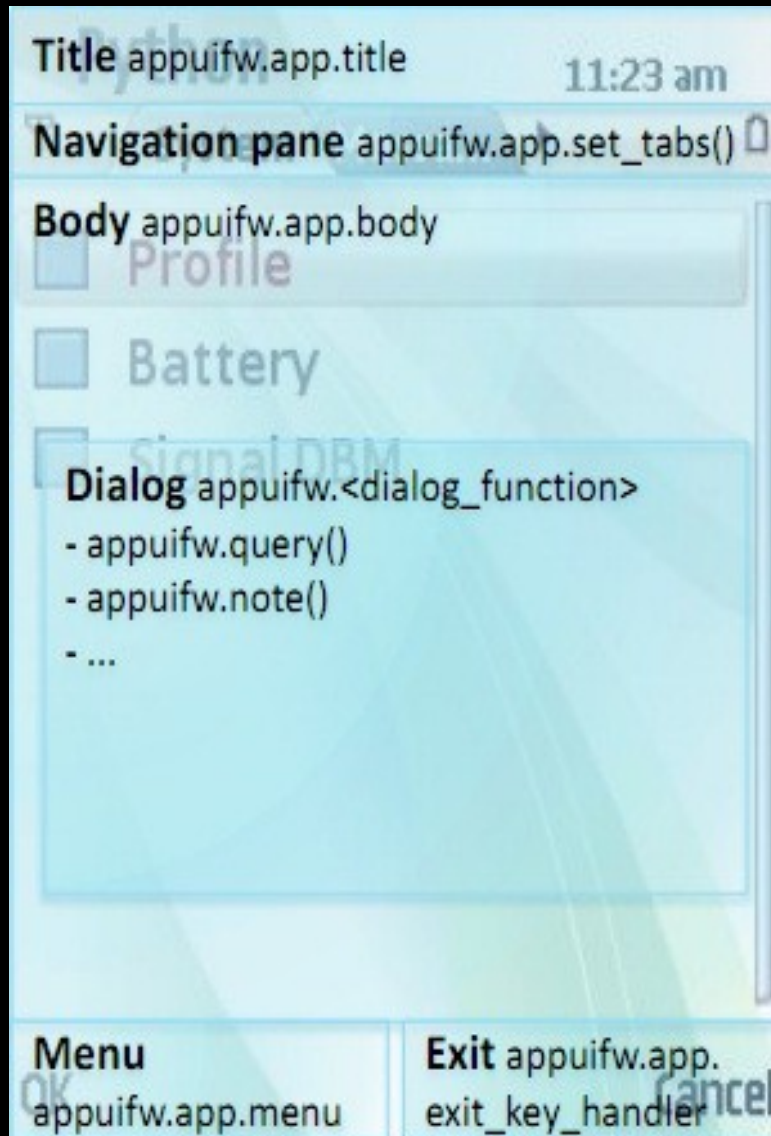
- Function → `selection_list(choices, [search_field=0])`
- Allows a user to select an item from a list of choices and returns the index
- Choices is list of unicode strings `['a', 'b']`
- Setting `search_field` to 1 enables the find pane, allowing the user to search through choices

multi_selection_list

- Function → `multi_selection_list(choices[, style='checkbox', search_field=0])`
- Allows user to select multiple items from list
- Returns a tuple of index or empty tuple on cancel
- Supported style → checkbox, checkmark
- Setting `search_field` to 1 enables the find pane, allowing the user to search through choices

Lets move to
Applications

Basic structure of an app



- Title → title of the application
- Tabs → multiple tabs for the application
- Body → main part of the application
- Menu → Command/option selection menu
- exit_key_handler → default exit handler, for application exit

appuifw (cont)

- In `appuifw.app` , `app` is the default instance of class `Application`
- Only one instance of `Application` Class i.e `app` exists in any running application
- `appuifw.app.title = u"abc"` #sets the title
- `appuifw.app.screen = "large"` #sets the screen size
- Supported screen sizes → normal, large, full, full_max

appuifw (cont)

- `appuifw.app.focus` → set it to callback of a single argument function
- Called with argument 0 when application set to background (loses focus)
- Called with argument 1 when application gains focus again
- `appuifw.app.orientation` → changes the orientation of the application
- Orientation options → landscape and portrait
- Only supported on S60 3rd Edition

appuifw.app.menu

- displays option menu for selection by user on left softkey press
- List of name, callback tuples
- Name - u'name to be displayed'
- Callback - <function callback name>
- Can have nesting for submenus
- Maximum number of items in a menu or submenu is 30

Ao_lock class

- Present in module e32
- active object based synchronization service, enables non blocking UI handling
- Two functions wait() and signal()
- Wait() → blocks in wait for lock to be signaled
- signal() → signals the lock, releases the waiter

Ao_lock class (cont)

- Dont worry if you dont understand everything
- Just remember
 - Create a lock object and call wait() to stop the script execution, all the UI event will still be handled
 - Call signal() to signal the lock and release the waiter allowing script to proceed
- Code →

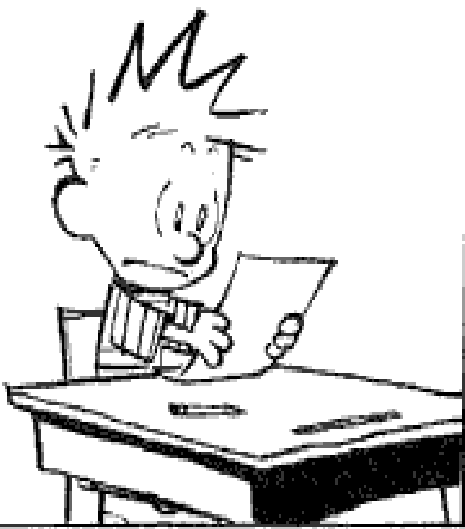
```
app_lock = e32.Ao_lock()
```

```
app_lock.wait()
```

Time for questions

Please stay sensible

1. Explain Newton's First Law of Motion in your own words.



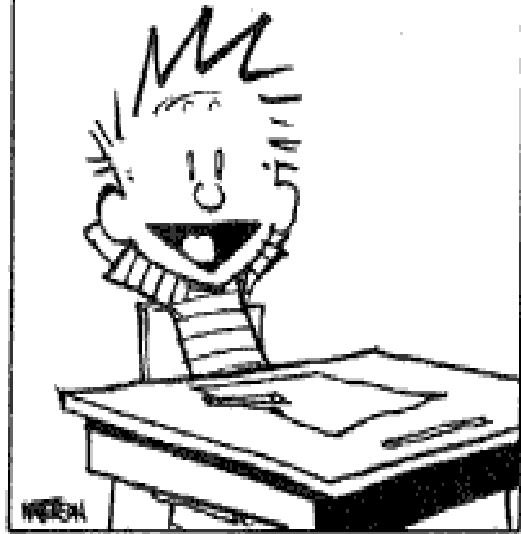
!



Yakka Foob Mog. GRUG
PubbaWup zink wattooM
Gazork. CHUMBLE Spuzz.



I LOVE
LOOPHOLES.



Question:-

Make an app to manage a shopping list, the application should allow the user to add/remove individual items, it should also allow user to mark multiple item as bought and display bought and remaining items.

Time: 10 min

Application tabs

- Function →
`appuifw.app.set_tabs(tab_texts[,callback=None])`
- Activates tabs in the current application
- `tab_text` → list of tab names
- `Callback` → function called on tab change with `tabindex` as argument
- Passing `tab_texts=[]` disables tabs
- `appuifw.app.activate_tab(index)` → activates tab at index

Application Body

- `appuifw.app.body` → UI control that is visible in the application's main window
- `supports` → Text, Listbox, Canvas or None
- Text → allows you to have a text editor like main window
- Listbox → displays text/icons as a list
- Canvas → allows creations of arbitrary figures, just like a normal canvas

Text Type

- Class → `appuifw.Text()`
- Instance methods →
 - `add(text)` → inserts unicode text at current position
 - `clear()` → clears the editor
 - `len()` → returns the length
 - `get([pos=0, length=len()])` → retrieves text of length starting from `pos`
 - `set(text)` → sets inside text to `text`
 - `get_pos()` → returns the current cursor position
 - `set_pos(pos)` → sets current cursor position to `pos`
 - Read Documentation for more...

Listbox type

- `appuifw.ListBox(list, callback)`
- list of 4 types →
 - `[u'a', u'b', u'c']` → normal item names
 - `[(u'a', u'description'), (u'b', u'description'), (u'c', u'description')]` → item and its description
 - `[(u'a', icon_a), (u'b', icon_b), (u'c', icon_c)]` → item and its icon
 - `[(u'a', u'description', icon_a), (u'b', u'description', icon_a), (u'c', u'description', icon_a)]` → item, its description and its icon

Listbox (cont)

- Instance Methods ->
 - `current()` -> returns index of currently selected item
 - `set_list(list[,current])` -> sets items as list, current set currently selected item (index)
 - `bind(event_code,callback)` -> callback is called when every event in event_code triggers (we will see later)

Form class

- Form implements a dynamically configurable, editable multi-field dialog
- Class `appuifw.Form(fields[, flags=0])` →
- Fields → list of (label, type[, value]) displaying entry for label of type with value (default)
 - type → 'text', 'number', 'date', 'time', etc
- Instance attributes →
 - menu → takes a list of (name, callback) for menu
 - save_hook → point to callable which is called when ever form is saved, if save_hook returns True the form is saved, else if it returns False form is reset to original values

Form (cont)

- Instance Methods ->
 - `execute()` -> makes the form visible
 - `pop()` -> returns the last field descriptor
 - `insert(index, field_descriptor)` -> inserts `field_descriptor` at `index`
 - `length()` -> returns the number of `field_descriptor`

InfoPopup

- Class `appuifw.InfoPopup()`
- Provides info on top of other windows
- Instance Methods →
 - `show(text, [(x_coord, y_coord), time_shown, time_before, alignment])` → shows text at pos → (x, y) , for `time_shown` after `time_before`
 - `hide()` → hides the popup immediately
 - `alignment` → can be alignment form `appuifw` like
 - `EHLeftVCenter`
 - `EHCenterVTop`
 - `EHCenterVBottom`

Question:-

Try to create a simple TODO application, which allows the user to specify a title and description of the Task (Form) and Displays them in application body, it should allow user to mark TODO's as done and move them to a done list, user should be able to switch between TODO and Done list using tabs, also give notification of saving of Task using InfoPopup

Time: 15 min

globalui

globalui

- Allows query to be made by an app without an active UI
- 4 functions
 - `global_note(note_text[,type])`
 - `global_query(query_text[,timeout])`
 - `global_msg_query(query_text,header_text[,timeout])`
 - `global_popup_menu(option_items[,header_text,timeout])`
- Useful for background applications

globalui (cont)

- `global_note(note_text[,type])` → displays the note globally, same as `appuifw.note`
- `global_query(query_text[,timeout])` → displays a confirmation query, if user presses yes it returns 1 else 0, returns None on user cancel
- `global_msg_query(query_text,header_text[,timeout])`
→ same as `global_query`, `header_text` sets the header string of the query
- `global_popup_menu(option_items[,header_text,timeout])` → displays a global menu, `header_text` sets the heading string of menu, returns the index of the selected item

keycapture

- Allows capturing of keys globally
- Useful for assigning short codes to specific functions in application
- Class → `keycapture.KeyCapturer(<call back>)`
- `<call back>` → function with 1 argument
- Instance properties →
 - `keys` → list of keys to be captured
 - Key codes in `keycapture.key_codes`
- Instance Methods →
 - `start()` – starts key capture
 - `stop()` – stops key capture
 - `last_key()` – returns the last key captured

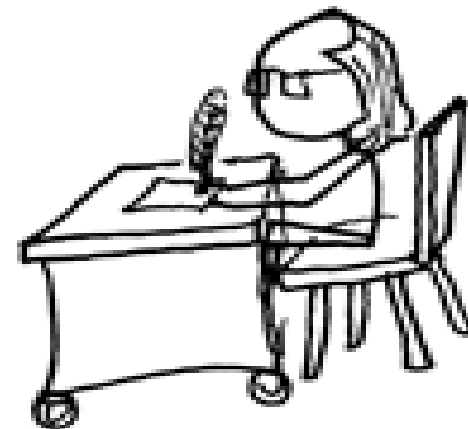
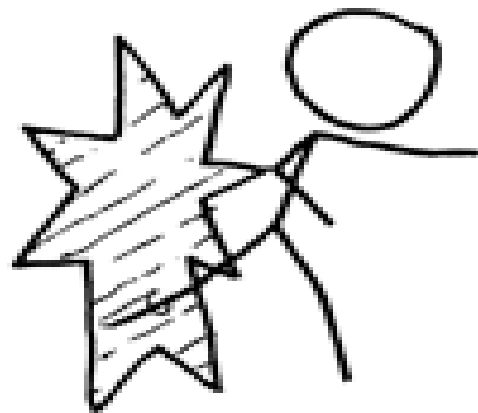
BENJAMIN FRANKLIN?

I BRING A MESSAGE
FROM THE FUTURE!
I DON'T HAVE MUCH TIME.

YES?

WHAT IS IT?

THE CONVENTION YOU'RE SETTING
FOR ELECTRIC CHARGE IS BACKWARD.
THE ONE LEFT ON GLASS BY SILK
SHOULD BE THE *NEGATIVE* CHARGE.



WE WERE GOING TO USE THE TIME MACHINE TO
PREVENT THE ROBOT APOCALYPSE, BUT THE
GUY WHO BUILT IT WAS AN ELECTRICAL ENGINEER.

Lets move to
services

telephone

- Provides access to phone's telephone features
- `dial(number)` → dials the number (string)
- `hang_up()` → hang up the call if call is active, else raises an error
- `answer()` → answers the call,
- function `incoming_call()` should be called before `answer()` to enable call handling
- Wont work on emulator!!

messaging

- Give access to phone's message functions
- Function \rightarrow `sms_send(number, msg, [encoding='7bit', callback=None, name=""])`
- `number` \rightarrow phone number as string
- `msg` \rightarrow message to be sent
- `encoding` \rightarrow encoding for the message
- `callback` \rightarrow function to be called with 1 argument equal to the state
- `state` \rightarrow `messaging.ESent`, `messaging.ESendFailed`, etc..
- `name` \rightarrow name to be displayed in sent messages

inbox

- Provides access to phone's inbox
- Class → `inbox.Inbox([type])`
- Type →
 - `inbox.EInbox` → default inbox
 - `inbox.EOutbox` → out box of phone
 - `inbox.ESent` → sent messages
 - `inbox.EDraft` → draft messages

inbox (cont)

- Instance methods →
 - `sms_messages()` → gives a list of sms id's
 - `content(sms_id)` → gives content of sms_id
 - `time(sms_id)` → gives time of sms in epoch
 - `address(sms_id)` → gives address of sender in unicode
 - `delete(sms_id)` → deletes the message
 - `unread(sms_id)` → returns 0 (read) or 1 (unread)
 - `set_unread(sms_id, status)` → status 0 or 1
 - `bind(callback)` → callback is called when a new message arrives with 1 parameter (sms_id)

location

- Currently only one function → `gsm_location`
- `location.gsm_location()` → returns a list of Mobile Country code, Mobile network code, Location area code and cell id
- Mobile Country code for india → 404
- Can be used to find a location (cell)
- `gsm_location` is very limited and can not provide exact location, its always better to use GPS

Time for
Data Management

contacts

- Class ContactDB → object creation using `contacts.open([filename[,mode]])`
- Instance methods →
 - `add_contact()` → returns a contact object (locked !), representing the new contact
 - `find(term)` → searches for term in DB and returns a list of all matching objects
 - `keys()` → returns ID's of all Contacts in DB
 - `import_vcards(vcards)` → imports contacts in vcards (string) into the database
 - `export_vcards(ids)` → returns a string containing vcards of given ids,
ids → tuple of ids

Contact

- Object Attributes →

- Id → a unique id given to the contact
- title → title of the contact
- last_modified → last modified date/time
- is_group → 1 if group, 0 otherwise

- Object Methods →

- begin() → lock the contact, preventing other apps from modifying the contact, raise ContactBusy if lock held by other application
- Commit() → commits (stores) the contact and releases the lock
- Rollback() → discards all the changes that were made
- as_vcard() → returns the contact as a vcard string

Contact (cont)

- Object Methods ->
 - `add_field(type,value) -> (unicode)`
 - Adds a new field into the contact
 - Type can be
 - City
 - Country
 - first_name
 - last_name
 - phone_number
 - mobile_number
 - For whole list see docs 5.1.3
 - `Find()` -> returns list of all the fields present

Question:-

Make an application which allows any person to sms "number <name>" to a phone and the phone returns number of <name> as sms (return the first match)

Time: 10min

e32calendar

- Provides calendar services
- There are 5 types of entries
 - AppointmentEntry
 - EventEntry
 - AnniversaryEntry
 - ReminderEntry
 - TodoEntry
- Class CalanderDb → object creation using `e32calendar.open()`

CalendarDb

- Instance Method ->
 - add_appointment() -> returns an AppointmentEntry
 - add_event() -> returns an EventEntry
 - add_anniversary() -> returns an AnniversaryEntry
 - add_todo() -> returns an TodoEntry
 - add_reminder() -> returns an ReminderEntry
- All entries are object of Entry, only TodoEntry has one extra attribute
 - cross_out_time -> which stores when it was crossed out

Entry

- Instance Attributes →
 - content → contains content of the entry
 - start_time → the start datetime value
 - end_time → the end datetime value
 - last_modified → last modified datetime value
 - location → entry's location data
- Instance Methods →
 - commit() → saves the changes
 - rollback() → restores the changes to the previous commit
 - set_time(start[,end]) → sets time for entry
 - as_vcalendar() → returns entry as vcalendar string

logs

- Allows access to information provided in phone's log
- logs are stored only for a finite time (say 30 days)
- Functions →
 - calls() → returns a list of calls description (dict)
 - sms() → returns a list of sms description (dict)
 - data_logs() → returns a list of data calls description (dict)
 - emails() → returns a list of email description (dict)
 - More in documentation

And we are done

Thank You..

Questions and
Answers ...