



ID		Creation date	June 25, 2009
Platform	S60 3rd and 5th Editions	Tested on devices	E71, XpressMusic 5800
Category	Python	Subcategory	

Keywords (APIs, classes, methods, functions): search, SMS

Introduction

Tired of looking for valuable information lost in your hundreds of SMSs ? Looking for a fast and reliable search method ? Want a single application that searches in all your SMS boxes automatically ? Say goodbye to old methods of searching SMS messages and try SMSearch, a new application written in Python for S60 3rd and 5th editions !

SMSearch

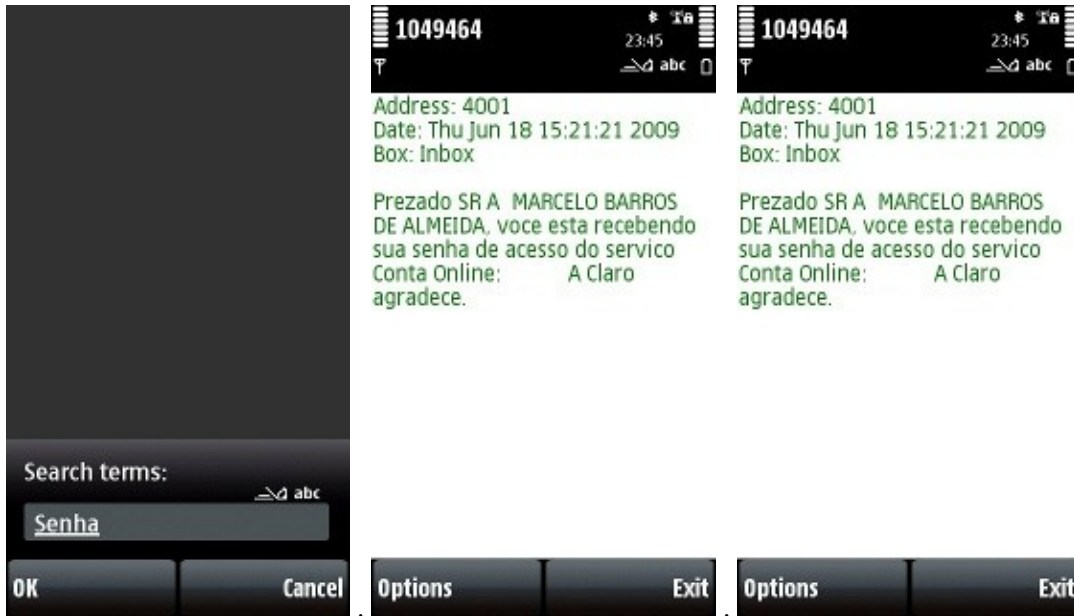
Just kidding, but the program is real. It is a small application for those people like me that are all the time looking for something in their SMSs. Yes, newer phones have already a builtin application for this task. In this case, you may use it as an example for inbox API when programming in Python.

The searching algorithm is based upon Boyer-Moore string searching, a famous and efficient string searching algorithm ([python code](#) borrowed from active state site, by [Nelson Rush](#))

The code (Python 1.9.5) and screenshots are below. You can download the [sis](#) or module [window.py](#) from the [program repository](#)s.

Thanks to [Javsmo](#) for drawing nice application icon !

Searching_all_your_SMSs_at_once_with_SMSearch



```
# -*- coding: utf-8 -*-
# Marcelo Barros de Almeida
# marcelobarrosalmeida@gmail.com
# License: GPL 3

from window import *
from appuifw import *
import inbox
import time

class ShowSMS(Dialog):
    def __init__(self,cbk,title,msg):
        Dialog.__init__(self,cbk,title,Text(msg))

class SMSearch(Application):
    def __init__(self):
        self.dlg = None
        self.results = [u""]
        self.terms = u""
        body = Canvas()
        menu = [(u"Search", self.get_pattern),
                (u>About", self.about),
                (u"Exit", self.close_app)]
        Application.__init__(self,u"SMSearch",body,menu)

    def about(self):
        note( u"SMSearch.\nMarcelo Barros de Almeida\nmarcelobarrosalmeida@gmail.com", "info" )

    def lst_cbk(self):
        idx = self.body.current()
        (sms_id,txt,tmr,addr,fn) = self.results[idx]
        msg = u"Address: " + addr + \
            u"\nDate: " + tmr + \
            u"\nBox: " + fn + \
            u"\n\n" + txt
        self.dlg = ShowSMS(lambda:self.refresh(),unicode(sms_id),msg)
        self.dlg.run()

    def bmh_search(self, pattern, text):
        # http://code.activestate.com/recipes/117223/
```

Searching_all_your_SMSs_at_once_with_SMSearch

```
m = len(pattern)
n = len(text)
if m > n: return -1
skip = []
for k in range(256): skip.append(m)
for k in range(m - 1): skip[ord(pattern[k])] = m - k - 1
skip = tuple(skip)
k = m - 1
while k < n:
    j = m - 1; i = k
    while j >= 0 and text[i] == pattern[j]:
        j -= 1; i -= 1
    if j == -1: return i + 1
    k += skip[ord(text[k])]
return -1

def search(self, pattern):
    self.results = []
    lst = []
    folders = {inbox.EInbox:u"Inbox",
               inbox.EOutbox:u"Outbox",
               inbox.ESent:u"Sent",
               inbox.EDraft:u"Draft"}
    for f,fn in folders.iteritems():
        ibx = inbox.Inbox(f)
        for sms_id in ibx.sms_messages():
            txt = ibx.content(sms_id)
            txt_utf8 = txt.encode('utf-8')
            pattern_utf8 = pattern.encode('utf-8')
            if self.bmh_search(pattern_utf8.lower(),txt_utf8.lower()) > -1:
                tm = ibx.time(sms_id)
                dt = unicode(time.ctime(tm), 'utf-8', errors='ignore')
                self.results.append((sms_id,txt,dt,ibx.address(sms_id),fn))
                lst.append((tm,dt,txt[:50]))
    if self.results:
        # order following unix time and remove it after
        lst.sort(reverse=True)
        lst = [x[1:] for x in lst]
        self.body = Listbox(lst,self.lst_cbk)
        app.screen = 'normal' # avoid wrong screen redraw
        self.refresh()
    else:
        note(u"No results for " + self.terms,"info")

def get_pattern(self):
    pattern = query(u"Search terms:", "text", self.terms)
    if pattern is not None:
        if pattern:
            pattern = pattern.strip()
            self.terms = pattern
            self.search(pattern)

if __name__ == "__main__":
    sms = SMSearch()
    sms.run()
```