Writing win32 applications with python and glade

Author: Nzeka Gilbert
Author e-mail: khaalel@gmail.com
Author website: www.nzeka-labs.com, http://korriban-planet.blogspot.com/
Document Version: 1.0

Index

I] License

This tutorial is covered by the FDL.
The program listed here is covered by the GPL.

II] What are we going to do?

The goal of this article is to build a podcasting tool using Pygtk (Python + Glade + Pygtk). This tool, named BitPodder, is a podcasting tool allowing getting .torrent files from RSS Feeds. By explaining each step of the making of this tool, I hope I will help people having some difficulties with Python and Glade under win32 platform.

III] What you should have before starting...

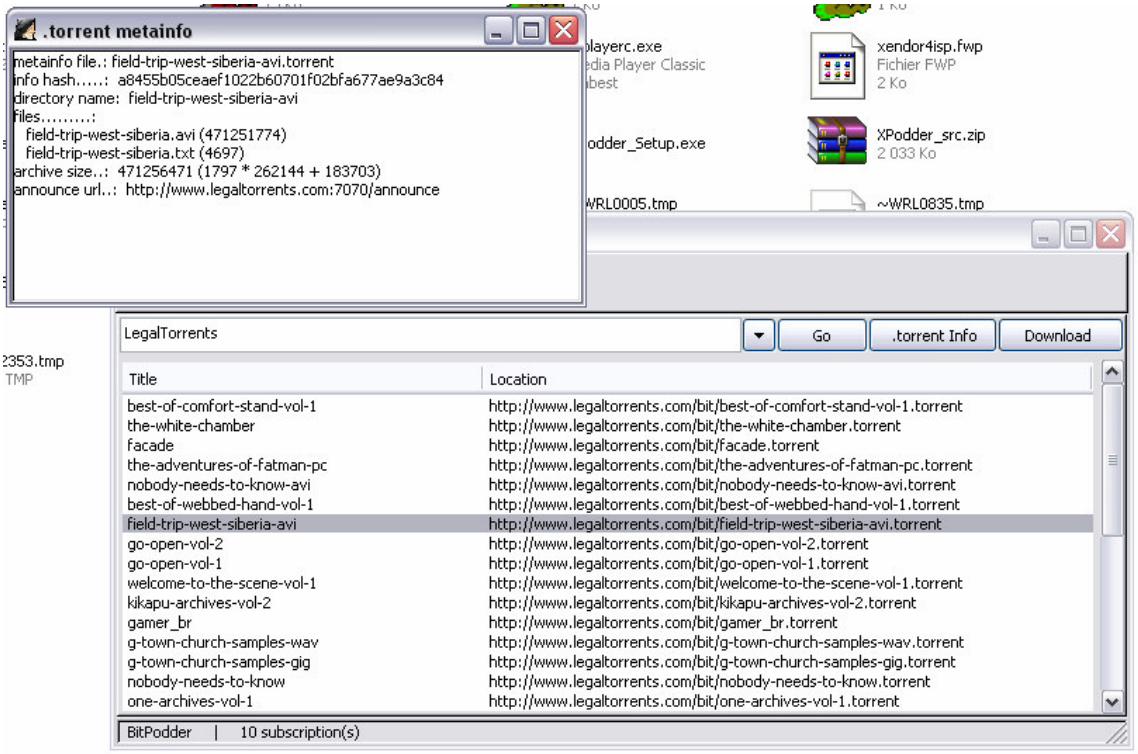
- python
- Gtk+/Win32 Development Environment (http://gladewin32.sourceforge.net/)
- Gtk+/Win32 Runtime Environment (http://gladewin32.sourceforge.net/)
- Pygtk and gtkmm for Win32 (http://www.pcpm.ucl.ac.be/~gustin/win32_ports/)
- Tepache (http://primates.ximian.com/~sandino/python-glade/)
- You can get BitPodder_src.zip the tarball that contains the source code of BitPodder at: http://sourceforge.net/projects/korriban

IV] What is PyGTK?

"PyGTK provides **a convenient wrapper** for the GTK+ library for use in Python programs, taking care of many of the boring details such as managing memory and type casting. When combined with PyORBit and gnome-python, it can be used to write full featured Gnome applications." (http://www.pygtk.org/about.html)

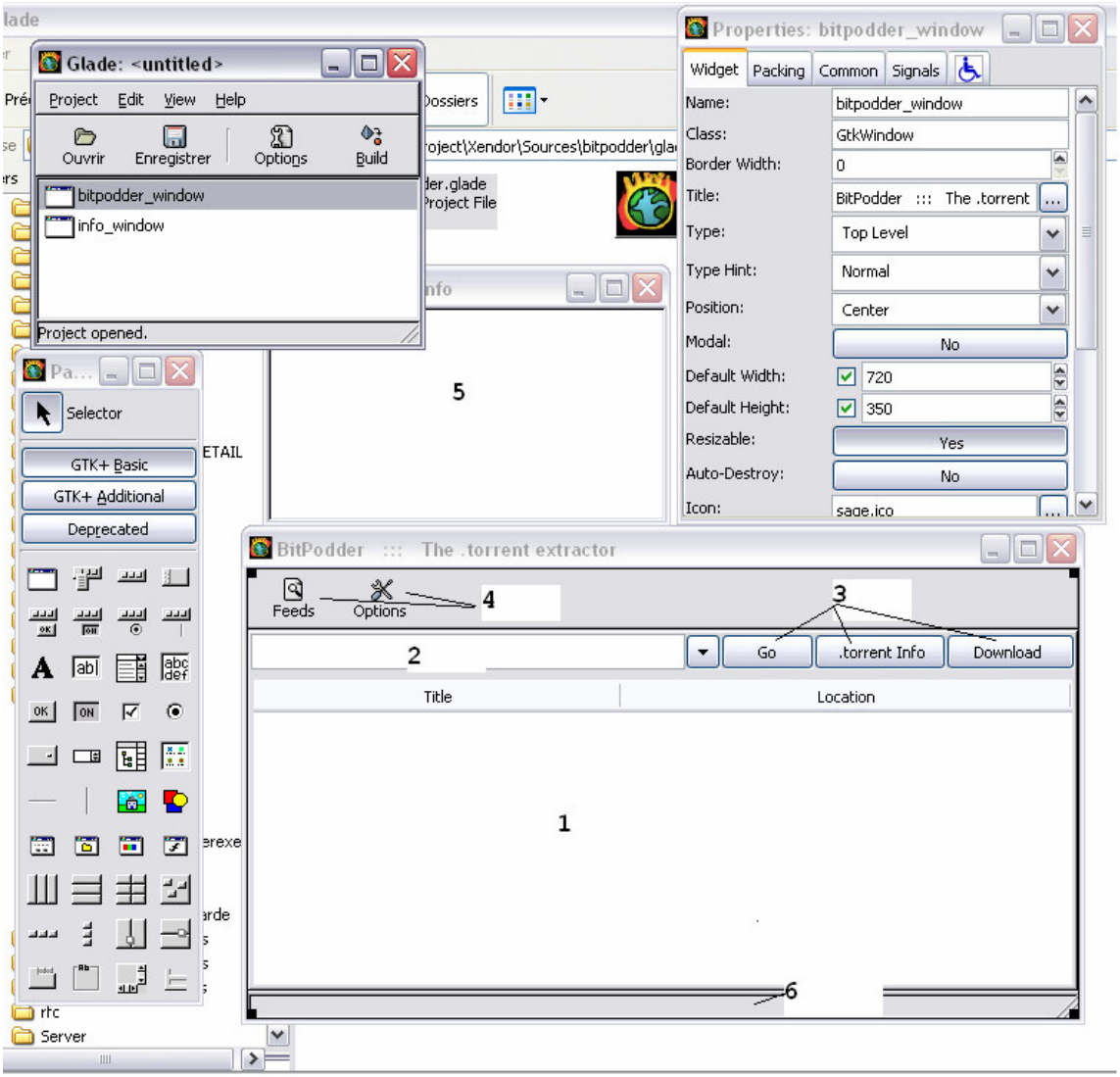
V] Let us start

This is the win32 app we will code in this article. The screenshot is not clear but the goal is to show you what looks like BitPodder.

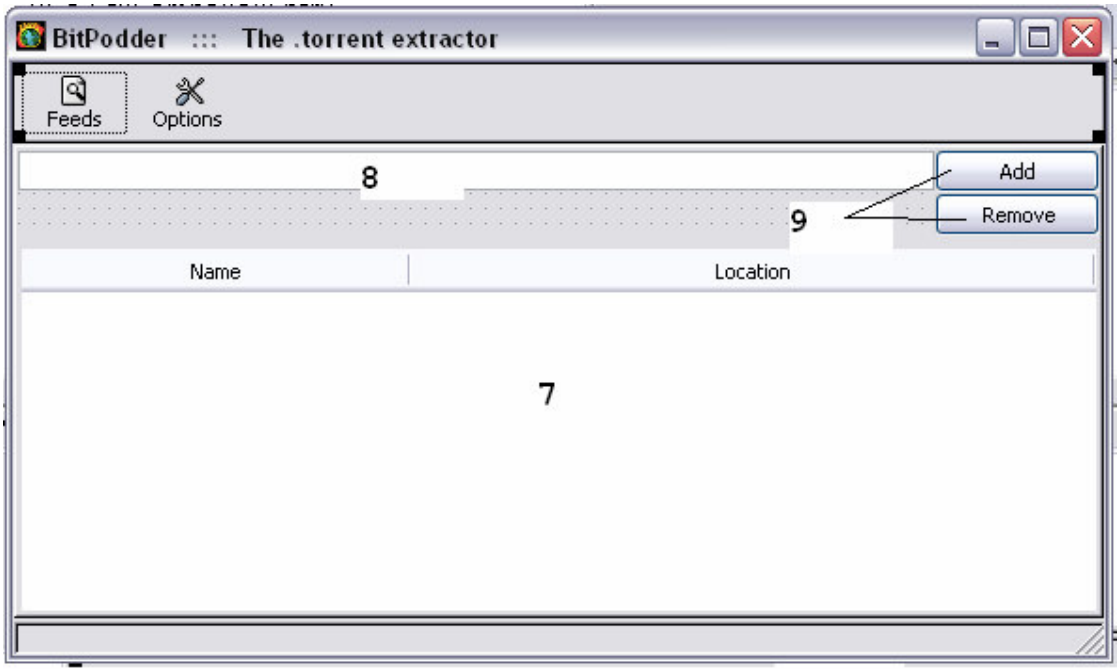


VI] Building the windows

First, we have to create the windows BitPodder will use. Thanks to glade, we can do that in a visual way. BitPodder contains a lot of widgets so before continuing this article, I hope you already have used glade or you know some widgets and know how works signals under GTK+. If all your answers are: no, I advise you to read articles like “Writing PyGTK applications in a visual way” (http://primates.ximian.com/~sandino/python-glade/) to know how work glade.

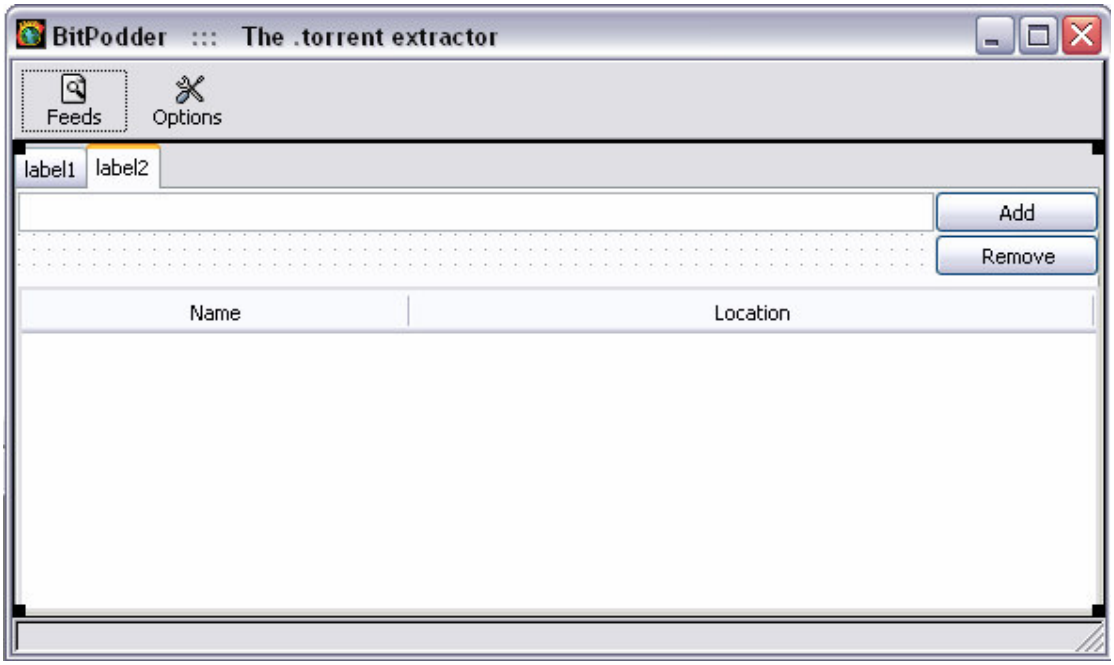


1: a GtkCList
2: a GtkComboBoxEntry
3: some GtkButton
4: some GtkToolButton
5: a GtkTextView
6: a GtkStatusbar

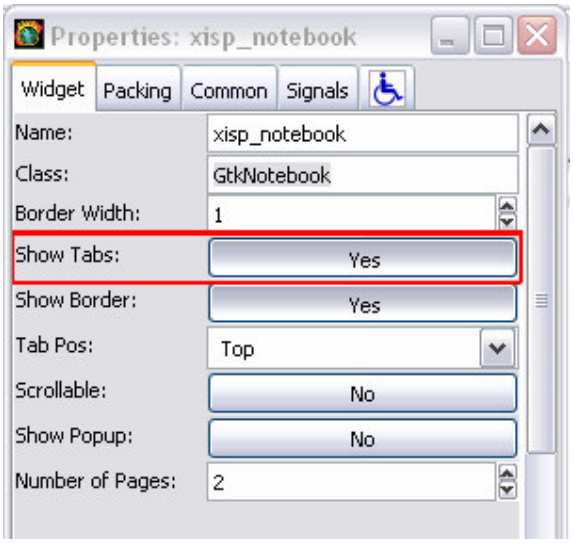


Now the widgets of the Option page:
7: a GtkCList
8: a GtkEntry
9: some GtkButton

I said "page" because to switch between Feeds and Options page, I used a GtkNotebook then I hided the tabs of this NoteBook. Here is the real window without hiding the tabs of the notebook:



To hide the tabs, you have to click here:



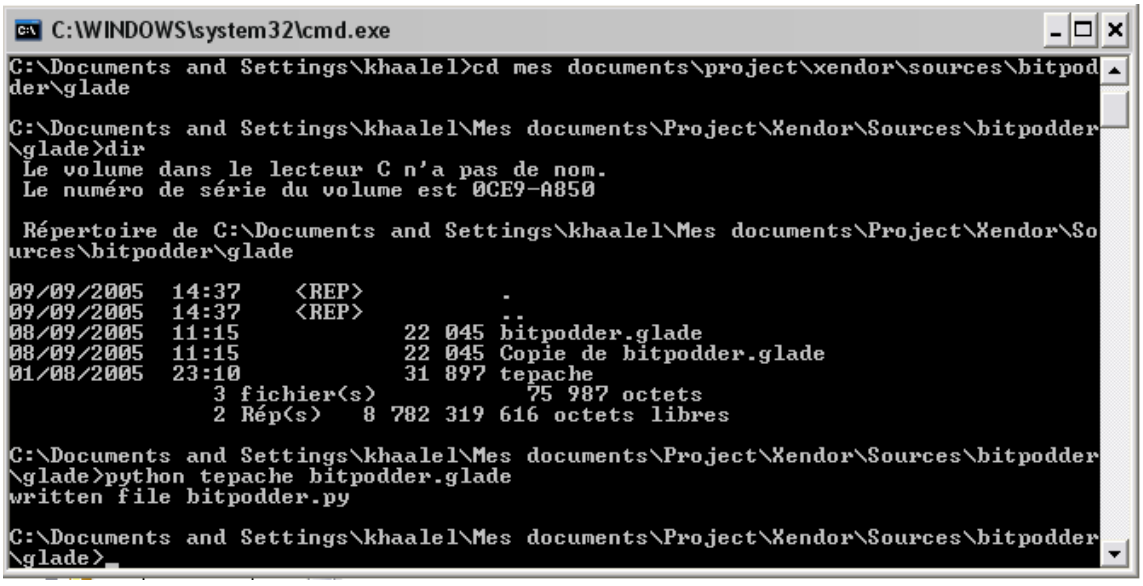
I won't continue to explain how I created the windows of BitPodder project, because this article would be so long and you can read the glade file to see how it was made.

VII] How to code this app in python?

We will use a wonderful tool named tepache (http://primates.ximian.com/~sandino/python-glade/tepache/). With this tool, it will be possible to generate a python file with clean classes allowing controlling the widgets of your glade app.

Before, using tepache, download the tarball, unzip it and then install tepache by doing "python setup.py install" (thanks to cmd.exe) in the directory where you unzipped tepache.

Open another cmd.exe window and go where your glade file is. Put tepache in this directory then enter the following command in cmd.exe: **python tepache xxx.glade**. Where xxx.glade is the name of your glade file: for me it's **bitpodder.glade**.



If all work fine, you should have 3 new files in the current directory: bitpodder.py, bitpodder.py.orig and SimpleGladeApp.py.

bitpodder.py is the python file in which we will write the python code of the soft. bitpodder.py.orig must not be modify or delete because if you want to update bitpodder.py, after having modifying the glade file, tepache will use this file in order to not overwrite the code you wrote.
SimpleGladeApp.py contain the python code that your app will execute when the users will do default actions like closing the window of your app.

In a next article, I will explain you how to write your own tepache or your own SimpleGladeApp.py file to customize your app: for example, sometimes, when users close the window of the app, it would be nice to write a file that would contain the last positions of the windows, or information about the user…

Now, we will open bitpodder.py to add the code of the app.

Bitpodder.py should looks like this:

```
#!/usr/bin/env python
# -*- coding: UTF8 -*-

# Python module bitpodder.py
# Autogenerated from bitpodder.glade
# Generated on Fri Sep 09 14:38:31 2005

# Warning: Do not modify any context comment such as #--
# They are required to keep user's code

import os

import gtk

from SimpleGladeApp import SimpleGladeApp
from SimpleGladeApp import bindtextdomain

app_name = "bitpodder"
app_version = "0.0.1"

glade_dir = ""
locale_dir = ""

bindtextdomain(app_name, locale_dir)


class BitpodderWindow(SimpleGladeApp):

    def __init__(self, path="bitpodder.glade",
                 root="bitpodder_window",
                 domain=app_name, **kwargs):
        path = os.path.join(glade_dir, path)
        SimpleGladeApp.__init__(self, path, root, domain, **kwargs)

    #-- BitpodderWindow.new {
    def new(self):
        print "A new %s has been created" % self.__class__.__name__
    #-- BitpodderWindow.new }

    #-- BitpodderWindow custom methods {
```

```
    #   Write your own methods here
    #-- BitpodderWindow custom methods }

    #-- BitpodderWindow.on_feed_toolbutton_clicked {
    def on_feed_toolbutton_clicked(self, widget, *args):
        print "on_feed_toolbutton_clicked called with self.%s" % widget.get_name()
    #-- BitpodderWindow.on_feed_toolbutton_clicked }

    #-- BitpodderWindow.on_options_toolbutton_clicked {
    def on_options_toolbutton_clicked(self, widget, *args):
        print "on_options_toolbutton_clicked called with self.%s" % widget.get_name()
    #-- BitpodderWindow.on_options_toolbutton_clicked }

    #-- BitpodderWindow.on_xisp_go_button_clicked {
    def on_xisp_go_button_clicked(self, widget, *args):
        print "on_xisp_go_button_clicked called with self.%s" % widget.get_name()
    #-- BitpodderWindow.on_xisp_go_button_clicked }

    #-- BitpodderWindow.on_show_button_clicked {
    def on_show_button_clicked(self, widget, *args):
        print "on_show_button_clicked called with self.%s" % widget.get_name()
    #-- BitpodderWindow.on_show_button_clicked }

    #-- BitpodderWindow.on_xisp_play_button_clicked {
    def on_xisp_play_button_clicked(self, widget, *args):
        print "on_xisp_play_button_clicked called with self.%s" % widget.get_name()
    #-- BitpodderWindow.on_xisp_play_button_clicked }

    #-- BitpodderWindow.on_xisp_results_clist_select_row {
    def on_xisp_results_clist_select_row(self, widget, *args):
        print "on_xisp_results_clist_select_row called with self.%s" % widget.get_name()
    #-- BitpodderWindow.on_xisp_results_clist_select_row }

    #-- BitpodderWindow.on_xisp_add_button_clicked {
    def on_xisp_add_button_clicked(self, widget, *args):
        print "on_xisp_add_button_clicked called with self.%s" % widget.get_name()
    #-- BitpodderWindow.on_xisp_add_button_clicked }

    #-- BitpodderWindow.on_xisp_remove_button_clicked {
    def on_xisp_remove_button_clicked(self, widget, *args):
        print "on_xisp_remove_button_clicked called with self.%s" % widget.get_name()
    #-- BitpodderWindow.on_xisp_remove_button_clicked }

    #-- BitpodderWindow.on_xisp_option_clist_select_row {
    def on_xisp_option_clist_select_row(self, widget, *args):
        print "on_xisp_option_clist_select_row called with self.%s" % widget.get_name()
    #-- BitpodderWindow.on_xisp_option_clist_select_row }


class InfoWindow(SimpleGladeApp):

    def __init__(self, path="bitpodder.glade",
                 root="info_window",
                 domain=app_name, **kwargs):
        path = os.path.join(glade_dir, path)
        SimpleGladeApp.__init__(self, path, root, domain, **kwargs)

    #-- InfoWindow.new {
    def new(self):
        print "A new %s has been created" % self.__class__.__name__
    #-- InfoWindow.new }

    #-- InfoWindow custom methods {
```

```
    #   Write your own methods here
    #-- InfoWindow custom methods }


#-- main {

def main():
    bitpodder_window = BitpodderWindow()
    info_window = InfoWindow()

    bitpodder_window.run()

if __name__ == "__main__":
    main()

#-- main }
```

But we want the code looks like this (it's the complete code of BitPodder, don't worry I will explain each line after the listing):

```
#!/usr/bin/env python
# -*- coding: UTF8 -*-

# Python module xpodder.py
# Autogenerated from xpodder.glade
# Generated on Tue Sep 06 00:21:46 2005

# Warning: Do not modify any context comment such as #--
# They are required to keep user's code

import os, re, urllib, urlparse, feedparser, codecs
from sys import *
from os.path import *
from sha import *
from bencode import *
import gtk

from SimpleGladeApp import SimpleGladeApp
from SimpleGladeApp import bindtextdomain

app_name = "bitpodder"
app_version = "0.1"

glade_dir = ""
locale_dir = ""

bindtextdomain(app_name, locale_dir)
XISP_Results_Row_Variable = []
XISP_Options_Row_Variable = []
XISP_Link = ""

class BitPodderWindow(SimpleGladeApp):

    def __init__(self, path="glade\\bitpodder.glade",
                 root="bitpodder_window",
                 domain=app_name, **kwargs):
        path = os.path.join(glade_dir, path)
        SimpleGladeApp.__init__(self, path, root, domain, **kwargs)

    #-- Xendor4ispWindow.new {
    def new(self):
        counter = 0
        self.xisp_results_clist.clear()
```

```
        self.xisp_option_clist.clear()
        fdw = open("bitpodder.fwp", 'r').readlines()
        for ligne in fdw:
            lesplit = ligne.split(":?:?:")
            lien = lesplit[1].strip()
            rpln = self.xisp_option_clist.append([lesplit[0], lien])
            rpln = self.xisp_comboboxentry.append_text(lesplit[0])
            counter = counter + 1

                # http://www.pygtk.org/pygtktutorial/sec-statusbars.html
        global xisp_id, xispext_id
        stamess = "  BitPodder     |     " + str(counter) + " subscription(s)"
        xispext_id = self.xisp_statusbar.get_context_id("XISP Statusbar")
        xisp_id = self.xisp_statusbar.push(xispext_id, stamess)
    #-- Xendor4ispWindow.new }

    #-- Xendor4ispWindow custom methods {
    #   Write your own methods here
    #-- Xendor4ispWindow custom methods }

    #-- Xendor4ispWindow.on_feed_toolbutton_clicked {
    def on_feed_toolbutton_clicked(self, widget, *args):
        self.xisp_notebook.set_current_page(0)
    #-- Xendor4ispWindow.on_feed_toolbutton_clicked }

    #-- Xendor4ispWindow.on_options_toolbutton_clicked {
    def on_options_toolbutton_clicked(self, widget, *args):
        self.xisp_notebook.set_current_page(1)
    #-- Xendor4ispWindow.on_options_toolbutton_clicked }

    #-- Xendor4ispWindow.on_xisp_go_button_clicked {
    def on_xisp_go_button_clicked(self, widget, *args):
        self.xisp_results_clist.clear()
        xentry = self.xisp_comboboxentry.child
        xentry = self.xisp_comboboxentry.get_child()
        choosen = xentry.get_text()

        urrl = get_xisp_id(choosen)

        page = urllib.urlopen(urrl).read()
        page = page.replace( '&quot;', '"')
        page = page.replace( '&amp;', '&')
        page = page.replace( '&lt;', '<')
        page = page.replace( '&gt;', '>')
        page = page.replace( ' ', ' ')
#       t = re.compile('.*<title>[a-zA-Z0-9_].*</title>')
        p = re.compile('.*url=["\']([^"\']*)["\'].*torrent')
        files = p.findall(page)
#       titless = t.findall(page)

        cou = 0
        for fff in files:
#           na = titless[cou+1].strip("<title>")
#           na = na.strip("</title>")
#           try:
#               metainfo_file = urllib.urlopen(fff)
#               metainfo = bdecode(metainfo_file.read())
#               info = metainfo['info']
#               na = info['name']
#           except ValueError:
#                 na = os.path.splitext(os.path.split(fff)[1])[0]
#           if info.has_key('length'):
#              na = info['name']
```

```
#           else:
#               na = info['name']
#           na = os.path.splitext(os.path.split(fff)[1])[0]
          na = os.path.splitext(os.path.split(fff)[1])[0]
          scomplink = self.xisp_results_clist.append([na, fff])
          cou = cou + 1

    #-- Xendor4ispWindow.on_xisp_go_button_clicked }

    #-- Xendor4ispWindow.on_xisp_play_button_clicked {
    def on_xisp_play_button_clicked(self, widget, *args):
        if len(XISP_Results_Row_Variable) != 0:
          xrow = XISP_Results_Row_Variable[0]
          scomplink = self.xisp_results_clist.get_text(xrow, 1)
          title = os.path.split(scomplink)[1]

          sharedfolder = 0
          for x in os.listdir(os.getcwd()):
              if x == "Torrents Files": sharedfolder = 1

          fol = os.getcwd() + "\\" + "Torrents Files"
          if sharedfolder == 0:
            err = os.mkdir(fol)

          title = fol + "\\" + str(title)
          fdm = open(title, 'a')
          fdm.close()
          tor = urllib.urlretrieve(scomplink, title)
    #-- Xendor4ispWindow.on_xisp_play_button_clicked }

    #-- Xendor4ispWindow.on_xisp_results_clist_select_row {
    def on_xisp_results_clist_select_row(self, widget, *args):
        global XISP_Results_Row_Variable
        XISP_Results_Row_Variable = self.xisp_results_clist.selection
    #-- Xendor4ispWindow.on_xisp_results_clist_select_row }

    #-- Xendor4ispWindow.on_show_button_clicked {
    def on_show_button_clicked(self, widget, *args):
        if len(XISP_Results_Row_Variable) != 0:
          xrow = XISP_Results_Row_Variable[0]
          scomplink = self.xisp_results_clist.get_text(xrow, 1)
          global XISP_Link
          XISP_Link = scomplink
          info_window = InfoWindow()
    #-- Xendor4ispWindow.on_show_button_clicked }

    #-- Xendor4ispWindow.on_xisp_add_button_clicked {
    def on_xisp_add_button_clicked(self, widget, *args):
        xtring = self.xisp_add_entry.get_text()
        if xtring != "":
          lurl = xtring
          lxml = feedparser.parse(lurl.strip())
          title = lxml.channel.title
          fdw = open("bitpodder.fwp", 'a')
          if xtring != "":
            ch = title + ":?:?:" + xtring
            fdw.write(ch)
            fdw.write('\n')
          fdw.close()

          xtring = self.xisp_add_entry.set_text("")
          self.xisp_option_clist.clear()
```

```
        bitpodder_window = BitPodderWindow()
        self.bitpodder_window.destroy()

    #-- Xendor4ispWindow.on_xisp_add_button_clicked }

    #-- Xendor4ispWindow.on_xisp_remove_button_clicked {
    def on_xisp_remove_button_clicked(self, widget, *args):
        if len(XISP_Options_Row_Variable)!= 0:
            xtrow = XISP_Options_Row_Variable[0]
            if xtrow != "":
                self.xisp_option_clist.remove(xtrow)

            fdw = open("bitpodder.fwp", 'w')
            for drow in range(self.xisp_option_clist.rows):
                ame = self.xisp_option_clist.get_text(drow, 0)
                d = self.xisp_option_clist.get_text(drow, 1)
                ch = ame + ":?:?:" + d
                fdw.write(ch)
                fdw.write('\n')
            fdw.close()

            bitpodder_window = BitPodderWindow()
            self.bitpodder_window.destroy()

    #-- Xendor4ispWindow.on_xisp_remove_button_clicked }

    #-- Xendor4ispWindow.on_xisp_option_clist_select_row {
    def on_xisp_option_clist_select_row(self, widget, *args):
        global XISP_Options_Row_Variable
        XISP_Options_Row_Variable = self.xisp_option_clist.selection
    #-- Xendor4ispWindow.on_xisp_option_clist_select_row }



class InfoWindow(SimpleGladeApp):

    def __init__(self, path="glade\\bitpodder.glade",
                 root="info_window",
                 domain=app_name, **kwargs):
        path = os.path.join(glade_dir, path)
        SimpleGladeApp.__init__(self, path, root, domain, **kwargs)

    #-- InfoWindow.new {
    def new(self):
        global XISP_Link
        link = XISP_Link

        metainfo_file = urllib.urlopen(link)
        metainfo = bdecode(metainfo_file.read())

        info = metainfo['info']
        info_hash = sha(bencode(info))
        txt = "metainfo file.: %s" % basename(link) + "\n"
        txt = txt + "info hash.....:  %s" % info_hash.hexdigest() + "\n"
        piece_length = info['piece length']
        if info.has_key('length'):
            # let's assume we just have a file
            txt = txt + "file name.....:  %s" % info['name'] + "\n"
            file_length = info['length']
            name ='file size.....:  '
        else:
            # let's assume we have a directory structure
            txt = txt + "directory name:  %s" % info['name'] + "\n"
```

```
            txt = txt + "files.........:  " + "\n"
            file_length = 0
            for file in info['files']:
                path = ''
                for item in file['path']:
                    if (path != ''):
                        path = path + "/"
                    path = path + item
                txt = txt + "   %s (%d)" % (path, file['length']) + "\n"
                file_length += file['length']
                name ='archive size..: '
        piece_number, last_piece_length = divmod(file_length, piece_length)
        txt = txt + "%s %i (%i * %i + %i)" \
              % (name,file_length, piece_number, piece_length, last_piece_length) + "\n"
        txt = txt + "announce url..:  %s" % metainfo['announce'] + "\n"
        if metainfo.has_key('announce-list'):
            list = []
            for tier in metainfo['announce-list']:
                for tracker in tier:
                    list+=[tracker,',']
                del list[-1]
                list+=['|']
            del list[-1]
            liststring = ''
            for i in list:
                liststring+=i
            txt = txt + "announce-list.:  %s" % liststring + "\n"
        if metainfo.has_key('httpseeds'):
            list = []
            for seed in metainfo['httpseeds']:
                list += [seed,'|']
            del list[-1]
            liststring = ''
            for i in list:
                liststring+=i
            txt = txt + "http seeds....:  %s" % liststring + "\n"
        if metainfo.has_key('comment'):
            txt = txt + "comment.......:  %s" % metainfo['comment'] + "\n"


        buf = self.metainfo_textview.get_buffer()
        buf.set_text(txt)
    #-- InfoWindow.new }

    #-- InfoWindow custom methods {
    #   Write your own methods here
    #-- InfoWindow custom methods }



def get_xisp_id(identifiant):
    fdpn = open("bitpodder.fwp").readlines()
    reqxisp = re.compile(identifiant)
    for pe in fdpn:
        if reqxisp.search(pe):
            PeersString = pe.split(":?:?:")
    return PeersString[1]
#-- main {

def main():
    bitpodder_window = BitPodderWindow()

    bitpodder_window.run()
```

```
if __name__ == "__main__":
    main()

#-- main }
```

Like you can see, BitPodder is a small soft of only 300 lines: thanks to python!!!
Ok, ok I promise I will not act like a python prophet lol!!!

Let's start analyzing bitpodder.py (beta) line by line.

Before anything, we have to include some modules:

```
import os, re, urllib, urlparse, feedparser, codecs
from sys import *
from os.path import *
from sha import *
from bencode import *
import gtk
```

Feedparser can be found at feedparser.org, and bencode can be found in Bit Torrent package. But to make your life easy, you can found these modules in the tarball of BitPodder_src.

Then, we have the code generated by tepache:

```
from SimpleGladeApp import SimpleGladeApp
from SimpleGladeApp import bindtextdomain

app_name = "bitpodder"
app_version = "0.1"

glade_dir = ""
locale_dir = ""

bindtextdomain(app_name, locale_dir)
```

Now, there are 3 global variables we will use in the app:

```
XISP_Results_Row_Variable = []
XISP_Options_Row_Variable = []
XISP_Link = ""
```

Here is the entry of the main window class I named in the glade file: bitpodder_window. Like you can see: I modify a line: "glade\\bitpodder.glade". Why? Because I decided to put the glade file in another directory so I created a new directory named "\glade\". The line contains 2 backslash because \b have a meaning in python and we want to annul the backslash coming before the "b".

```
class BitPodderWindow(SimpleGladeApp):

    def __init__(self, path="glade\\bitpodder.glade",
                 root="bitpodder_window",
                 domain=app_name, **kwargs):
        path = os.path.join(glade_dir, path)
        SimpleGladeApp.__init__(self, path, root, domain, **kwargs)

    #-- Xendor4ispWindow.new {
```

Now we will write the new() method. What is this method? The code inside this method will be execute before the window appear, thanks to that we will initializate some widgets before they appear.

Let's start by clearing the 2 CLists we use in BitPodder.

```
    def new(self):
        counter = 0
        self.xisp_results_clist.clear()
        self.xisp_option_clist.clear()
```

Then we will open the file that contains the RSS Feeds links. A line of this file looks like this: "BitTorrent @ AnimeSuki.com:?:?:http://www.animesuki.com/rss.php".

Explication: first, there is the name of the feed, then there is this ":?:?:", then there is the feed url.

So we open the file and at each line, we will split the line thanks to this: ":?:?:".
Then we add the name and the url of each RSS Feed in the CList of the Option page and only the name of each feed in the comboboxentry.

Warning!!! Glade have a bug… you have to add this code in the glade file where the comboboxentry is defined: "<property name="items" translatable="yes"></property>" otherwise you will a beautiful error when the app will be launch.
The code that defines the comboboxentry should look like this:

```
                    <child>
                      <widget class="GtkComboBoxEntry" id="xisp_comboboxentry">
                        <property name="visible">True</property>
                        <property name="items" translatable="yes"></property>
                        <property name="add_tearoffs">False</property>
                        <property name="has_frame">True</property>
                        <property name="focus_on_click">True</property>
                      </widget>
                      <packing>
                        <property name="padding">0</property>
                        <property name="expand">True</property>
                        <property name="fill">True</property>
                      </packing>
                    </child>
```

```
        fdw = open("bitpodder.fwp", 'r').readlines()
        for ligne in fdw:
            lesplit = ligne.split(":?:?:")
            lien = lesplit[1].strip()
            rpln = self.xisp_option_clist.append([lesplit[0], lien])
            rpln = self.xisp_comboboxentry.append_text(lesplit[0])
            counter = counter + 1
```

Now we can add the string in the status bar.

```
                # http://www.pygtk.org/pygtktutorial/sec-statusbars.html
        global xisp_id, xispext_id
        stamess = "  BitPodder     |     " + str(counter) + " subscription(s)"
        xispext_id = self.xisp_statusbar.get_context_id("XISP Statusbar")
        xisp_id = self.xisp_statusbar.push(xispext_id, stamess)
    #-- Xendor4ispWindow.new }
```

```
    #-- Xendor4ispWindow custom methods {
    #   Write your own methods here
    #-- Xendor4ispWindow custom methods }
```

Now we will tell the app what to do when users click on Feeds and Options buttons.
Like I said, I used a notebook, so I just have to open the right page: each page is identified by a number.

```
#-- Xendor4ispWindow.on_feed_toolbutton_clicked {
    def on_feed_toolbutton_clicked(self, widget, *args):
        self.xisp_notebook.set_current_page(0)
    #-- Xendor4ispWindow.on_feed_toolbutton_clicked }

    #-- Xendor4ispWindow.on_options_toolbutton_clicked {
    def on_options_toolbutton_clicked(self, widget, *args):
        self.xisp_notebook.set_current_page(1)
    #-- Xendor4ispWindow.on_options_toolbutton_clicked }
```

Now we will tell the app what to do when users click on Go button.
We will obtain the feed name the user chooses in the comboboxentry and put it in the variable *choosen*.

```
    #-- Xendor4ispWindow.on_xisp_go_button_clicked {
    def on_xisp_go_button_clicked(self, widget, *args):
        self.xisp_results_clist.clear()
        xentry = self.xisp_comboboxentry.child
        xentry = self.xisp_comboboxentry.get_child()
        choosen = xentry.get_text()
```

After having the name, we have to get the url associated with the name so I created a function named get_xisp_id() that I will define later. After having the url feed, I will read the web page, replace some strings and search all the string that look like this "url=http://blablabla.torrent" (you have to know the .torrent urls are put in the <enclosure> tag that look like that: "<enclosure url="http://www.anime-kraze.org/torrent/[Ani-Kraze]_Tsubasa_Chronicle_-_16_[2CE95BC4].avi.torrent" length="14008" type="application/x-bittorrent"/>"). When I get all the links BitPodder found, first I will parse the url to obtain the .torrent name thanks to os.path.splitext(xxx)[0], then I will add the results in the Feeds page's CList.

```
        urrl = get_xisp_id(choosen)

        page = urllib.urlopen(urrl).read()
        page = page.replace( '&quot;', '"')
        page = page.replace( '&amp;', '&')
        page = page.replace( '&lt;', '<')
        page = page.replace( '&gt;', '>')
        page = page.replace( ' ', ' ')
#       t = re.compile('.*<title>[a-zA-Z0-9_].*</title>')
        p = re.compile('.*url=["\'](.*)["\'].*torrent')
        files = p.findall(page)
#       titless = t.findall(page)

        cou = 0
        for fff in files:
#           na = titless[cou+1].strip("<title>")
#           na = na.strip("</title>")
#           try:
#               metainfo_file = urllib.urlopen(fff)
#               metainfo = bdecode(metainfo_file.read())
#               info = metainfo['info']
#               na = info['name']
```

```
#           except ValueError:
#                 na = os.path.splitext(os.path.split(fff)[1])[0]
#           if info.has_key('length'):
#              na = info['name']
#           else:
#               na = info['name']
#           na = os.path.splitext(os.path.split(fff)[1])[0]
          na = os.path.splitext(os.path.split(fff)[1])[0]
          scomplink = self.xisp_results_clist.append([na, fff])
          cou = cou + 1

    #-- Xendor4ispWindow.on_xisp_go_button_clicked }
```

Now we will tell the app what to do when users select a line in the CList.
When a row (line) is selected, GTK+ send the number of the line selected in a list. So we will use a global variable to be able to use this number with different functions.

```
    #-- Xendor4ispWindow.on_xisp_results_clist_select_row {
    def on_xisp_results_clist_select_row(self, widget, *args):
        global XISP_Results_Row_Variable
        XISP_Results_Row_Variable = self.xisp_results_clist.selection
    #-- Xendor4ispWindow.on_xisp_results_clist_select_row }
```

Now we will tell the app what to do when users click on Download button.
First we will check if the global variable we associated to the CList has a value, if not we do nothing. If yes, we take the number of the line (row) and take the value of the second column of the line (the torrent url) then we download the .torrent file in Torrent Files directory (that we create if it's the first time the user download a torrent)

```
    #-- Xendor4ispWindow.on_xisp_play_button_clicked {
    def on_xisp_play_button_clicked(self, widget, *args):
        if len(XISP_Results_Row_Variable) != 0:
            xrow = XISP_Results_Row_Variable[0]
            scomplink = self.xisp_results_clist.get_text(xrow, 1)
            title = os.path.split(scomplink)[1]

            sharedfolder = 0
            for x in os.listdir(os.getcwd()):
                if x == "Torrents Files": sharedfolder = 1

            fol = os.getcwd() + "\\" + "Torrents Files"
            if sharedfolder == 0:
                err = os.mkdir(fol)

            title = fol + "\\" + str(title)
            fdm = open(title, 'a')
            fdm.close()
            tor = urllib.urlretrieve(scomplink, title)
    #-- Xendor4ispWindow.on_xisp_play_button_clicked }
```

Now we will tell the app what to do when users click on torrent Info button.
Like for Download button we try to get the number of the row then the torrent url that we will put in a global variable. To finish, we will open the second window, in which the user will be able to see information about the torrent.

```
    #-- Xendor4ispWindow.on_show_button_clicked {
    def on_show_button_clicked(self, widget, *args):
        if len(XISP_Results_Row_Variable) != 0:
            xrow = XISP_Results_Row_Variable[0]
            scomplink = self.xisp_results_clist.get_text(xrow, 1)
```

```
        global XISP_Link
        XISP_Link = scomplink
        info_window = InfoWindow()
    #-- Xendor4ispWindow.on_show_button_clicked }
```

Now we will write the code for the Option page.
Now we will tell the app what to do when users click on Add button.
To resume the code, when the user will enter the link of an RSS Feed, we will open this feed to obtain the name of the channel and then we add this new feed to bitpodder.fwp file. To finish, we will reload the app by opening a new window and destroying the current window.

```
    #-- Xendor4ispWindow.on_xisp_add_button_clicked {
    def on_xisp_add_button_clicked(self, widget, *args):
        xtring = self.xisp_add_entry.get_text()
        if xtring != "":
            lurl = xtring
            lxml = feedparser.parse(lurl.strip())
            title = lxml.channel.title
            fdw = open("bitpodder.fwp", 'a')
            if xtring != "":
                ch = title + ":?:?:" + xtring
                fdw.write(ch)
                fdw.write('\n')
            fdw.close()

            xtring = self.xisp_add_entry.set_text("")
            self.xisp_option_clist.clear()

            bitpodder_window = BitPodderWindow()
            self.bitpodder_window.destroy()

    #-- Xendor4ispWindow.on_xisp_add_button_clicked }
```

Now we will tell the app what to do when users click on Remove button.
First we check if a line of the CList has been selected, if yes, we remove the line, delete the bitpodder.fwd file, create a new bitpodder.fwp file and store the remaining lines of the CList. To finish, we will reload the app by opening a new window and destroying the current window.

```
    #-- Xendor4ispWindow.on_xisp_remove_button_clicked {
    def on_xisp_remove_button_clicked(self, widget, *args):
        if len(XISP_Options_Row_Variable)!= 0:
            xtrow = XISP_Options_Row_Variable[0]
            if xtrow != "":
                self.xisp_option_clist.remove(xtrow)

            fdw = open("bitpodder.fwp", 'w')
            for drow in range(self.xisp_option_clist.rows):
                ame = self.xisp_option_clist.get_text(drow, 0)
                d = self.xisp_option_clist.get_text(drow, 1)
                ch = ame + ":?:?:" + d
                fdw.write(ch)
                fdw.write('\n')
            fdw.close()

            bitpodder_window = BitPodderWindow()
            self.bitpodder_window.destroy()

    #-- Xendor4ispWindow.on_xisp_remove_button_clicked }
```

```
    #-- Xendor4ispWindow.on_xisp_option_clist_select_row {
    def on_xisp_option_clist_select_row(self, widget, *args):
        global XISP_Options_Row_Variable
        XISP_Options_Row_Variable = self.xisp_option_clist.selection
    #-- Xendor4ispWindow.on_xisp_option_clist_select_row }
```

Now we will see how work the second window of BitPodder. This window will print informations about the torrent, if there are no informations printed: the torrent is not valid or a timeoutsocket error appears.

When the window is launched, we will check if the global variable XISP_Link contains a valid url. If yes, we open the .torrent file without downloading it and search for informations.

I will not explain how Bram Cohen decided to write the .torrent files; if you are interested in Bit Torrent working, I advise you to read the code of Bit Torrent that is well written (Bit Torrent is written in python for those who didn't know it) or read my code below, it's very easy to understand.

```
class InfoWindow(SimpleGladeApp):

    def __init__(self, path="glade\\bitpodder.glade",
                 root="info_window",
                 domain=app_name, **kwargs):
        path = os.path.join(glade_dir, path)
        SimpleGladeApp.__init__(self, path, root, domain, **kwargs)

    #-- InfoWindow.new {
    def new(self):
        global XISP_Link
        link = XISP_Link

        metainfo_file = urllib.urlopen(link)
        metainfo = bdecode(metainfo_file.read())

        info = metainfo['info']
        info_hash = sha(bencode(info))
        txt = "metainfo file.: %s" % basename(link) + "\n"
        txt = txt + "info hash.....:  %s" % info_hash.hexdigest() + "\n"
        piece_length = info['piece length']
        if info.has_key('length'):
            # let's assume we just have a file
            txt = txt + "file name.....:  %s" % info['name'] + "\n"
            file_length = info['length']
            name ='file size.....:  '
        else:
            # let's assume we have a directory structure
            txt = txt + "directory name:  %s" % info['name'] + "\n"
            txt = txt + "files.....:  " + "\n"
            file_length = 0
            for file in info['files']:
                path = ''
                for item in file['path']:
                    if (path != ''):
                        path = path + "/"
                    path = path + item
                txt = txt + "   %s (%d)" % (path, file['length']) + "\n"
                file_length += file['length']
                name ='archive size..: '
        piece_number, last_piece_length = divmod(file_length, piece_length)
        txt = txt + "%s %i (%i * %i + %i)" \
```

```
            % (name,file_length, piece_number, piece_length, last_piece_length) + "\n"
        txt = txt + "announce url..:  %s" % metainfo['announce'] + "\n"
        if metainfo.has_key('announce-list'):
            list = []
            for tier in metainfo['announce-list']:
                for tracker in tier:
                    list+=[tracker,',']
                del list[-1]
                list+=['|']
            del list[-1]
            liststring = ''
            for i in list:
                liststring+=i
            txt = txt + "announce-list.:  %s" % liststring + "\n"
        if metainfo.has_key('httpseeds'):
            list = []
            for seed in metainfo['httpseeds']:
                list += [seed,'|']
            del list[-1]
            liststring = ''
            for i in list:
                liststring+=i
            txt = txt + "http seeds....:  %s" % liststring + "\n"
        if metainfo.has_key('comment'):
            txt = txt + "comment.......:  %s" % metainfo['comment'] + "\n"


        buf = self.metainfo_textview.get_buffer()
        buf.set_text(txt)
    #-- InfoWindow.new }

    #-- InfoWindow custom methods {
    #   Write your own methods here
    #-- InfoWindow custom methods }
```

Now to finish the app, we have to define the function get_xisp_id() that will return the url associated with the feed name we have to provide it like parameter. Then there is the entry point of the python script: the main function where we will launch the main window of the app.

```
def get_xisp_id(identifiant):
    fdpn = open("bitpodder.fwp").readlines()
    reqxisp = re.compile(identifiant)
    for pe in fdpn:
        if reqxisp.search(pe):
            PeersString = pe.split(":?:?:")
    return PeersString[1]

#-- main {

def main():
    bitpodder_window = BitPodderWindow()

    bitpodder_window.run()

if __name__ == "__main__":
    main()

#-- main }
```

Now try to launch bitpodder.py after having created a new file bitpodder.fwp in the same directory where bitpodder.py is located. BitPodder is ready and works fine!!!

VIII] How to compile BitPodder?

To compile a python application under windows platform, you have to create a setup.py script. This script will use the distutils packages and py2exe. Before writing this setup script, you have to download py2exe and install it (http://sourceforge.net/projects/py2exe).

Then you have to create the setup.py script. There is a good article explaining how to create setup.py scripts here: http://www.python.org/doc/current/dist/setup-script.html, but this article doesn't explain how to create .exe files with py2exe, so I decided to give you the setup.py script I use to compile BitPodder.

```
#!/usr/bin/env python
# setup.py
from distutils.core import setup
import py2exe
import glob

opts = {
    "py2exe": {
        "includes": "pango,atk,gobject",
        "dll_excludes": [
        "iconv.dll","intl.dll","libatk-1.0-0.dll",
        "libgdk_pixbuf-2.0-0.dll","libgdk-win32-2.0-0.dll",
        "libglib-2.0-0.dll","libgmodule-2.0-0.dll",
        "libgobject-2.0-0.dll","libgthread-2.0-0.dll",
        "libgtk-win32-2.0-0.dll","libpango-1.0-0.dll",
        "libpangowin32-1.0-0.dll"],
        }
    }

setup(
    name = "BitPodder",
    description = ".torrent podcaster from Nzeka Labs",
    version = "0.1",
    author="Nzeka Gilbert",
    author_email="khaalel@gmail.com",
    url="http://www.nzeka-labs.com",
    windows = [
        {"script": "bitpodder.py",
        "icon_resources": [(1, "sage.ico")]
        }
    ],
    options=opts,
    data_files=[("pixmaps", glob.glob("pixmaps/*.png")),
                ("glade", glob.glob("glade/*.*"))
    ],
)
```

When compiling, don't forget to put sage.ico in the directory or you will have a beautiful error during compilation. If compilation succeeded, you should have 2 new repertories: \dist\ and \build\.
The .exe is in dist repertory. You can delete the build repertory.

If you want to create an installer, I advise you to use 7-zip or NSIS (from the creators of Winamp). Here is the script I used with zip2exe from NSIS Menu:

```
;Change this file to customize zip2exe generated installers with a modern interface

!include "MUI.nsh"

!insertmacro MUI_PAGE_DIRECTORY
!insertmacro MUI_PAGE_INSTFILES

!insertmacro MUI_LANGUAGE "English"

Section -post
 SetOutPath $INSTDIR
 CreateShortCut "$DESKTOP\BitPodder.lnk" "$INSTDIR\bitpodder.exe"
 CreateDirectory "$SMPROGRAMS\Nzeka Labs"
 CreateDirectory "$SMPROGRAMS\Nzeka Labs\BitPodder"
 CreateShortCut "$SMPROGRAMS\Nzeka Labs\BitPodder\BitPodder.lnk" "$INSTDIR\bitpodder.exe"
""
 WriteINIStr "$SMPROGRAMS\Nzeka Labs\Nzeka Labs.url" "InternetShortcut" "URL"
"http://www.nzeka-labs.com/"
 WriteINIStr "$SMPROGRAMS\Nzeka Labs\Xendor Site.url" "InternetShortcut" "URL"
"http://korriban-planet.blogspot.com/"
SectionEnd
```

To understand what I wrote in this NSIS script, I advise you to read the NSIS help.

This article is finished. If you find this article interesting and/or useful, please send me an email in order to let me know what you think about this article. If you find this article boring/null, please send me an email too with suggestion to improve the article.

I hope my (perhaps bad) english didn't bother you; it's not my first language: I'm a 18-years-old french.

END