SPOKEN TUTORIAL

ABSRACT

This 'Spoken Tutorial' Project is a module spoken video tutorials. Developing a new software using Python and QT in GNU/Linux Platform. This project is to assist the people contributing to the project Spoken Tutorial project that brings video tutorials on most of the Indian Languages on various Free Open Source Software and Daily computer uses.

This project will help the contributors of the Spoken Tutorial Project to tranlsate the existing videos into other languages with a great speed and accuracty by eleminating all the human intervance on editing the audio file. The Qt – Designer will newly developing and creating system applications, Qt written by c++ languages, Qt Designer is a tool for designing and building graphical user interfaces (Guis) from Qt widgets. It is possible to compose and customize the widgets or dialogs and test them using different styles and resolutions directly in the editor. Widgets and forms created with Qt Designer are integrated with programmed code, using the Qt signals and slots mechanism.

HARDWARE & SOFTWATE SPECIFICATION

1. Hardware specification:

- Processor : Pentium IV
- Processor Clock speed : 1.4 GHz
- RAM : 512 MSB
- HDD : 40 GB
- MONITOR : 14 INCH
- KEYBOARD : 108 KEYS
- MOUSE : 3 BUTTON SCROLL

2. Software specification:

- Technology : Pymedia
- Front end : QT-Designer
- Operating Platform : Linux, Window,Mac...
- Languages : Python,Pyqt

SOFRWARE ENVIRNOMENTS

Python:

- Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python’s elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

- The Python interpreter is easily extended with new functions and data types implemented in C or C++ (or other languages callable from C). Python is also suitable as an extension language for customizable applications.

- Python is a high-level, interpreted, interactive and object oriented-scripting language.

- Python was designed to be highly readable which uses English keywords frequently where as other languages use punctuation and it has fewer syntactical constructions than other languages.

- **Python is Interpreted:** This means that it is processed at runtime by the interpreter and you do not need to compile your program before executing it. This is similar to PERL and PHP.

- **Python is Interactive:** This means that you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

- **Python is Object-Oriented:** This means that Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

- **Gui Programming:** Python supports Gui applications that can be created and ported to many system calls, libraries, and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

QT-Designer:

- Qt is multiplatform C++ (with interfaces to other languages) Gui library. There are also other possibilities, even easier, how to write the Gui application. But Qt is quite common and popular way to build stable, portable, nice-looking application. There are many reasons for this solution and also against it. From educational angle of view, by using Qt you will learn a lot of useful things and gain necessary experience which could be exploited in other projects.

- Qt Creator is a cross-platform integrated development environment (IDE) tailored to the needs of Qt developers. Qt Creator runs on Windows, Linux/X11 and Mac OS X desktop operating systems, and allows developers to create applications for multiple desktop and mobile device platforms.

- **Developing Application UI -** Large high-resolution screens, touch input, and significant graphics power are becoming common in portable consumer devices, such as mobile phones, media players, set-top boxes, and netbooks. To fully benefit from these features and to create intuitive, modern-looking, fluid user interfaces, you can use Qt Quick.

- **Qt Quick -** Qt Quick is a collection of technologies that are designed to help developers create the kind of intuitive, modern, fluid user interfaces that are increasingly used on mobile phones, media players, set-top boxes and other portable devices. Qt Quick consists of a rich set of user interface elements, a declarative language for describing user interfaces and a language runtime. A collection of C++ APIs is used to integrate these high level features with classic Qt applications.

- **Qt Simulator Manual -** With Qt Simulator, you can test Qt applications that are intended for mobile devices in an environment similar to that of the device. You can change the information that the device has about its configuration and environment. Qt Simulator does not support any device specific APIs by design. Therefore, applications that run well on Qt Simulator also run on any device that hosts the Qt and Qt Mobility libraries. However, this means that you cannot use Qt Simulator to test applications that use device specific libraries, such as Symbian C++ APIs.

PyQt4:

- PyQt brings together the Qt C++ cross-platform application framework and the cross-platform interpreted language Python.

- Qt also includes Qt Designer, a graphical user interface designer. PyQt is able to generate Python code from Qt Designer. It is also possible to add new Gui controls written in Python to Qt Designer.

- This is PyQt4 tutorial. The tutorial is suited for beginners and intermediate programmers. After reading this tutorial, you will be able to program non trivial PyQt4 applications.

- PyQt4 is a toolkit for creating Gui applications. It is a blending of Python programming language and the successfull Qt library. Qt library is one of the most powerful Gui libraries. The official home site for PyQt4 is on www.riverbankcomputing.co.uk/news It was developed by **Phil Thompson**.

- PyQt4 is implemented as a set of Python modules. It has over 300 classes and almost 6000 functions and methods. It is a multiplatform toolkit. It runs on all major operating systems. Including Unix, Windows and Mac. PyQt4 is dual licenced. Developers can choose between GPL and commercial licence. Previously, GPL version was available only on Unix. Starting from PyQt version 4, GPL licence is available on all supported platforms.

- **Classes available:**

    1. QtGui
    2. QtNetwork
    3. QtXml
    4. QtSvg
    5. QtOpenGL
    6. QtSql

- The QtCore module contains the core non-Gui functionality. This module is used for working with time, files and directories, various data types, streams, urls, mime types, threads or processes. The QtGui module contains the graphical components and related classes. These include for example buttons, windows, status bars, toolbars, sliders, bitmaps, colors, fonts etc. The QtNetwork module contains the classes for network programming. These classes allow to write TCP/IP and UDP clients and servers. They make the network programming easier and more portable. The QtXml contains classes for working with xml files. This module provides implementation for both SAX and DOM APIs. The QtSvg module provides classes for displaying the contents of SVG files. Scalable Vector Graphics (SVG) is a language for describing two-dimensional graphics and graphical applications in XML. The QtOpenGL module is used for rendering 3D and 2D graphics using the OpenGL library. The module enables seamless integration of the Qt Gui libary and the OpenGL library. The QtSql module provides classes for working with databases.

Pymedia:

- Pymedia is a Python module for wav, mp3, ogg, avi, divx, dvd, cdda etc files manipulations. It allows you to parse, demutiplex, multiplex, decode and encode all supported formats. It can be compiled for Windows, Linux and cygwin.

- This small tutorial describes basic objects that allow you to play sound on all supported platforms. In many ways it is very similar to ossaudiodev module which is standard module for Python on many Unix based systems.
- Encode/decode audio compressed streams. The following formats are available:

    - **WMA**(v1 and v2)
    - **AC3**
    - **OGG**(optional with vorbis library)
    - **AAC**(optional with faad library)
    - **MP3,MP2**

- Encode/decode video compressed streams. The following formats supported:

    - **AVI(divx, xvid)**, generic file format, carrying many possible streams
    - **ASF(wmv1/2)**, generic file format, carrying many possible streams
    - **MPEG1,2**(VCD, SVCD, DVD compatible)

- Sound output through the **OSS / Waveout.** Mutlichannel for digital output
- Sound input through the **OSS / Wavein.**
- Sound mixer manipulation, list all lines, set/get values of every line.
- Sound manipulation classes such as SpectrAnalyzer, Resampler.
- Video manipulation to convert video frames between YUV and RGB formats.
- Direct CD/DVD ROM access to read audio/video tracks in a raw format.
  This way you can play Audio, DVD and Video CDs using the same interface

PROJECT DESCRIPTION

The “SPOKEN TUTORIAL” project as two type of use that

- Terminal – Interpreted
- Gui - Application

Terminal:

- The Spoken Tutorial using python, it as a interpreted language we will create voice record audio file dependancy pymedia.
- The Spoken Tutorial almost create in voice recored and recored audio files non-Gui application.
- Everything create voice recored file using in Terminal it also called in interpreted.
- When pymedia program execute on the terminal as soon as see that result that aslo called in interpreted.
- The file execute in normal type of this is.Ex:- file_name.py, it is simple execute in progarm will be using terminal.
- Before should know that how to use in the terminal and easy execute in the program.

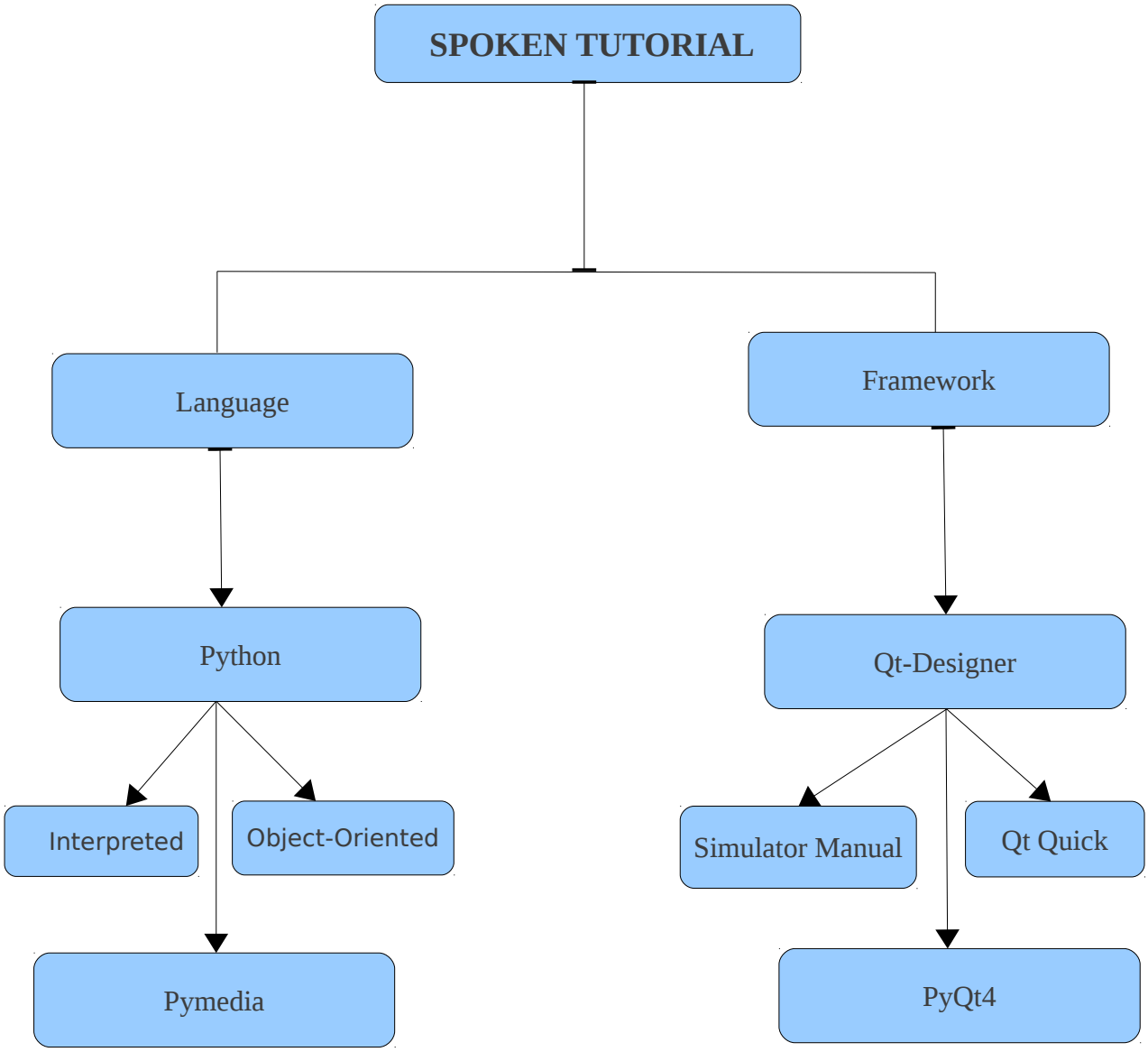
Qt-Designer:

- The spoken tutorial project easy understand any one, whatever we need to system based Gui applications easy create using in Qt-Designer.
- This project will be for anyone to use, share, modify and contribute for free, without any resrictions.
- The python language has efficient high-level data structures and a simple but effective approach to object-oriented programming.
- Qt-Designer is Framework, Qt-Designer in support python language, whatever we can need Gui application python language use create.
- Python and qt-Designer in compare that, Pyqt is name.
- The pymedia can be use create voice record whichever languages we need it and whichever format we convert.
- First we should be using Qt-Designer whatever model need to Gui application create and both languages using pyton and pymedia.
- Create pyqt Gui application execute in terminal.
  First pyuic4 -x file_name.ui -o sample.py

  1. pyuic4 - it is version of software in pymedia.
  2. -x - it is complier of the ui application.
  3. -o - it is code generate Ex:- sample.py

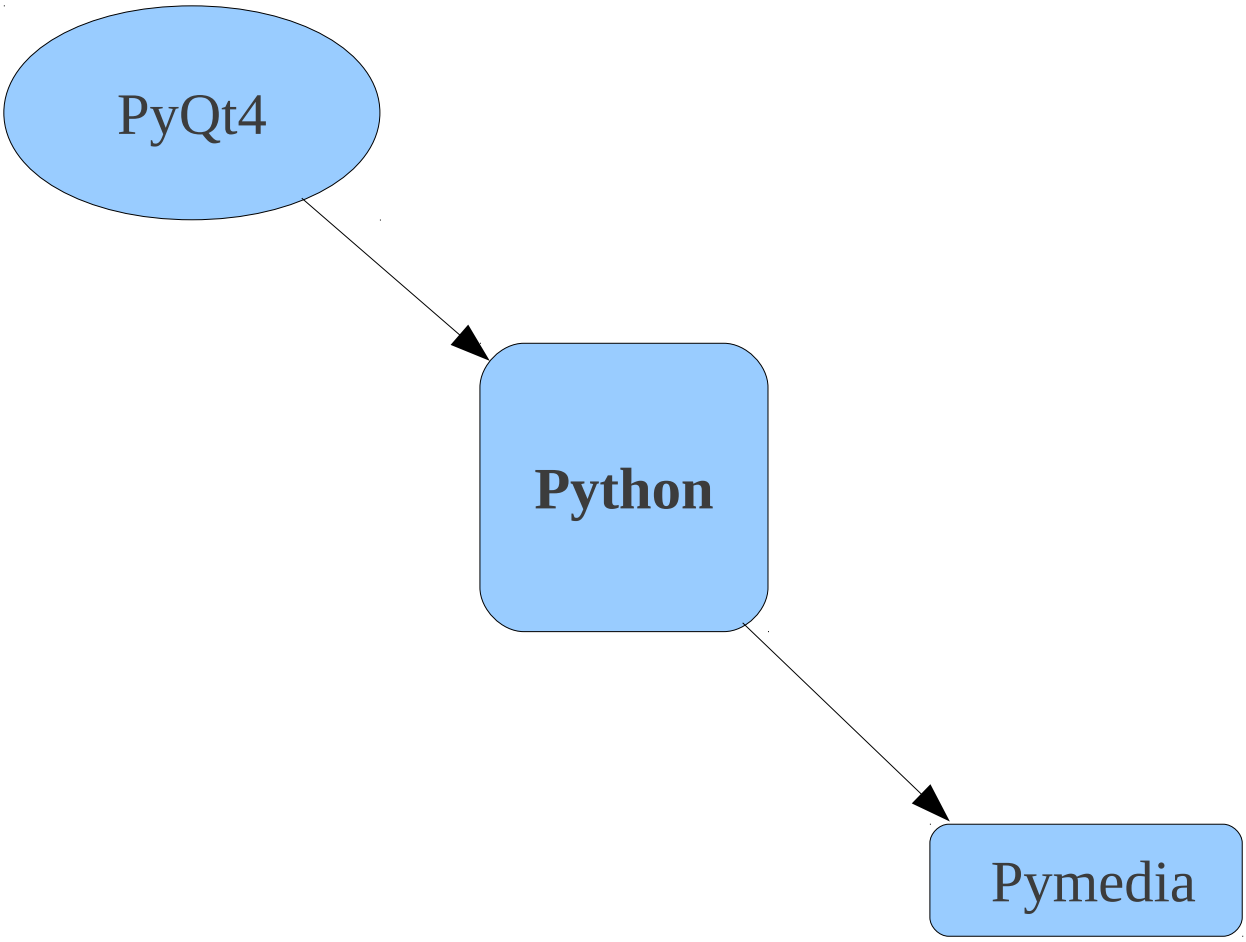
- Manual code in Ex:- code.py... The code.py with in inside we will be write
  manual using python and pymedia.
- We already create Gui application generate file sample.py name called inside the code.py Ex:- from sample import UI_MainWindow.
- Here also two files, 1.voice record 2. record audio.

DATA FLOW DIAGRAM



DATA FLOW DIAGRAM

Program structure



SAMPLE CODING

I. Voice record
II. Record audio

I. Voice record

a. It will first execute manual use in the terminal.

b. Creating Gui application uisng PyQt and dependancy file will manual create.

II. Record audio

a. It will first execute manual use in the terminal.

b.Creating Gui application uisng PyQt and dependancy file will manual create.

Voice Record

- This is Voice record program it will be execute in Terminal.
- The Voice record need to Spoken Tutorial because will be creating new sub voice audio Ex:- Tamil, English, Hindi, Telung.
- The voice record program whichever will be voice record it and whatever we need formate it available Ex:- Mp3, wav, ogg.
- The Voice record appliations non-Gui application.

Program:-

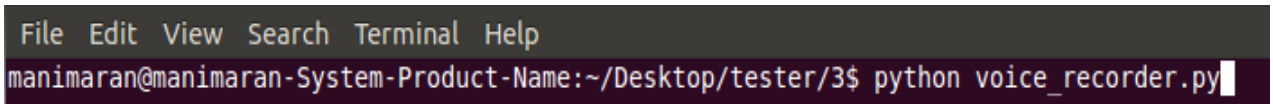
voice_record.py

```
import time, sys
import pymedia.audio.sound as sound
import pymedia.audio.acodec as acodec

def voiceRecorder( secs, channels, name ):
  f= open( name, 'wb' )
  # Minimum set of parameters we need to create Encoder
  cparams= { 'id': acodec.getCodecID( 'mp3' ),
             'bitrate': 138000,
             'sample_rate': 44100,
             'channels': channels }
  ac= acodec.Encoder( cparams )
  snd= sound.Input( 44100, channels, sound.AFMT_S16_LE )
  snd.start()

  # Loop until recorded position greater than the limit specified
  while snd.getPosition()<= secs:
    s= snd.getData()
    if s and len( s ):
      for fr in ac.encode( s ):
        # We definitely should use mux first, but for
        # simplicity reasons this way it'll work also
        f.write( fr )
    else:
      time.sleep( .003 )

  # Stop listening the incoming sound from the microphone or line in
  snd.stop()

# --------------------------------------------------------------------------------
# Record stereo sound from the line in or microphone and save it as mp3
file
# Specify length and output file name
# http://pymedia.org/
if __name__ == "__main__":
  if len( sys.argv )!= 4:
    print 'Usage: voice_recorder <seconds> <channels> <file_name>'
  else:
    voiceRecorder( int( sys.argv[ 1 ] ), int( sys.argv[ 2 ] ), sys.argv[ 3 ]  )
```

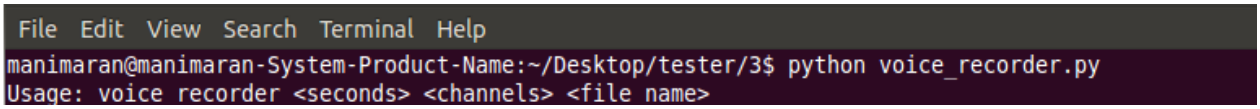
Execute the program:-

1. The program was normally execute in python.
2. Open terminal python File_name.py

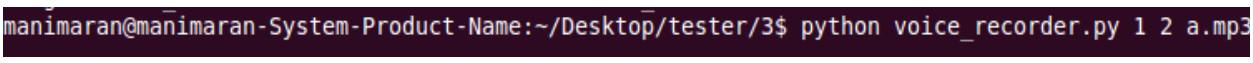
A . Program execute



B. Call voice_recorder.py
seconds : 1 channels: 2 file name: a.mp3



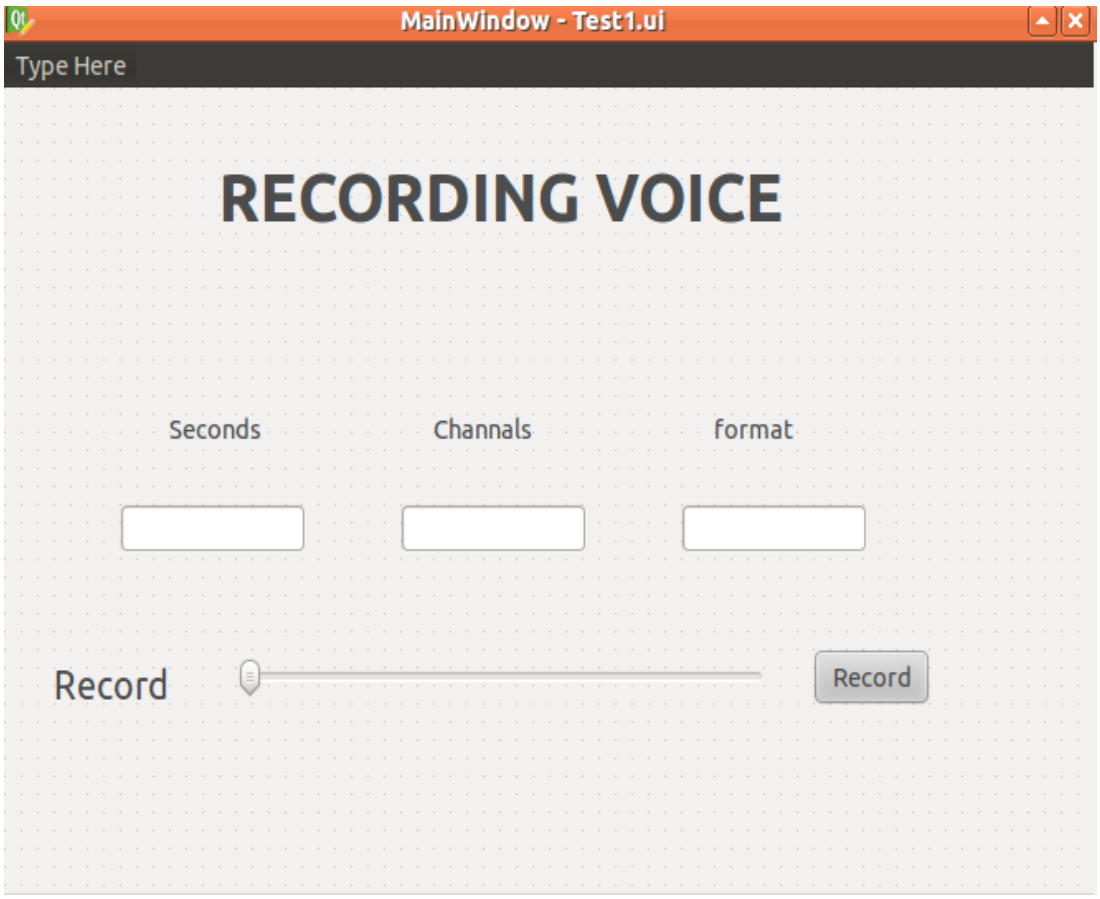
C. Program call voice _recorder.py 1 2 a.mp3



voice recording now.

Voice record – Gui application

a. Create Gui – Test1.ui
b. Code generate - Test_interface.py
c. Manual code write – code.py



b. Code generate

```
    # -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'Test1.ui'
#
# Created: Wed Oct 10 01:51:45 2012
#      by: PyQt4 UI code generator 4.7.4
#
# WARNING! All changes made in this file will be lost!

from PyQt4 import QtCore, QtGui

class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(640, 480)
        font = QtGui.QFont()
        font.setStrikeOut(False)
        MainWindow.setFont(font)
        MainWindow.setMouseTracking(False)
        self.centralwidget = QtGui.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.label = QtGui.QLabel(self.centralwidget)
        self.label.setGeometry(QtCore.QRect(100, 180, 67, 17))
        self.label.setObjectName("label")
        self.label_2 = QtGui.QLabel(self.centralwidget)
        self.label_2.setGeometry(QtCore.QRect(260, 180, 67, 17))
        self.label_2.setObjectName("label_2")
        self.label_3 = QtGui.QLabel(self.centralwidget)
        self.label_3.setGeometry(QtCore.QRect(430, 180, 67, 17))
        self.label_3.setObjectName("label_3")
        self.lineEdit = QtGui.QLineEdit(self.centralwidget)
        self.lineEdit.setGeometry(QtCore.QRect(70, 230, 113, 27))
        self.lineEdit.setObjectName("lineEdit")
        self.lineEdit_2 = QtGui.QLineEdit(self.centralwidget)
        self.lineEdit_2.setGeometry(QtCore.QRect(240, 230, 113, 27))
        self.lineEdit_2.setObjectName("lineEdit_2")
        self.lineEdit_3 = QtGui.QLineEdit(self.centralwidget)
        self.lineEdit_3.setGeometry(QtCore.QRect(410, 230, 113, 27))
        self.lineEdit_3.setObjectName("lineEdit_3")
        self.pushButton = QtGui.QPushButton(self.centralwidget)
        self.pushButton.setGeometry(QtCore.QRect(80, 310, 71, 31))
        self.pushButton.setObjectName("pushButton")
        self.horizontalSlider = QtGui.QSlider(self.centralwidget)
```

```
        self.horizontalSlider.setGeometry(QtCore.QRect(210, 310, 321, 31))
        self.horizontalSlider.setOrientation(QtCore.Qt.Horizontal)
        self.horizontalSlider.setObjectName("horizontalSlider")
        self.label_4 = QtGui.QLabel(self.centralwidget)
        self.label_4.setGeometry(QtCore.QRect(130, 30, 351, 61))
        font = QtGui.QFont()
        font.setPointSize(28)
        font.setWeight(75)
        font.setBold(True)
        self.label_4.setFont(font)
        self.label_4.setObjectName("label_4")
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtGui.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 640, 25))
        self.menubar.setObjectName("menubar")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtGui.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)

        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):

MainWindow.setWindowTitle(QtGui.QApplication.translate("MainWindow",
"MainWindow", None, QtGui.QApplication.UnicodeUTF8))
        self.label.setText(QtGui.QApplication.translate("MainWindow",
"Seconds", None, QtGui.QApplication.UnicodeUTF8))
        self.label_2.setText(QtGui.QApplication.translate("MainWindow",
"Channals", None, QtGui.QApplication.UnicodeUTF8))
        self.label_3.setText(QtGui.QApplication.translate("MainWindow",
"format", None, QtGui.QApplication.UnicodeUTF8))
        self.pushButton.setText(QtGui.QApplication.translate("MainWindow",
"Record", None, QtGui.QApplication.UnicodeUTF8))
        self.label_4.setText(QtGui.QApplication.translate("MainWindow",
"RECORDING VOICE", None, QtGui.QApplication.UnicodeUTF8))

if __name__ == "__main__":
    import sys
    app = QtGui.QApplication(sys.argv)
    MainWindow = QtGui.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())
```

C.Manual code write

```
import time, sys
from PyQt4 import QtGui,QtCore
from Test_interface import Ui_MainWindow

import pymedia.audio.sound as sound
import pymedia.audio.acodec as acodec

class Test_one(QtGui.QMainWindow):
	def __init__(self,Parent = None):
		self.ui = Ui_MainWindow()
		self.ui.setupUi(self)
		#self.initUI()

		# connect in button line

	#def initUI(self):

		self.ui.connect(self.ui.pushButton,
QtCore.SIGNAL('buttonPressed(int)'), self.VoiceRecord)

		#self.ui.button_open.clicked.connect(self.voiceRecorder)


	def voiceRecorder(self, initUI,secs, channels, name ):
		f= open( name, 'wb' )
	  # Minimum set of parameters we need to create Encoder
		cparams= { 'id': acodec.getCodecID( 'mp3' ),
				'bitrate': 138000,
				'sample_rate': 44100,
				'channels': channels }
		ac= acodec.Encoder( cparams )
		snd= sound.Input( 44100, channels,
sound.AFMT_S16_LE )
		snd.start()

	  # Loop until recorded position greater than the limit specified
		while snd.getPosition()<= secs:
			s= snd.getData()
		if s and len( s ):
```

```
                for fr in ac.encode( s ):
                    # We definitely should use mux first, but for
                    # simplicity reasons this way it'll work also
                    f.write( fr )
            else:
                time.sleep( .003 )

        # Stop listening the incoming sound from the microphone or
line in
            snd.stop()

    # ------------------------------------------------------------------------
    # Record stereo sound from the line in or microphone and save
it as mp3 file
    # Specify length and output file name
    # http://pymedia.org/
#if __name__ == "__main__":
if len( sys.argv )!= 4:
    print 'Usage: voice_recorder <seconds> <channels>
<file_name>'
else:
    voiceRecorder( int( sys.argv[ 1 ] ), int( sys.argv[ 2 ] ), sys.argv[
3 ]  )

if __name__ == "__main__":
    #app = QtGui.QApplication(sys,argv)
    myapp =  Test_one()
    myapp.show()
    sys.exit(app.exec__())
```

Recor audio

- This is Record audio program it will be execute in Terminal.
- The Record audio need to Spoken Tutorial because will be changing whichever format Ex:- ogg,wav,aac,ac3.
- The Voice record appliations non-Gui application.

Progaram:

record_audio.py

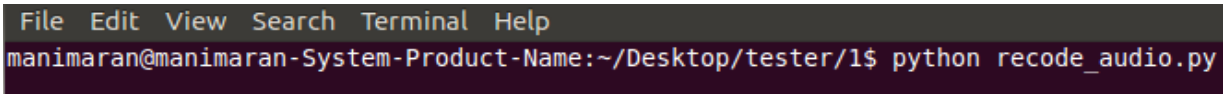
```
#! /bin/env python

import sys, time, traceback

# Simple  audio encoder
def recodeAudio( fName, fOutput, type, bitrate= None ):
  # ----------------------------------
  import pymedia.audio.acodec as acodec
  import pymedia.muxer as muxer
  # Open demuxer
  dm= muxer.Demuxer( fName.split( '.' )[ -1 ].lower() )
  f= open( fName, 'rb' )
  s= f.read( 90000 )
  dec= enc= mx= None
  print 'Recoding %s into %s' % ( fName, fOutput )
  while len( s ):
    frames= dm.parse( s )
    if frames:
      for fr in frames:
        # Assume for now only audio streams
        if dec== None:
          # Open decoder
          dec= acodec.Decoder( dm.streams[ fr[ 0 ] ] )
          print 'Decoder params:', dm.streams[ fr[ 0 ] ]

        # Decode audio frame
        r= dec.decode( fr[ 1 ] )
        if r:
          if bitrate== None:
            bitrate= r.bitrate

          # Open muxer and encoder
          if enc== None:
            params= { 'id': acodec.getCodecID(type),
```

```
                    'bitrate': bitrate,
                    'sample_rate': r.sample_rate,
                    'channels': r.channels }
            print 'Encoder params:', params
            mx= muxer.Muxer( type )
            stId= mx.addStream( muxer.CODEC_TYPE_AUDIO, params )
            enc= acodec.Encoder( params )
            fw= open(fOutput, 'wb')
            ss= mx.start()
            fw.write(ss)

          enc_frames= enc.encode( r.data )
          if enc_frames:
            for efr in enc_frames:
              ss= mx.write( stId, efr )
              if ss:
                fw.write(ss)

    s= f.read( 100000 )

  f.close()

  if fw:
    if mx:
      ss= mx.end()
      if ss:
        fw.write(ss)
    fw.close()

# --------------------------------------------------------------------------------
# Change the format of your compressed audio files to something
different
# http://pymedia.org/
if __name__== '__main__':
  if len( sys.argv )< 5 or len( sys.argv )> 6:
    print "Usage: recode_audio.py <audio_input_file>
<audio_output_file> <format_name> [ <bitrate> ]"
  else:
    if len( sys.argv )== 4:
      recodeAudio( sys.argv[1], sys.argv[2], sys.argv[3] )
    else:
      recodeAudio( sys.argv[1], sys.argv[2], sys.argv[3],
int( sys.argv[4] )* 1000 )
```

Execute the program:-

1. The program was normally execute in python.
2. Open terminal python File_name.py

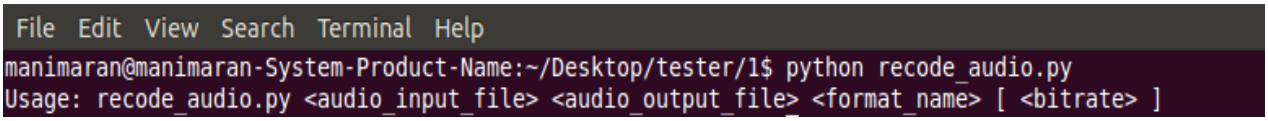
A. Program call record_audio.py



B. Program execute python record_audio.py

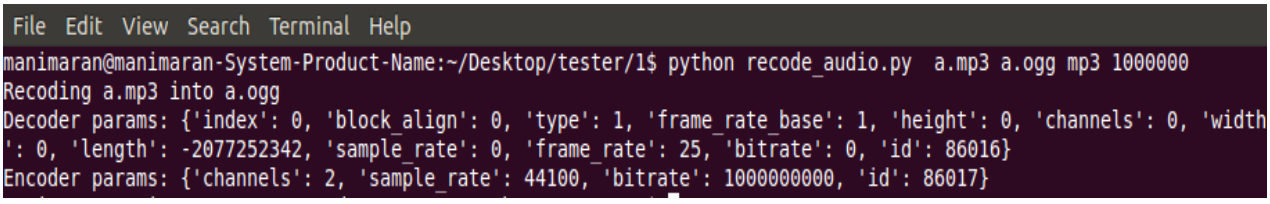
<audio_input_file> <audio_output_file> <format_name> [<bitrate>]

a.mp3 a.ogg mp3 1000000



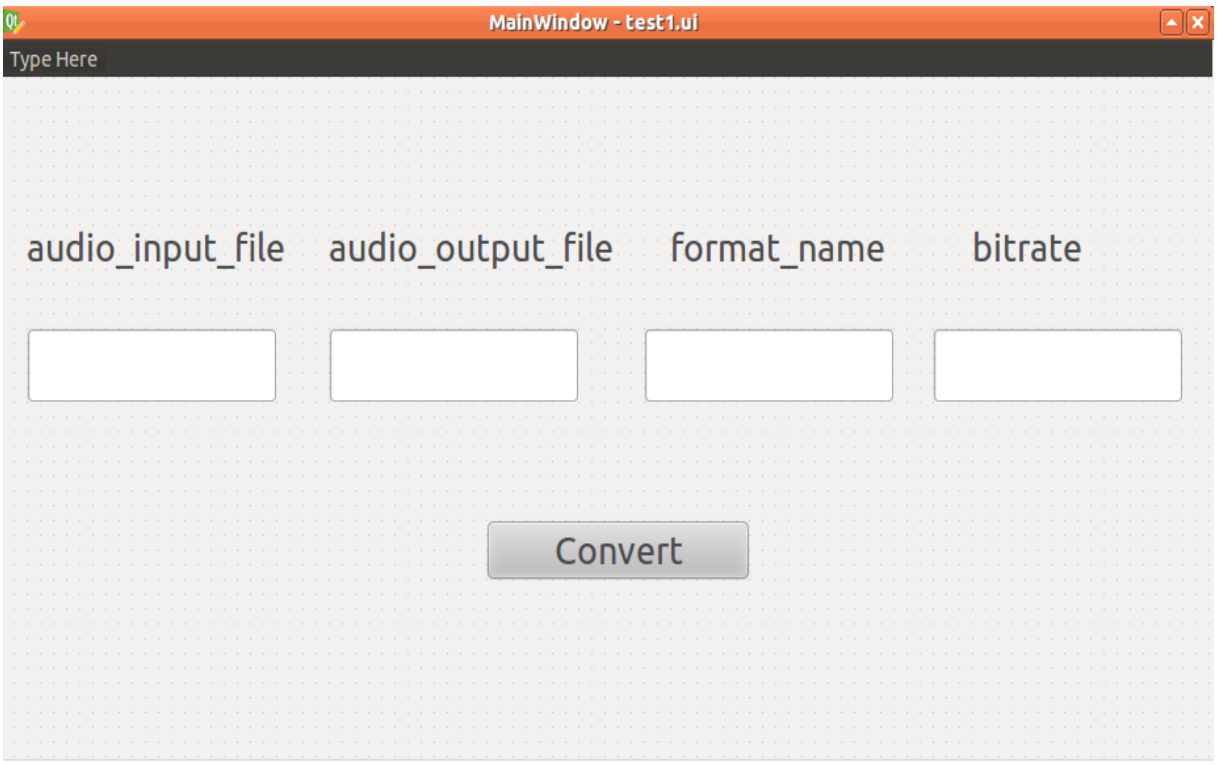
C. Program now record mp3 to ogg format

D. record audio finished



Record audio – Gui Application

a. Create Gui – Test1.ui
b. Code generate - Test_interface.py
c. Manual code write – code.py



b. Code generate

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'test1.ui'
#
# Created: Thu Oct 11 14:29:28 2012
#      by: PyQt4 UI code generator 4.7.4
#
# WARNING! All changes made in this file will be lost!

from PyQt4 import QtCore, QtGui

class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.setEnabled(True)
        MainWindow.resize(912, 420)
        self.centralwidget = QtGui.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.label = QtGui.QLabel(self.centralwidget)
        self.label.setGeometry(QtCore.QRect(20, 100, 201, 31))
        font = QtGui.QFont()
        font.setPointSize(20)
        self.label.setFont(font)
        self.label.setObjectName("label")
        self.label_2 = QtGui.QLabel(self.centralwidget)
        self.label_2.setGeometry(QtCore.QRect(250, 100, 221, 31))
        font = QtGui.QFont()
        font.setPointSize(20)
        self.label_2.setFont(font)
        self.label_2.setObjectName("label_2")
        self.label_3 = QtGui.QLabel(self.centralwidget)
        self.label_3.setGeometry(QtCore.QRect(510, 100, 171, 31))
        font = QtGui.QFont()
        font.setPointSize(20)
        self.label_3.setFont(font)
        self.label_3.setObjectName("label_3")
        self.label_4 = QtGui.QLabel(self.centralwidget)
```

```
        self.label_4.setGeometry(QtCore.QRect(740, 100, 91, 31))
        font = QtGui.QFont()
        font.setPointSize(20)
        self.label_4.setFont(font)
        self.label_4.setObjectName("label_4")
        self.lineEdit = QtGui.QLineEdit(self.centralwidget)
        self.lineEdit.setGeometry(QtCore.QRect(20, 170, 191, 51))
        font = QtGui.QFont()
        font.setPointSize(18)
        self.lineEdit.setFont(font)
        self.lineEdit.setObjectName("lineEdit")
        self.lineEdit_2 = QtGui.QLineEdit(self.centralwidget)
        self.lineEdit_2.setGeometry(QtCore.QRect(250, 170, 191, 51))
        font = QtGui.QFont()
        font.setPointSize(18)
        self.lineEdit_2.setFont(font)
        self.lineEdit_2.setObjectName("lineEdit_2")
        self.lineEdit_3 = QtGui.QLineEdit(self.centralwidget)
        self.lineEdit_3.setGeometry(QtCore.QRect(490, 170, 191, 51))
        font = QtGui.QFont()
        font.setPointSize(18)
        self.lineEdit_3.setFont(font)
        self.lineEdit_3.setObjectName("lineEdit_3")
        self.lineEdit_4 = QtGui.QLineEdit(self.centralwidget)
        self.lineEdit_4.setGeometry(QtCore.QRect(710, 170, 191, 51))
        font = QtGui.QFont()
        font.setPointSize(18)
        self.lineEdit_4.setFont(font)
        self.lineEdit_4.setObjectName("lineEdit_4")
        self.pushButton = QtGui.QPushButton(self.centralwidget)
        self.pushButton.setGeometry(QtCore.QRect(370, 300, 201,
41))
        font = QtGui.QFont()
        font.setPointSize(20)
        self.pushButton.setFont(font)
        self.pushButton.setObjectName("pushButton")
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtGui.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 912, 25))
        self.menubar.setObjectName("menubar")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtGui.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)
```

```
        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):

MainWindow.setWindowTitle(QtGui.QApplication.translate("MainWin
dow", "MainWindow", None, QtGui.QApplication.UnicodeUTF8))
        self.label.setText(QtGui.QApplication.translate("MainWindow",
"audio_input_file", None, QtGui.QApplication.UnicodeUTF8))

self.label_2.setText(QtGui.QApplication.translate("MainWindow",
"audio_output_file", None, QtGui.QApplication.UnicodeUTF8))

self.label_3.setText(QtGui.QApplication.translate("MainWindow",
"format_name", None, QtGui.QApplication.UnicodeUTF8))

self.label_4.setText(QtGui.QApplication.translate("MainWindow",
"bitrate", None, QtGui.QApplication.UnicodeUTF8))

self.pushButton.setText(QtGui.QApplication.translate("MainWindow",
"Convert", None, QtGui.QApplication.UnicodeUTF8))


if __name__ == "__main__":
    import sys
    app = QtGui.QApplication(sys.argv)
    MainWindow = QtGui.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())
```

c. Manual code write

```
import time, sys
from PyQt4 import QtGui,QtCore
from test_interface import Ui_MainWindow
import pymedia.audio.acodec as acodec
import pymedia.muxer as muxer

import sys, time, traceback


class test_two(QtGui.QMainWindow):
        def __init__(self,Parent = None):
                self.ui = Ui_MainWindow()
                self.ui.setUpUi()
                self.initUi()


        def initUI(self):




QtCore.QObject.connect(self.ui.pushButton,QtCore.SIGNAL("clicked(
)"), self.recodeAudio)


        def recodeAudio( fName, fOutput, type, bitrate= None ):
                # ----------------------------------

                # Open demuxer
```

```
        dm= muxer.Demuxer( fName.split( '.' )[ -1 ].lower() )
        f= open( fName, 'rb' )
        s= f.read( 90000 )
        dec= enc= mx= None
        print 'Recoding %s into %s' % ( fName, fOutput )
        while len( s ):
            frames= dm.parse( s )
            if frames:
              for fr in frames:
                # Assume for now only audio streams
                if dec== None:
                  # Open decoder
                  dec= acodec.Decoder( dm.streams[ fr[ 0 ] ] )
                  print 'Decoder params:', dm.streams[ fr[ 0 ] ]

                # Decode audio frame
                r= dec.decode( fr[ 1 ] )
                if r:
                  if bitrate== None:
                    bitrate= r.bitrate

                  # Open muxer and encoder
                  if enc== None:
                    params= { 'id': acodec.getCodecID(type),
                              'bitrate': bitrate,
                              'sample_rate': r.sample_rate,
                              'channels': r.channels }
                    print 'Encoder params:', params
                    mx= muxer.Muxer( type )
                    stId=
mx.addStream( muxer.CODEC_TYPE_AUDIO, params )
                    enc= acodec.Encoder( params )
                    fw= open(fOutput, 'wb')
                    ss= mx.start()
                    fw.write(ss)

                  enc_frames= enc.encode( r.data )
                  if enc_frames:
                    for efr in enc_frames:
                      ss= mx.write( stId, efr )
                      if ss:
                        fw.write(ss)

            s= f.read( 100000 )
```

```
                f.close()

                if fw:
                      if mx:
                        ss= mx.end()
                        if ss:
                              fw.write(ss)
                      fw.close()

               #
----------------------------------------------------------------------------
               # Change the format of your compressed audio files to
something different
               # http://pymedia.org/
               if __name__== '__main__':
                 if len( sys.argv )< 5 or len( sys.argv )> 6:
                      print "Usage: recode_audio.py <audio_input_file>
<audio_output_file> <format_name> [ <bitrate> ]"
                 else:
                      if len( sys.argv )== 4:
                        recodeAudio( sys.argv[1], sys.argv[2], sys.argv[3] )
                      else:
                        recodeAudio( sys.argv[1], sys.argv[2], sys.argv[3],
int( sys.argv[4] )* 1000 )

if __name__ =="__main__":
        myc = test_two()
        myc.show()
        sys.exit(app.exec__())
```

CONCLUSION

- This spoken video tutorial project, eleminating all the humanintervance on editing the audio files.
- Softwate like Python, PyQt4 and pymedia. Platform support in Linux,Mac,Windows.
- It will be about pyqt Gui application in show in Features still not now.
- It will have been now only execute in Terminal.

BIBLIOGRAPY

- Python : The Python Tutorial written by Guido van Rossum
  http://python.org/
- QT-Designer : doc.trolltech.com/3.3/designer-manual.html
- pyqt4 : www.rkblog.rk.edu.pl/w/p/python/
-
- pymedia : http://pymedia.org