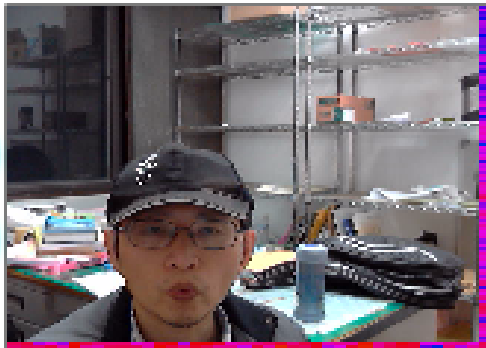
影音類，影音一起處理， by 呂仁園， 2014/03/30。



APAC 2014
PYCON
MAY 17-18 2014 in Taipei
May 18, 14:30, @R1

RyAudio, A Real-time Audio Spectrogram with Application to Sound-Driven Games in Python 3, Pyaudio, Pygame, and Pylab

Renyuan Lyu
呂仁園

Preface

- Python helps me implement the ***real-time spectrogram*** at the beginning of this year (2014).
  - After doing speech signal processing research for a long time, I feel so excited to share that excitement with friends.
  - So I submit my program with a youtube demo to this conference

- The followings are the scores and comments given by the reviewers

- Reviewer #1: Score: 2
  - No comments
- Reviewer #2: Score: 3
  - real-time speech recognizer !!!

- Reviewer #3: Score: 3
  - I'll admit being a bit selfish here. I have been planning to work on audio analysing for a while. This looks like a good start. :)

- Reviewer #4: Score: 0
  - After reviewing his code carefully, I have to say that his spectrogram analysis is not good for speech processing. He took every 512 samples to perform FFT to get spectrum under 16kHz sampling rate. As far as I know, speech processing will use so-called short-term frequency analysis, which is different than this one. Well, it might be an interesting topic for Python users as long as he provides accurate and correct information about DSP.

- Reviewer #5: Score: 2
  - Sound recognition is different than speech recognition, right?
- Reviewer #6: Score: 3
  - I am too excited to give any comment. I would even love to pay for his ticket just to listen to this talk.

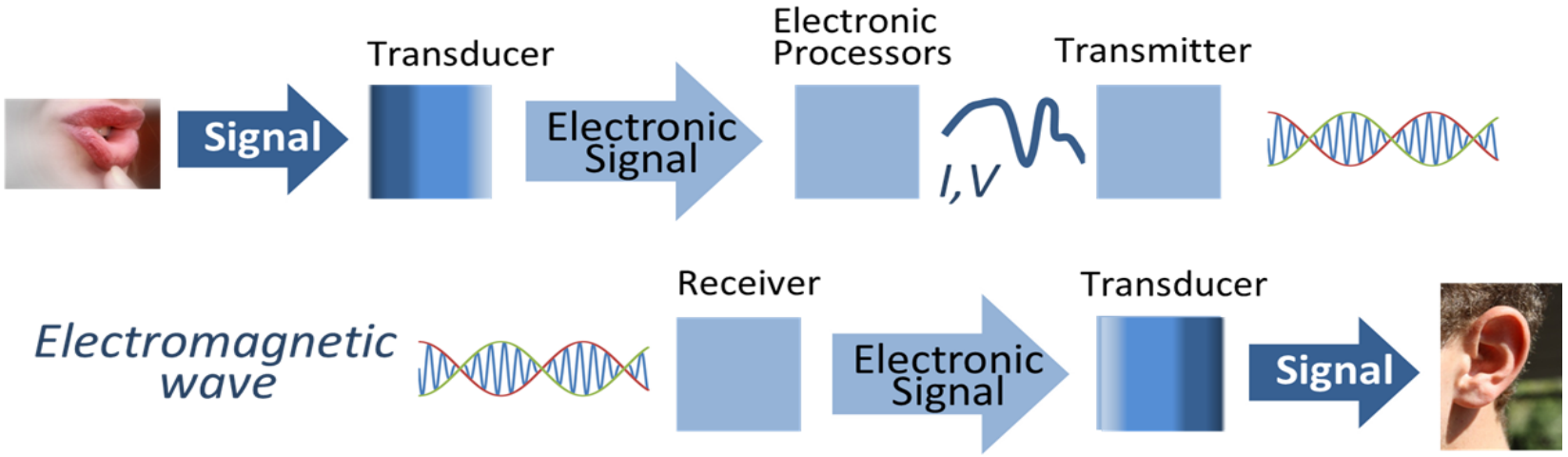
- By the way, **Python 3** allows me to use my ***native (most fluent) language*** to name the variables, functions, and classes.
  - That is even a more wonderful experience.
  - I can have much more precise, more elegant vocabulary to construct the program.

Overview

- Some Background on this talk
  - Signal Processing, Speech
  - Spectrum, Spectrogram
- Processing in Real Time
  - An Awesome Example: Friture
- RyAudio
  - A lighter example for realtime spectrogram
- Demo
- Some Comments on Programming in Native Languages
  - Using Chinese in Python 3

Signal Processing

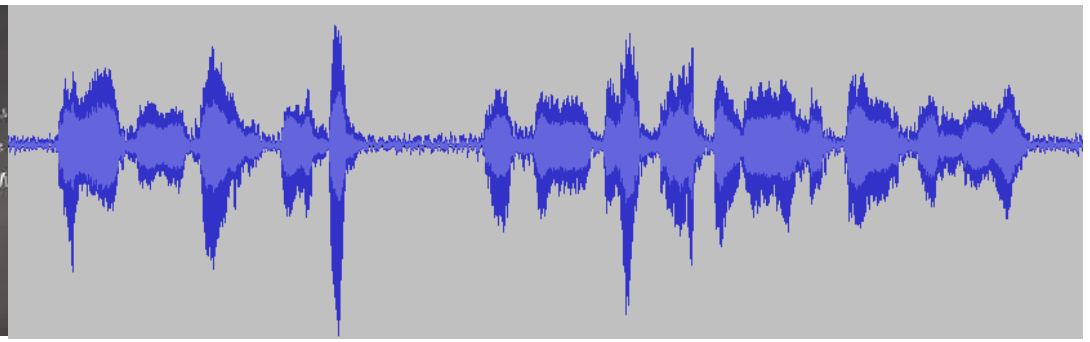
- Signal Processing deals with ***operations*** on or ***analysis*** of ***analog*** or ***digital*** signals, representing **time** varying or **spatially** varying physical quantities, like ***sound***, ***image*** or ***video***.



http://upload.wikimedia.org/wikipedia/commons/4/46/Signal_processing_system.png

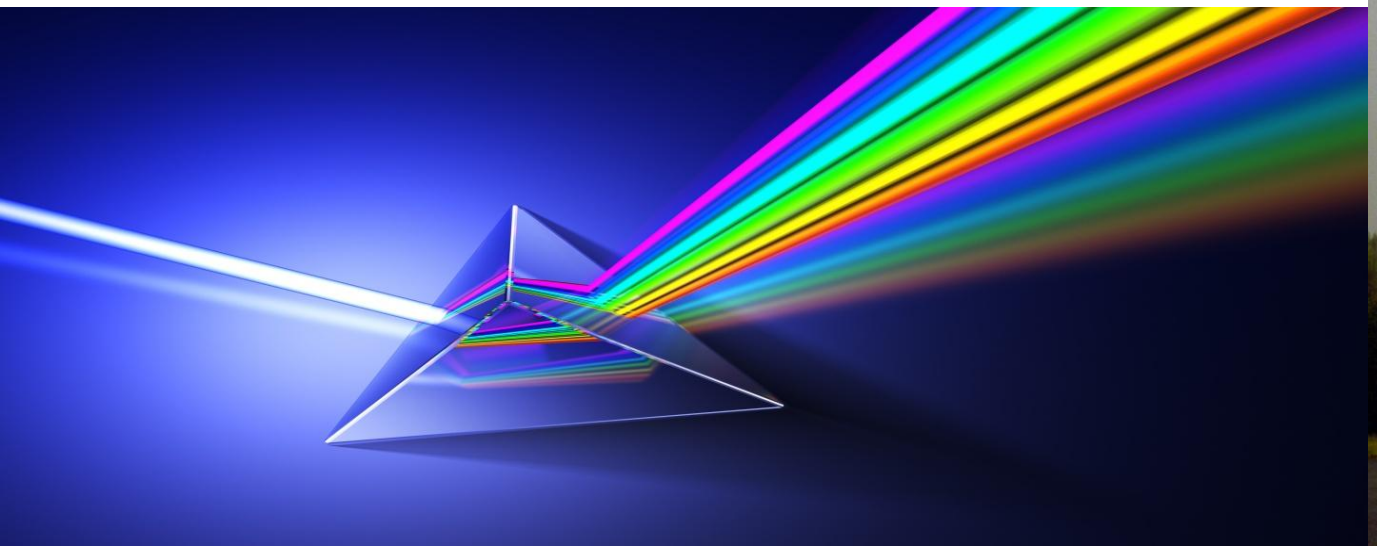
Speech

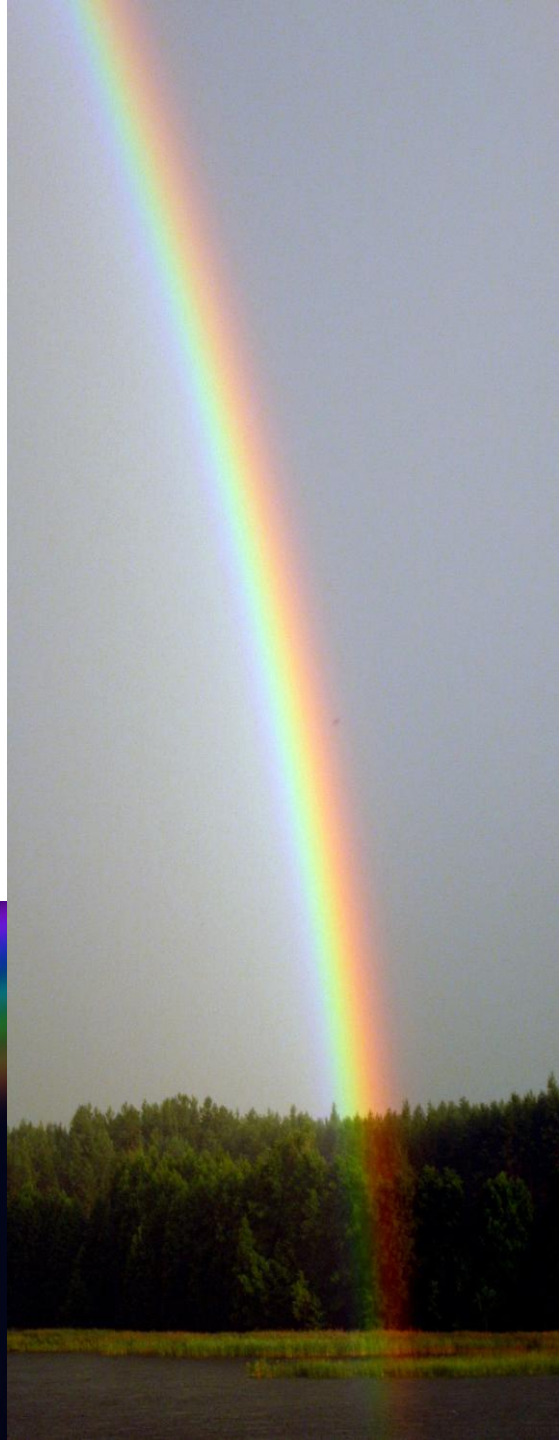
- **Speech** is a 1-dimentional signal
  - a subclass of **audio signal**
    - a representation of sound, typically as an **electrical voltage**
    - with frequencies in the **audio frequency range**
      - roughly 20 to 20,000 Hz (the limits of human hearing)
  - the vocalized form of human language
    - carrying linguistic information
      - the frequency range within 8,000 Hz is enough



(Optical) Spectrum

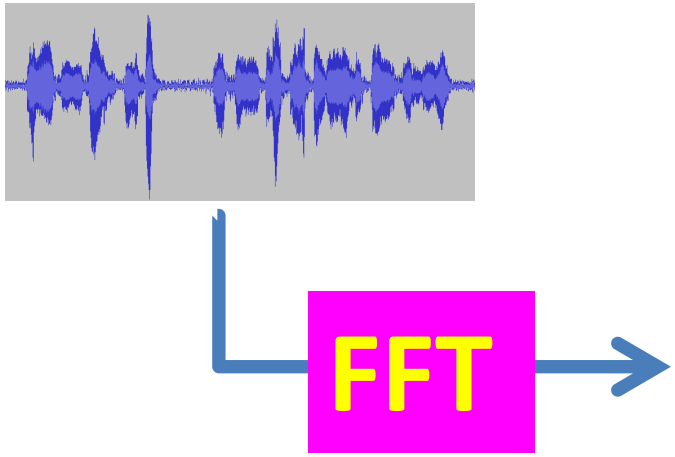
- The word ***spectrum*** was first used scientifically within the field of optics
  - to describe the ***rainbow of colors*** in visible light
    - when ***separated*** using a ***prism***.

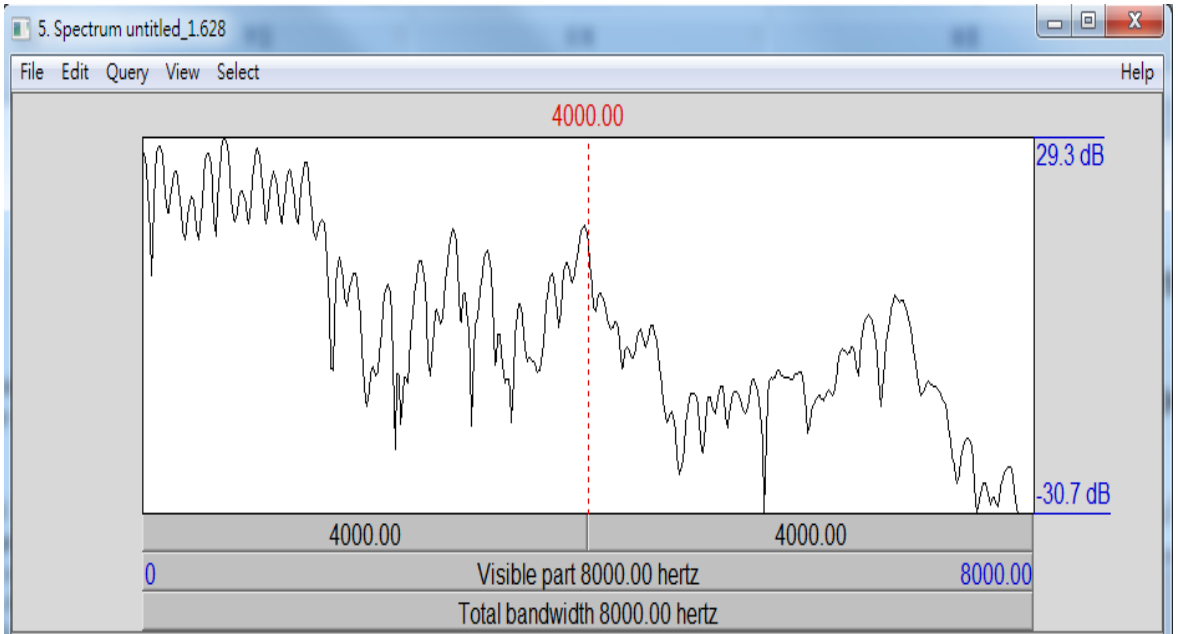




Audio/Speech Spectrum

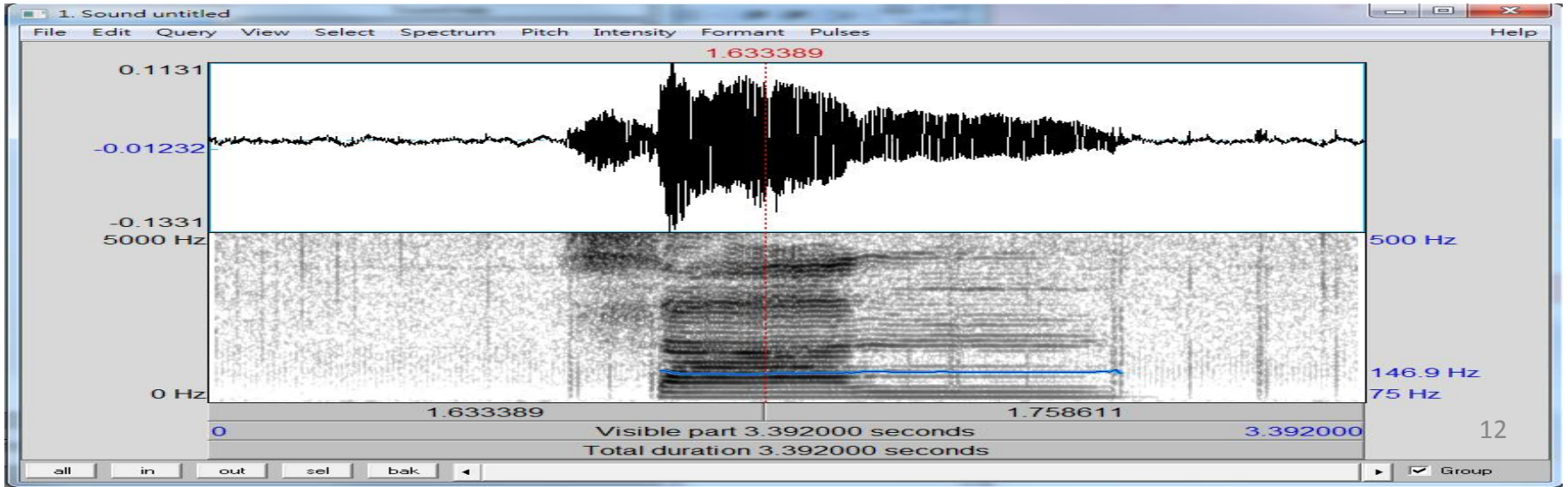
- ***Spectrum*** can be also obtained from audio/speech signal,
  - where it represents the ***frequency distribution*** of the signal.
- ***Fast Fourier Transform*** (***FFT***)
  - the core algorithm to get such a spectrum.





Spectrogram

- Speech as a **time-varying** signal
  - **short-time FFT** is applied in the spectral analysis to form a time-frequency ***spectrogram***
    - Typically the short-time frame is about **20 ms long**.
- Free ***analysis tools*** for speech processing
  - ***Audacity***, ***Praat***, ..etc
  - Perfect for ***off-line***, ***non-real-time*** processing

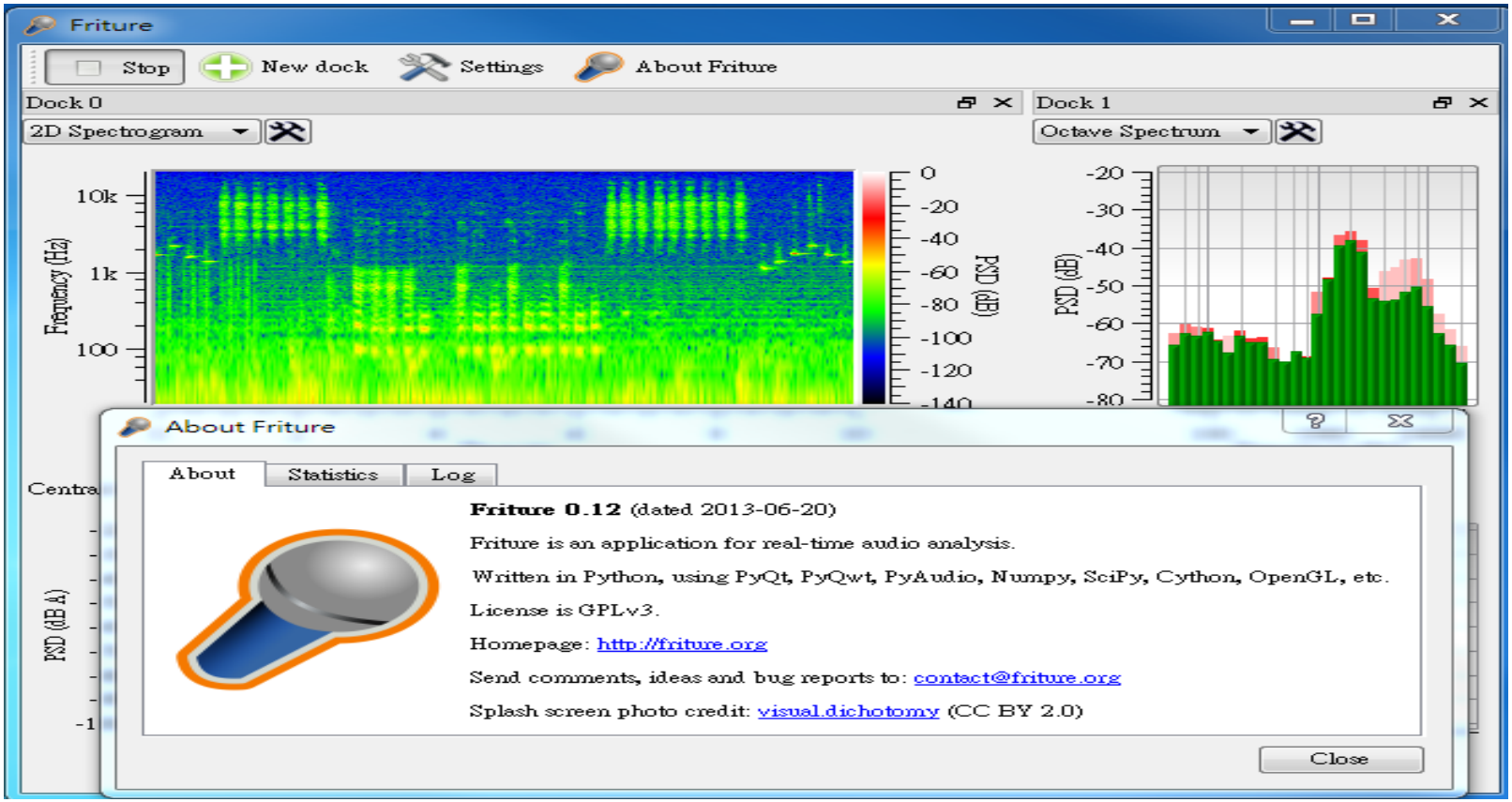


Processing in Real-time

- Real-time processing
  - Acquiring, processing, responding simultaneously
- **An example: Friture**
  - *A Python application to visualize and analyze live audio data in real-time.*
  - *importing PyQt, PyQwt, PyAudio, Numpy, Scipy, Cython, OpenGL, etc,..*
  - http://friture.org/

An Awesome Example: Friture

- I found this app in 2011.
- It was implemented in Python.
  - this is one of the reasons why I was attracted into Python's world



- Comments on Friture:
  - ***Cool, Splendid, Wonderful, Awesome***!!
  - But,
    - Importing too many modules
      - *PyQt, PyQwt, PyAudio, Numpy, Scipy, Cython, OpenGL, etc,..*
    - Only in Python-2, Not yet in Python-3
      - I have ONLY Python-3 environment installed
    - Too complicated for me as a newbie to follow
      - ***The Zen of Python***
        - » **Simple is better than complex.**
        - » **Complex is better than complicated.**

A smaller dependent set

- A smaller dependent module set to implement a real-time spectrogram

  - PyAudio
    - For acquiring speech
      - http://people.csail.mit.edu/hubert/pyaudio/

  - Pylab
    - For DSP (FFT, etc)
      - http://wiki.scipy.org/PyLab

  - PyGame
    - For displaying and GUI
      - http://www.pygame.org/news.html

RyAudio

~ a **class** to deal with audio processing

ryAudio.py

```
'''
ryAudio.py
==========
呂仁園
------
2014/04/20
----------
```

Source Code can be found here
https://gist.github.com/renyuanL/f9cb017a3a5b6c621b43

```
#
# Python standard modules
#
import math
import time
import threading as th
```

```
#
# PyAudio
#
import pyaudio      as pa

#
# Pylab
#(Numpy, Scipy, Matplotlib)
#
import scipy.signal
import pylab        as pl

```

```
#
# my major audio class
#
class RyAudio:
    '''
    This class help you
    to get audio signal from microphone
    and do some simple processing in real time.

    class name | aliase
    RyAudio | 音類

    e.g.

    >>> anAudio= RyAudio()
    >>> anAudio.start()
    >>> # put your code here after
    >>> #time.sleep(10)
    >>> anAudio.x
    >>> anAudio.en
    >>> anAudio.f0
    >>> anAudio.xBuf
    >>> anAudio.specgram
    >>> # finally, you should stop it
    >>> anAudio.stop()
    >>> #你也可以用中文變數名稱
    >>> 音= 音類()
    >>> 音.開始()
    >>> # 把程式碼放在底下。
    >>> # time.sleep(10)
    >>> 音.x # 振幅
    >>> 音.en # 能量
    >>> 音.f0 # 基頻, 基本頻率
    >>> 音.xBuf # 振幅暫存區, 一段暫存區的振幅
    >>> 音.specgram # 頻譜暫存區, 一段暫存區的頻譜
    >>> #最後，你要記得「停止」它
    >>> 音.停止()

    '''
```

```
 79
 80      def __init__(self, Fs=16000, TinSec=10):
127
128      def getSound(self):
268
269      def playSound(self):
317
318      def startGet(self):
329
330      def startPlay(self):
341
342      def start(self):
351
352      def stop(self):
382
383      #
384      # 建立 中文函數別名。
385      #
386
387      開始收音= 開始錄音=   startGet
388      開始放音=            startPlay
389      開始=                start
390      結束= 停止=          stop
391      pass
392
```

```
392  #
393  # 建立 中文物類別名。
394  #
395  音類= RyAudio
396
397  #
398  # 本模組到此結束。
399  #
400
```

```
405 def demo00():
406     '''
407     展示最簡單的多線錄放音功能。
408     '''
409     #
410     # 首先，要啟動音訊裝置。
411     #
412
413     音= 音類()
414     音.開始()
415
416     print('主線 睡 10 秒，音線 錄放音 10 秒。')
417
418     time.sleep(10) # 主線 睡 10 秒，音線 進入 錄放音 狀態
419
420     print('主線 醒來，音線 即將結束。')
421
422     音.結束()
```

```
424 def demo01():
425     '''
426     錄放音後，把音訊 en, f0 用 pylab 畫出來。
427     '''
428     #
429     # 首先，要啟動音訊裝置。
430     #
431     音= 音類()  # must (1)
432     音.開始()
433
434     #
435     # 音 啟動之後 如何抓住？
436     # 請看以下範例。 抓 T= 10 秒
437     #
438     T= 10 # sec
439     t0= time.time()
440     n= 0
441     音訊= []
```

```
442    while time.time()-t0< T:
443
444        t= 音.t    # 時間 (sec)
445        en= 音.en  # 能量 (energy, en)
446        f0= 音.f0  # 基頻 (fudamental frequency, f0)
447        音訊+= [(n, t, en, f0)]
448
449        # control sampling period,
450        # 0.01 sec for en and f0, that is enough
451        time.sleep(0.01)
452
453    # collecting the audio info to plot
454    tL= [t  for (n, t, en, f0) in 音訊]
455    eL= [en  for (n, t, en, f0) in 音訊]
456    fL= [f0  for (n, t, en, f0) in 音訊]
457
458    # Using Pylab to plot it
459    pl.subplot(211); pl.plot(tL, eL)
460    pl.subplot(212); pl.plot(tL, fL)
461    pl.show()
462
463    #
464    # 程式結束前要記得把 音 停止。
465    #
466    音.停止()
```

```
491  import turtle as tt
492
493  def demo03():
494      '''
495      利用本模組來聲控小烏龜。
496      '''
497
498      ###
499      ###    step1 to use RyAudio,
500      ###    generate it
501      ###
502      音= 音類()
503
504      ###
505      ###    step2 to use RyAudio,
506      ###    start it
507      ###
508      音.開始()
509
```

```
510     #
511     # 中文函數別名
512     #
513     時間=    time.time
514     開根號= math.sqrt
515     對數=    math.log
516     較小值= min
517
518     #
519     # turtle module
520     # set width and height of the screen
521     #
522     幕= tt.Screen()
523     龜= tt.Turtle()
524     W= H= 100
525     幕.setworldcoordinates (0, 0, W, H)
526     龜.penup()
527
528     #
529     # set time buffer, 10 sec is a good choice
530     #
531     T= 10 # sec
532     aMsg= 'get sound for %d sec, please wait...'%T
533     print(aMsg)
534     龜.write(aMsg)
535
```

```
536     t= t0= 時間()
537     while t - t0< T:
538         t= 時間()
539         x= (t*10)%W
540 
541         ###
542         ###    step3 to use RyAudio,
543         ###    get infomation (en) from it
544         ###
545         y= 音.en
546 
547         if y>0:
548             y= 對數(y)
549         y= 較小值(y, H)
550 
551         龜.goto(x,y)
552         龜.dot()
553 
554     aMsg='click X to close the screen and stop the sound.
555     print(aMsg)
556     龜.color('red')
557     龜.write(aMsg)
558 
559     幕.mainloop()
560 
561     ###
562     ###    step4 to use RyAudio,
563     ###    stop it
564     ###
565     音.結束()
566 
```

ryApp.py

~ an app of realtime spectrogram

ryAudio.py | ryApp.py

```
'''
========
ryApp.py
========
呂仁園
------
2014/04/20

運用 RyAudio.py
的即時語音頻譜。

This program use many Chinese names
for variables, functions and classes

First presentation
on PyCon APAC 2014

```

```
import pygame              as pg
import pygame.camera       as pgCam
from   pygame.locals import *

import pylab               as pl
import colorsys
import time

import ryAudio     as ra
#
# 這個 ryAudio 是我們的私房模組，
# 引入 pyaudio
#
# 專門用來 錄音, 放音,
# 以及簡單的聲音特徵擷取。
#

```

```
70
71  def 頻率轉顏色(頻率, 倍數= 1):
81
82  class 影音類:
83
84      幕寬, 幕高= 幕寬高= size = ( 640, 480 )
85
86      def __init__(它):
97
98      def 啟動視訊(它, 攝影機編號= 0):
135
136     def 取視訊且顯示於幕(它, 鍵盤= None):
181
182     def 啟動音訊(它):
207
208     def 取音訊且顯示頻譜於幕(它, 鍵盤= None):
307
308     def 滑鼠游標顯示音訊能量及頻率(它, 滑鼠x, 滑鼠y, 鍵盤= None):
331
332     def 主迴圈(它):
417
418 if __name__ == '__main__':
419
420     影音類().主迴圈()
421
```

ryAudio.py | ryApp.py

```
331
332     def 主迴圈(它):
333
334         簡單控制方法='''
335         用 K_abcd 來控制視訊處理
336         用 K_efgh 來控制音訊處理
337         用 K_ijk 來控制滑鼠處理
338         '''
339         print('簡單控制方法= ', 簡單控制方法)
340
341         滑鼠按著=          False
342         滑鼠x= 滑鼠y=  0
343         鍵盤=              None
344
345         主迴圈執行中=  True
346
347         while 主迴圈執行中:
410             #
411             # 跳出主迴圈了
412             #
413         print('主迴圈執行中= ', 主迴圈執行中)
414         它.攝影機.stop()
415         它.音.結束()
416         pg.quit()
417
```

ryAudio.py | ryApp.py

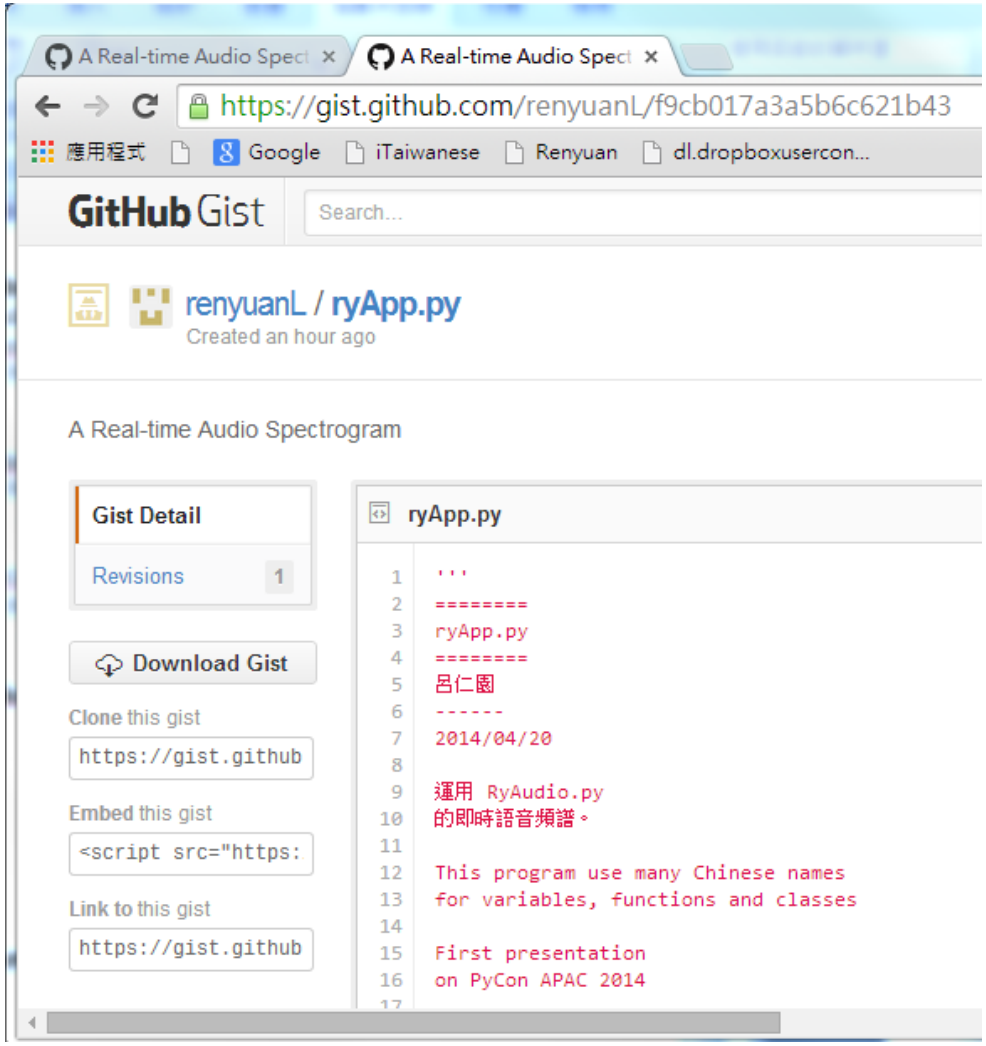
```
347         while 主迴圈執行中:
348             #
349             # 取得 使用者 輸入 事件
350             #
351             事件群= pg.event.get()
352 
353             #
354             # 處理 使用者 輸入 事件
355             #
356             for e in 事件群:
389 
390             #
391             # 視訊
392             #
393             它.取視訊且顯示於幕(鍵盤) # 用 K_abcd 來控制視訊處理
394 
395             #
396             # 音訊
397             #
398             它.取音訊且顯示頻譜於幕(鍵盤) # 用 K_efgh 來控制音訊處理
399 
400             #
401             # 滑鼠
402             #
403             if (滑鼠按著 is True):  # 用 K_ijk 來控制滑鼠處理
405 
406             #
407             # 畫面更新
408             #
409             pg.display.flip()
410         #
```

```
352
353                 #
354                 # 處理 使用者 輸入 事件
355                 #
356                 for e in 事件群:
357
358                     #
359                     # 首先 優先處理 如何結束，優雅的結束！
360                     #
361                     # 用滑鼠點擊 X (在 視窗 右上角) 結束！
362                     #
363                     if e.type in [QUIT]:
364                         主迴圈執行中= False
365
366                     #
367                     # 用鍵盤 按 Esc (在 鍵盤 左上角) 結束！
368                     #
369                     if e.type in [KEYDOWN]:
370                         鍵盤= e.key
371                         if e.key in [K_ESCAPE]:
372                             主迴圈執行中= False
373                     if e.type in [KEYUP]:
374                         鍵盤= None
```

```
375                     #
376                     # 以下 3 個 if , 用來 處理 滑鼠
377                     #
378                     if e.type in [MOUSEBUTTONDOWN]:
379                         滑鼠按著= True
380                         滑鼠x, 滑鼠y= x,y= e.pos
381
382                     if e.type in [MOUSEBUTTONUP]:
383                         滑鼠按著= False
384                         滑鼠x, 滑鼠y= x,y= e.pos
385
386                     if e.type in [MOUSEMOTION]:
387                         if (滑鼠按著 is True):
388                             滑鼠x, 滑鼠y= x,y= e.pos
389
390                 #
```

```
70
71  def 頻率轉顏色(頻率, 倍數= 1):
81
82  class 影音類:
83
84      幕寬, 幕高= 幕寬高= size = ( 640, 480 )
85
86      def __init__(它):
97
98      def 啟動視訊(它, 攝影機編號= 0):
135
136     def 取視訊且顯示於幕(它, 鍵盤= None):
181
182     def 啟動音訊(它):
207
208     def 取音訊且顯示頻譜於幕(它, 鍵盤= None):
307
308     def 滑鼠游標顯示音訊能量及頻率(它, 滑鼠x, 滑鼠y, 鍵盤= None):
331
332     def 主迴圈(它):
417
418 if __name__ == '__main__':
419
420     影音類().主迴圈()
421
```

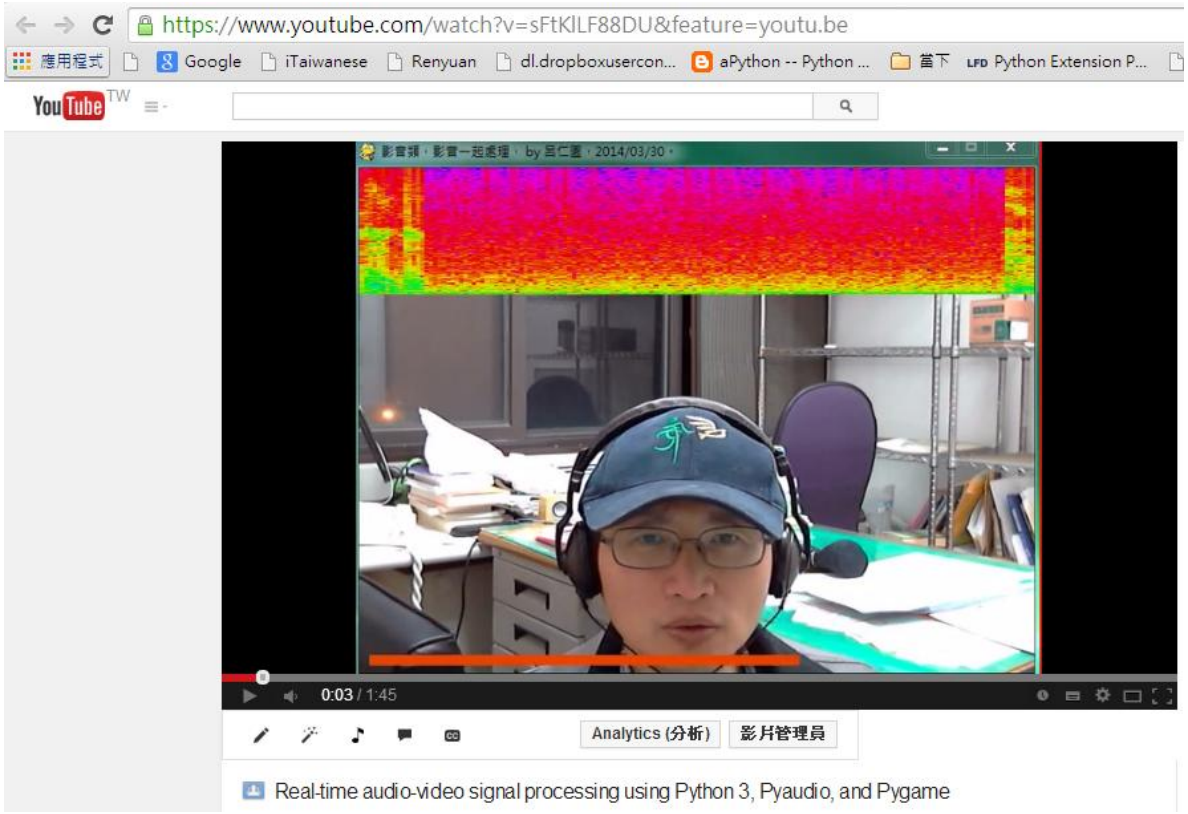
- Source Code can be found here
  - https://gist.github.com/renyuanL/f9cb017a3a5b6c621b43



Demo

ryApp.py - 捷徑.l

- Demo in Youtube
  - http://youtu.be/sFtKlLF88DU



Some Comments on Programming in Native Languages

設計程式對非英語為母語的人來說，(特別是小孩)
允許其運用 其「母語」來「寫」，
比較有可能「登門入室」甚至「文思泉湧」，
因此能迅速產生「內容」。

等到「內容」大致底定，
為了與全球人士分享智慧的結晶，
需要將這種「母語」程式轉成「英語」程式，
以利全球範圍的流通。

以寫文章來做類比，

金庸 要用 中文 才寫得出 「神鵰俠侶」，
Mark Twain 要用 English 才寫得出 'Tom Sawyer'

一旦作品優秀、揚名了，自然有轉成其他語言與更多人分享的需求

Python Code Translation

ryApp.py → ryApp_en.py

ryAudio.py | ryApp.py | ryApp_en.py

```
'''
========
ryApp.py
========
呂仁園
------
2014/04/20

運用 RyAudio.py
的即時語音頻譜。

This program use many Chinese names
for variables, functions and classes

First presentation
on PyCon APAC 2014
```

ryAudio.py | ryApp.py | ryApp_en.py

```
'''
=========================================
ryApp_en.py  <--  translated <-- ryApp.py
=========================================
Renyuan Lyu, 呂仁園
-------------------
2014/04/21

importing RyAudio.py
Realtime-audio spectrogram。

This program was originally written using many Chinese names
for variables, functions and classes.

For international audience,
it is translated into English.

Note that all Chinese names in the code was translated
by the original author himself;

however, the Chinese comments are translated into English (in parentheses)
by the Google translator,
just for international audience's reference,

I wish some English native speaker can help me to polish it.

This program was first presented on PyCon APAC 2014, Taipei, Taiwan.
```

ryApp.py → ryApp_en.py

- https://gist.github.com/renyuanL/f9cb017a3a5b6c621b43

ryAudio.py | ryApp.py | ryApp_en.py

```
70
71  def 頻率轉顏色(頻率, 倍數= 1):
81
82  class 影音類:
83
84      幕寬, 幕高= 幕寬高= size = ( 640, 480 )
85
86      def __init__(它):
97
98      def 啟動視訊(它, 攝影機編號= 0):
135
136     def 取視訊且顯示於幕(它, 鍵盤= None):
181
182     def 啟動音訊(它):
207
208     def 取音訊且顯示頻譜於幕(它, 鍵盤= None):
307
308     def 滑鼠游標顯示音訊能量及頻率(它, 滑鼠x, 滑鼠y, 鍵盤= None
331
332     def 主迴圈(它):
417
418 if __name__ == '__main__':
419
420     影音類().主迴圈()
421
```

ryAudio.py | ryApp.py | ryApp_en.py

```
80
81  def frequency2color(frequency, scale= 1):
91
92  class VideoAudio:
93
94      screenWidth, screenHeigth= screenSize= size = ( 640, 480 )
95
96      def __init__(self):
107
108     def initVideo(self, cameraIndex= 0):
145
146     def takeVideoAndDisplay(self, keyboard= None):
191
192     def initAudio(self):
218
219     def takeAudioAndDisplay(self, keyboard= None):
323
324     def mouseShowEnAndF0(self, mouseX, mouseY, keyboard= None):
347
348     def mainLoop(self):
433
434 if __name__ == '__main__':
435
436     VideoAudio().mainLoop()
437
```

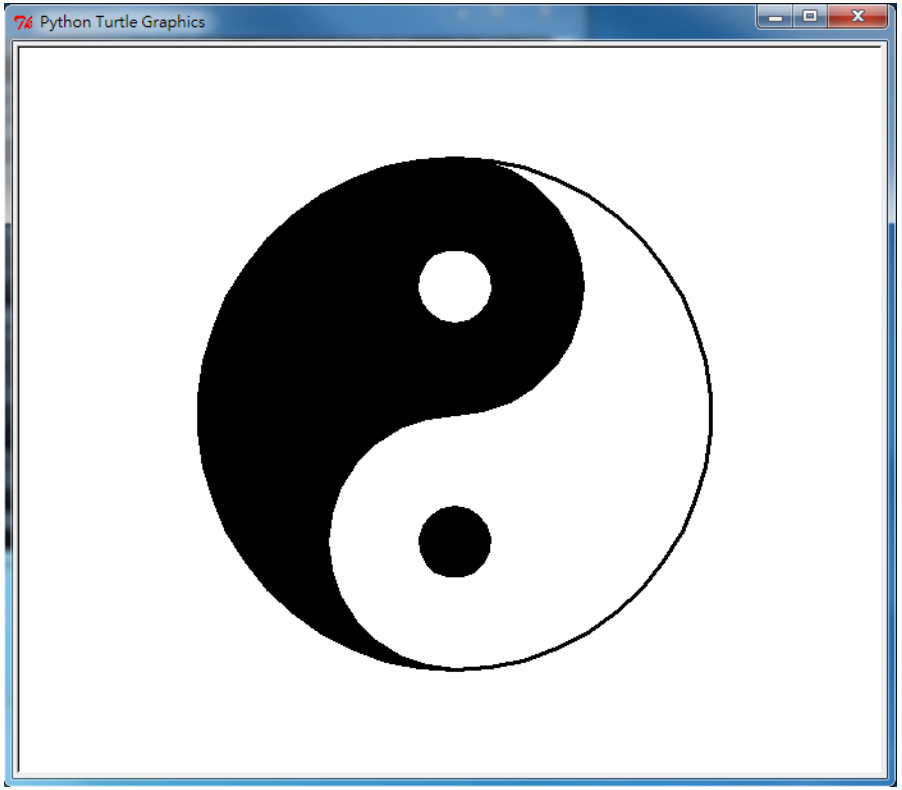
Examples of Chinese Programs

- http://apython.blogspot.tw/

- A set of Chinese Programs in Python 3
  - https://gist.github.com/renyuanL/044b6bc6142dc71086bc
  - https://gist.github.com/renyuanL/a36f2a121c4d27753d8c

陰陽　Yinyang

- C:\Python33\Lib\turtledemo\yinyang.py



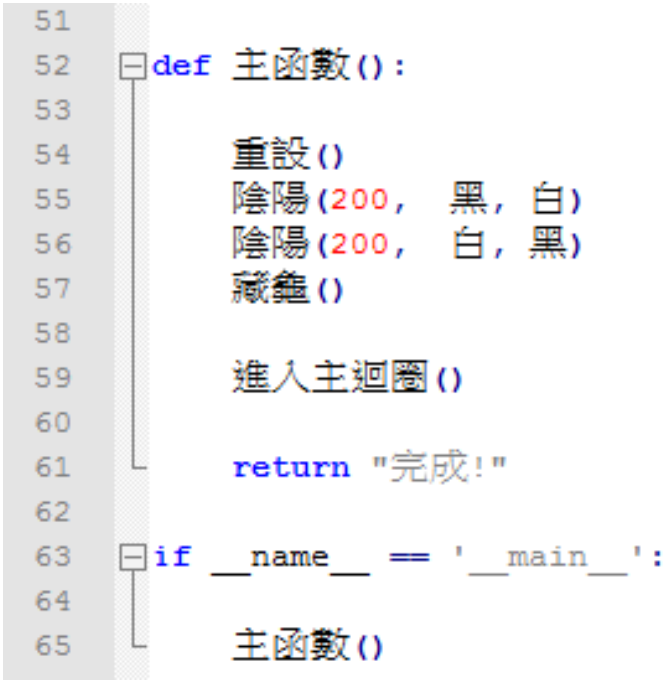
Coding in your own native language

ryYinyang.py

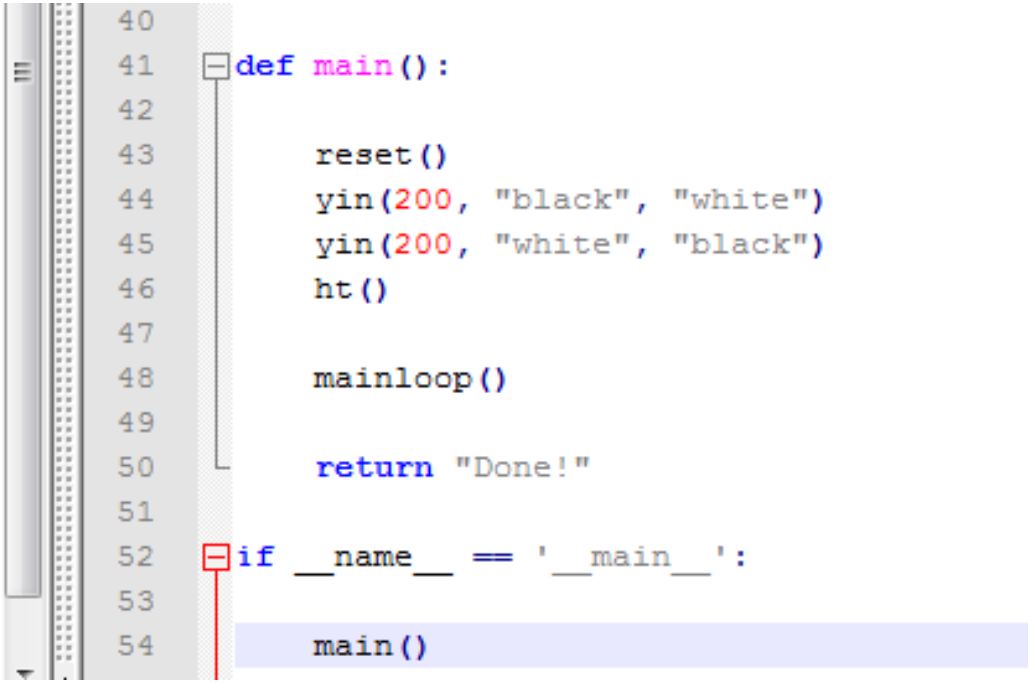
```
25  from  turtle_tc import *
26
27  def 陰陽(半徑, 色01, 色02):
28
29      筆大小(3)
30      顏色(黑, 色01)
31      開始填色()
32      畫圓(半徑/2, 180)
33      畫圓(半徑, 180)
34      左轉(180)
35      畫圓(-半徑/2, 180)
36      結束填色()
37      左轉(90)
38      提筆()
39      前進(半徑*0.35)
40      右轉(90)
41      下筆()
42      顏色(色01, 色02)
43      開始填色()
44      畫圓(半徑*0.15)
45      結束填色()
46      左轉(90)
47      提筆()
48      後退(半徑*0.35)
49      下筆()
50      左轉(90)
```

yinyang.py

```
14  from  turtle import *
15
16  def yin(radius, color1, color2):
17
18      width(3)
19      color("black", color1)
20      begin_fill()
21      circle(radius/2., 180)
22      circle(radius, 180)
23      left(180)
24      circle(-radius/2., 180)
25      end_fill()
26      left(90)
27      up()
28      forward(radius*0.35)
29      right(90)
30      down()
31      color(color1, color2)
32      begin_fill()
33      circle(radius*0.15)
34      end_fill()
35      left(90)
36      up()
37      backward(radius*0.35)
38      down()
39      left(90)
```



```
51
52  def 主函數():
53
54      重設()
55      陰陽(200,  黑, 白)
56      陰陽(200,  白, 黑)
57      藏龜()
58
59      進入主迴圈()
60
61      return "完成!"
62
63  if __name__ == '__main__':
64
65      主函數()
```



```
40
41  def main():
42
43      reset()
44      yin(200, "black", "white")
45      yin(200, "white", "black")
46      ht()
47
48      mainloop()
49
50      return "Done!"
51
52  if __name__ == '__main__':
53
54      main()
```

- If readability counts, then it will achieve maximum when coding in your own native language.

Thank you
for Listening