Machine Translation
of English Identifiers
in **Python Programs**
into Traditional Chinese

Ren-yuan Lyu (呂仁園), Yung-Hsin Kuo, Che-Ning Liu

Computer Science and Information Engineering

Chang Gung University (長庚大學), Taoyuan, Taiwan

投影片 在此： http://goo.gl/nHY16i

- Software Framework and Program Analysis
- Date: 10:15-12:00 Friday 16 Dec. 2016
- Chair: Nien-Lin Hsueh, (薛念林) Feng Chia University
- #36 : Machine Translation of English Identifiers in Python Programs into Traditional Chinese
- Ren-yuan Lyu , Yung-Hsin Kuo, Che-Ning Liu

Computer Programming for everybody!

https://www.python.org/doc/essays/cp4e/

By **Guido van Rossum**

老若男女と
プログラミング

Picture from Pycon JP 2015

**No matter Old or Young
No matter Boys or Girls**

3

**Computer Programming
for everybody?**

Really?

**As long as she/he knows
the English language!**

Is it possible
that the English language
is the first **barrier** for people
to learn programming?

We usually get the fundamental education
in native language or school language,
which is usually not English!

Python Turtle Graphics
# in Traditional Chinese (Kanji)

傳統漢字(繁體中文)の
Python 龜作圖

First appeared on **Pycon APAC 2015**

Review of Presentation on Python APAC 2015

- A set of 18 turtle demo programs has been **translated** into traditional Chinese (tc)

  - as an example to show the possibility to write Python code conveniently in non-English language.

- To **improve code clarity** and **readability**

  - for non-native English speakers, according to **Python PEP 3131**.

- Providing a full list of **tc alias (turtle_tc.py)**

  - for the official **python turtle** module.

- Availability in **Github**

  http://github.com/renyuanL/**pythonTurtleInChinese**

A quick Glance of Demonstration
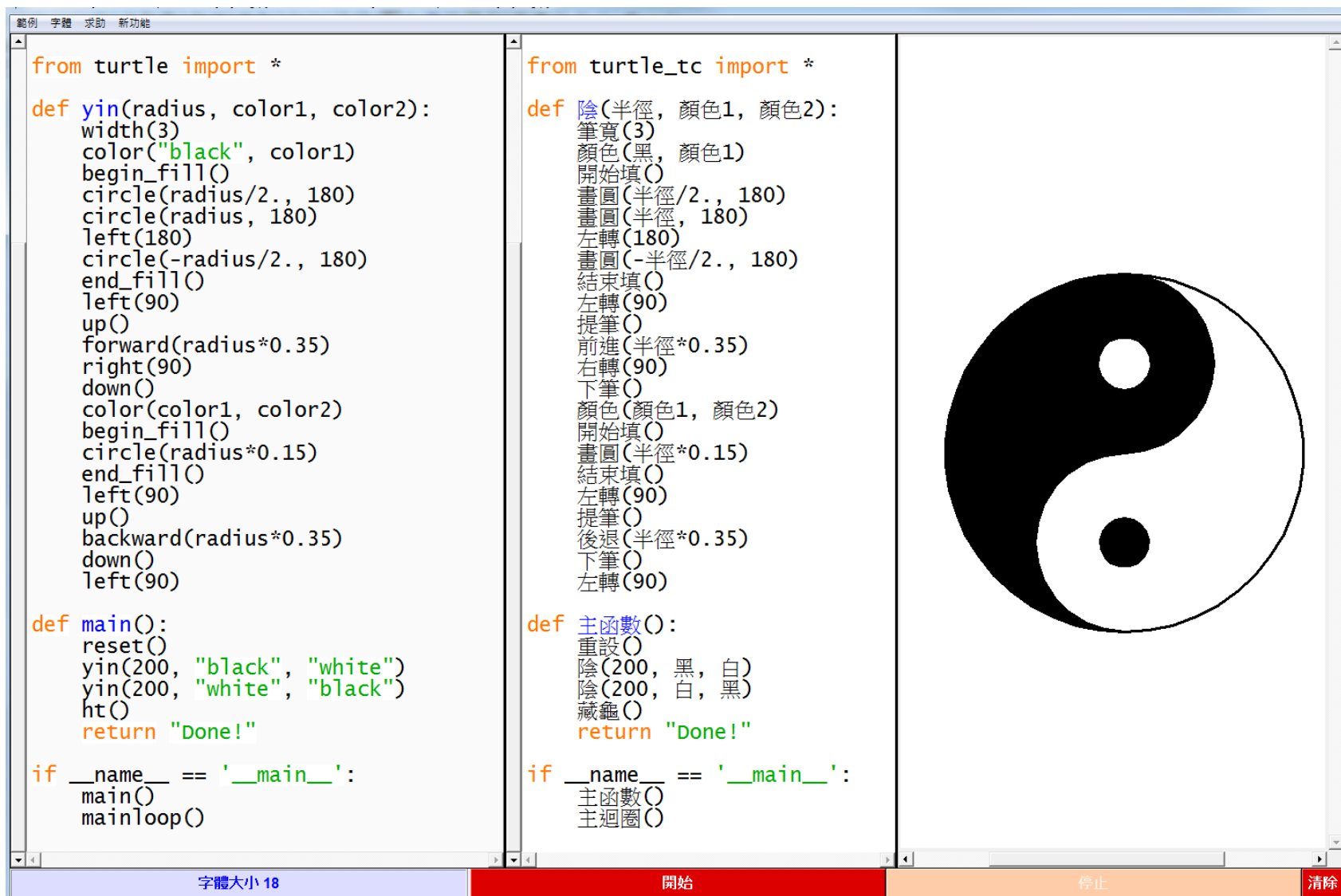
The motivation was partially from PEP 3131: "Supporting Non-ASCII Identifiers"

- many people in the world **not familiar with** the **English** language
  - *They'd like to define variables, functions and classes with names in their native languages*
- *code clarity and maintainability* of the code **among speakers of that language** improves.

- Original from **PEP 3131** : [https://www.python.org/dev/peps/pep-3131/]

Ignition:
UTF-8 as Source encoding

- After version **3.0**, the Python language has changed its source coding **from ASCII to UNICODE (UTF-8)**

- This is quite significant because it will be possible that non-English characters can be used as identifiers, which contain names of variables, functions, classes and methods. Here are examples:

```
>>> 甲 =   100
>>> 某數 = 甲 - 10
```

```
>>> 印=    print
>>> 範圍= range

>>> 某字串=      '你好，世界。'
>>> 重複的次數= 10
>>> for 數 in 範圍(重複的次數):
       印(某字串，數)
```

Readability counts

- Python's **Zen** (禪)

```
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
```
Readability counts.
```
...
...
Namespaces are one honking great idea
-- let's do more of those!
(19 sentences)
>>>
```
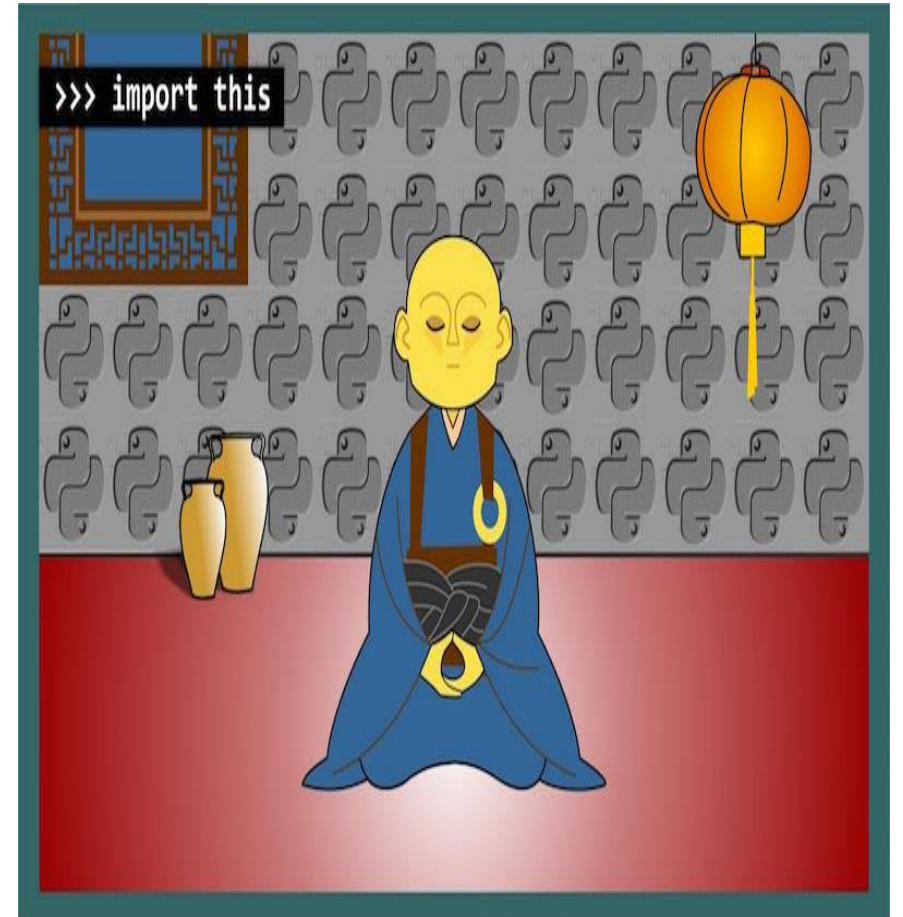
pic comes from:
http://www.itsmycodeblog.com/the-zen-of-python/

If **Readability really counts**,...

- Then, what can be more readable to write programs in your own native language, if the readers are those who use the same language,

  including ourselves, who read our own programs more frequently than the others**!**

Python Module for Turtle Graphics

- **Turtle graphics** is a term in **computer graphics**
    - [http://en.wikipedia.org/wiki/Turtle_graphics]
  - part of the original **Logo** programming language
  - by Wally Feurzig and Seymour Papert in **1966**.

- The **Python Turtle** module is an extended reimplementation
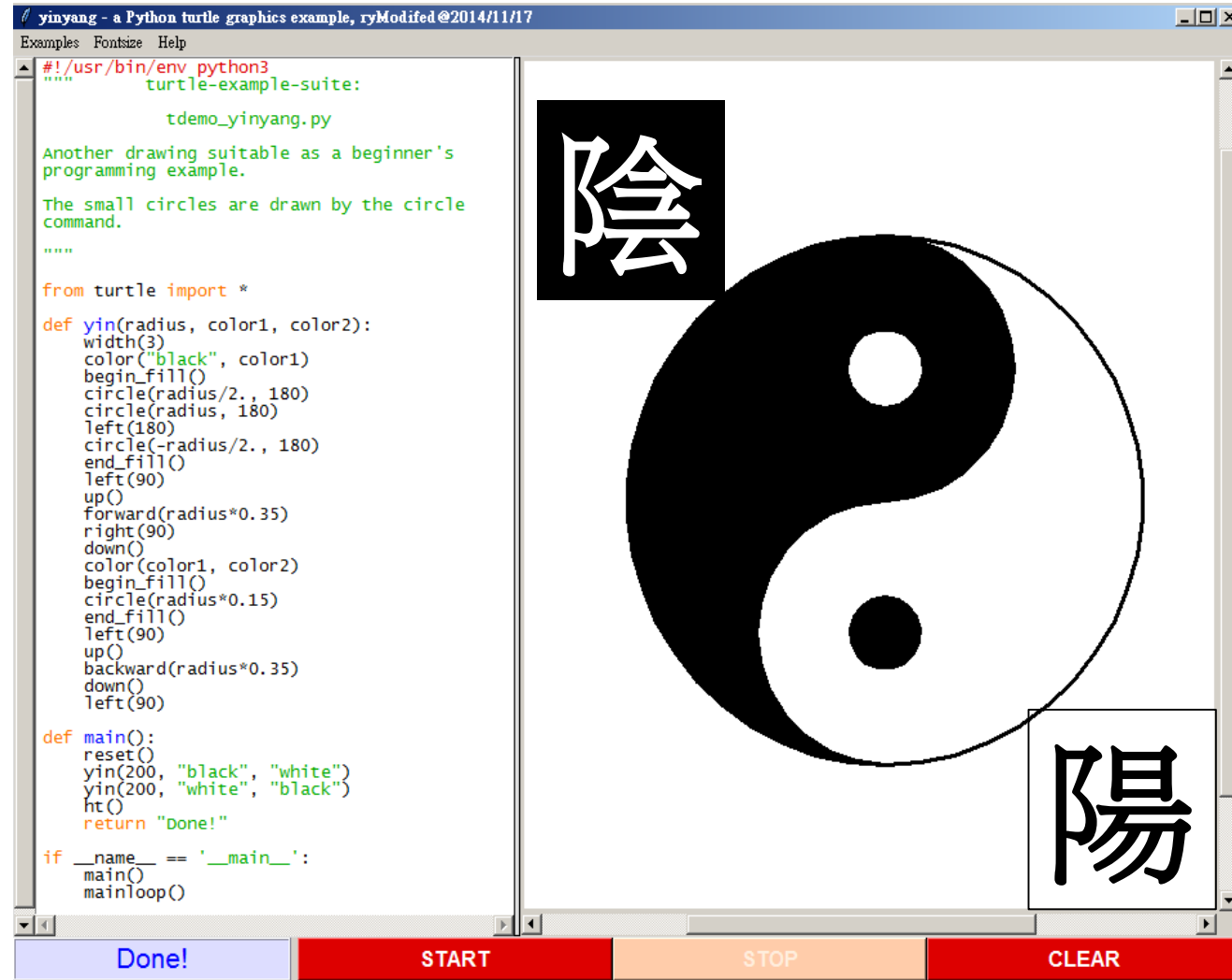   - from the **Python standard distribution** since Python 2.5.

Turtle Demo in IDLE Shell

- Starting from Python 3.4.2, a set of 18 turtle demo programs was promoted to appear in the main menu of IDLE Shell, just below Python Docs within the **Help** sub-memu.
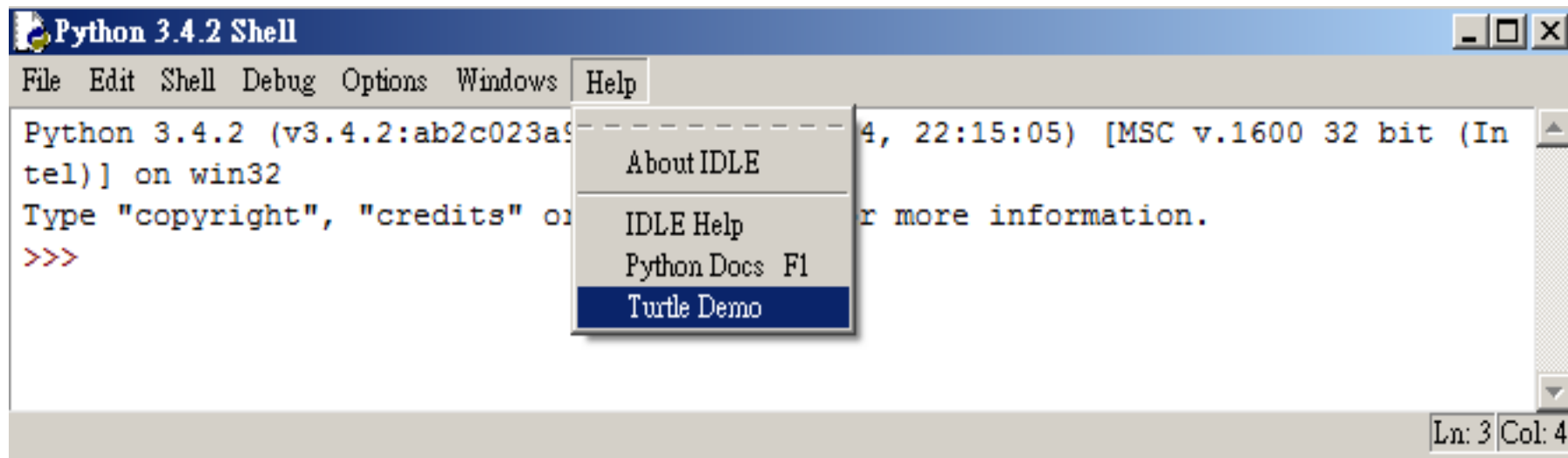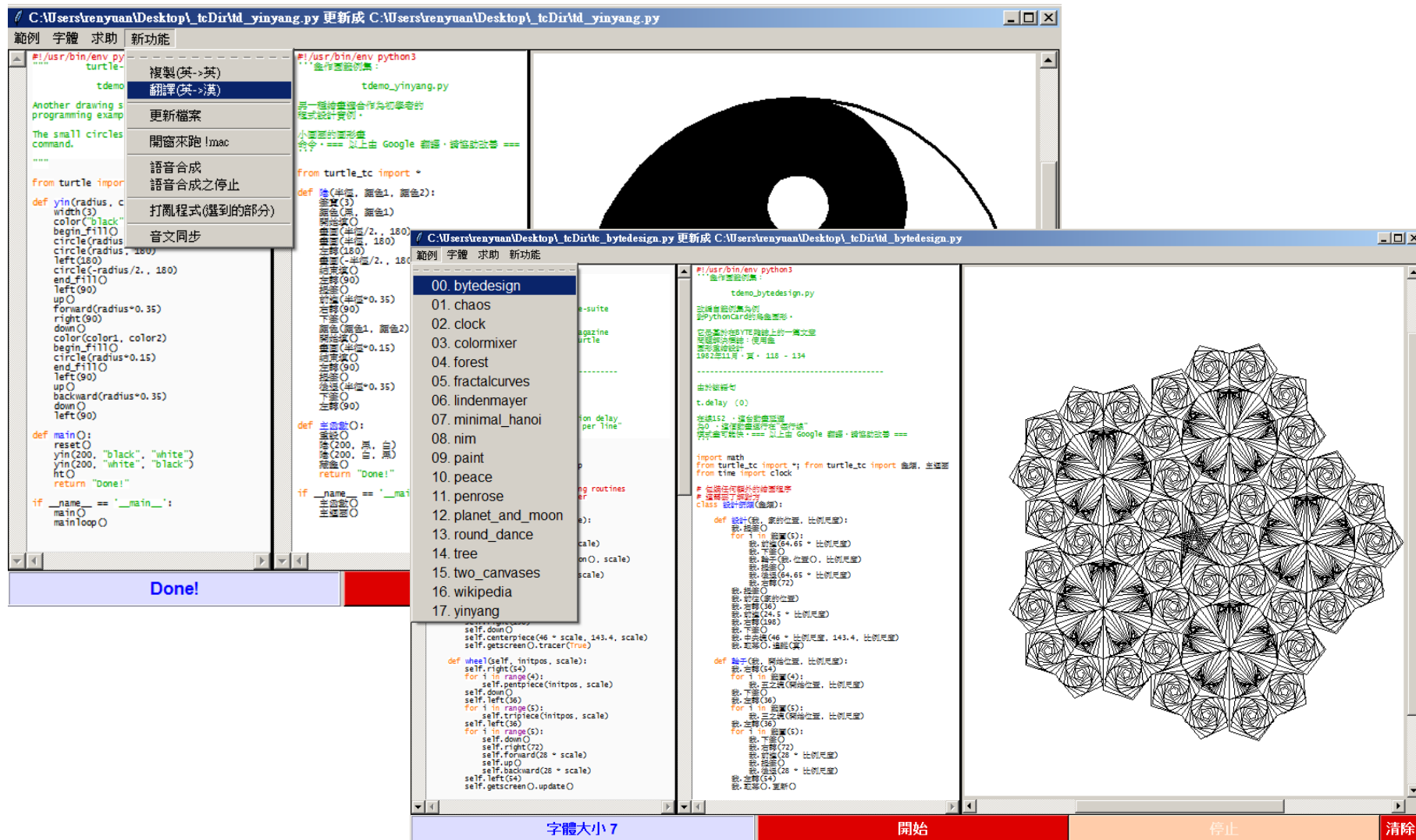
A typical example

- An example from the set of turtle demo programs: **yinyang.py**

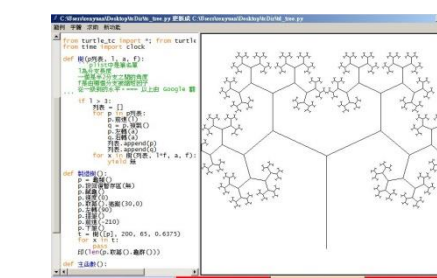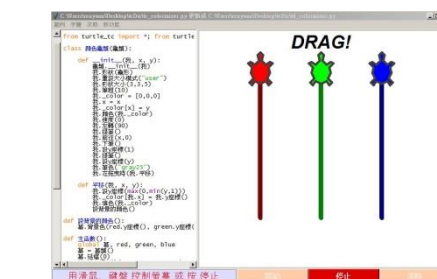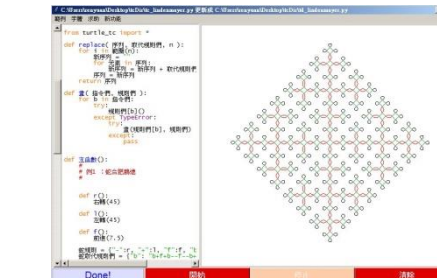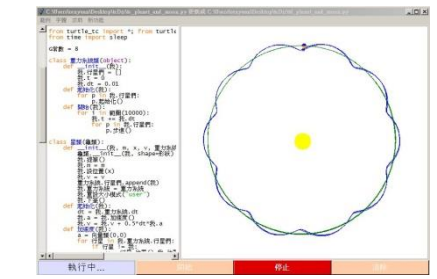# The whole set
of 18 Turtle Demo programs

18 programs
of Turtle Graphics
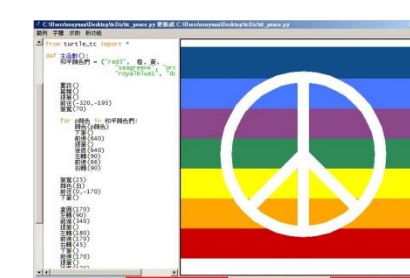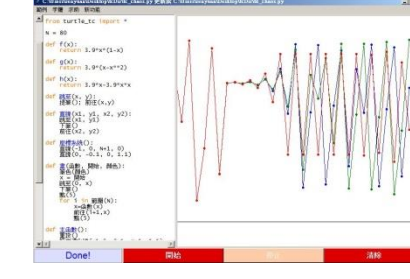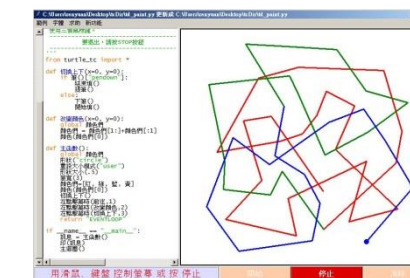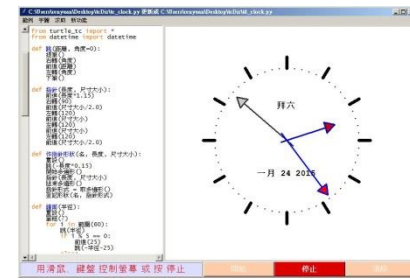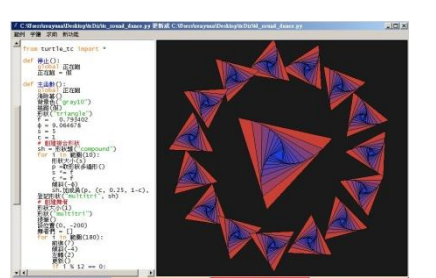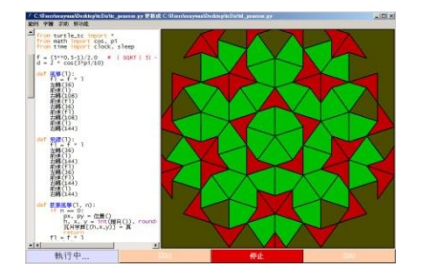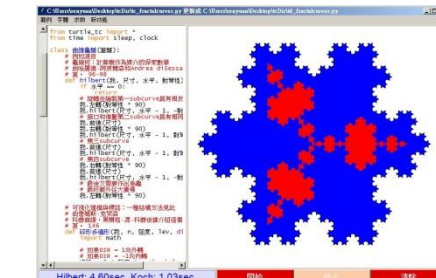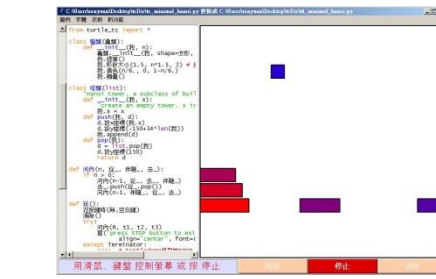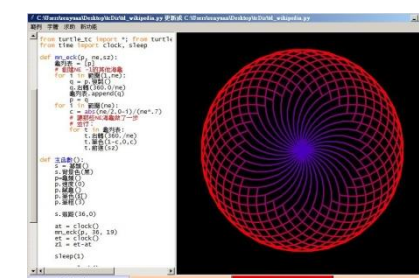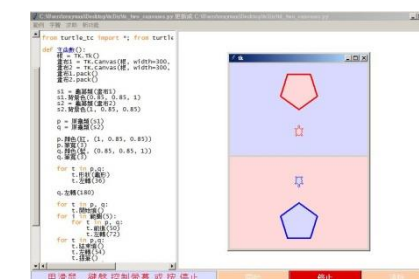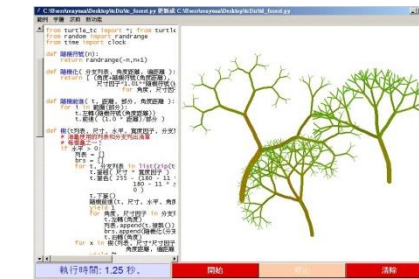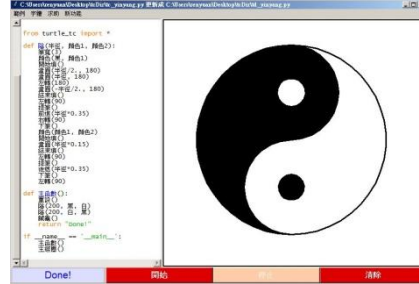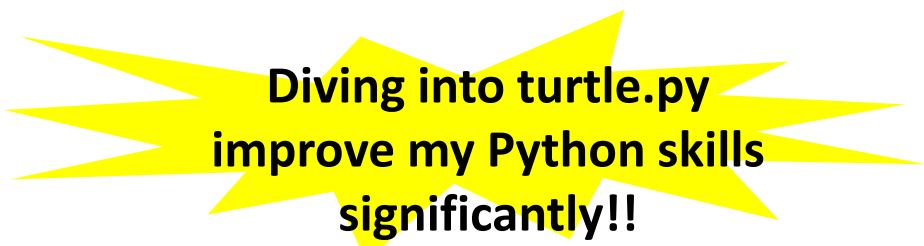
Inside the turtle module (**turtle.py**)

- The **class diagram** of the turtle module
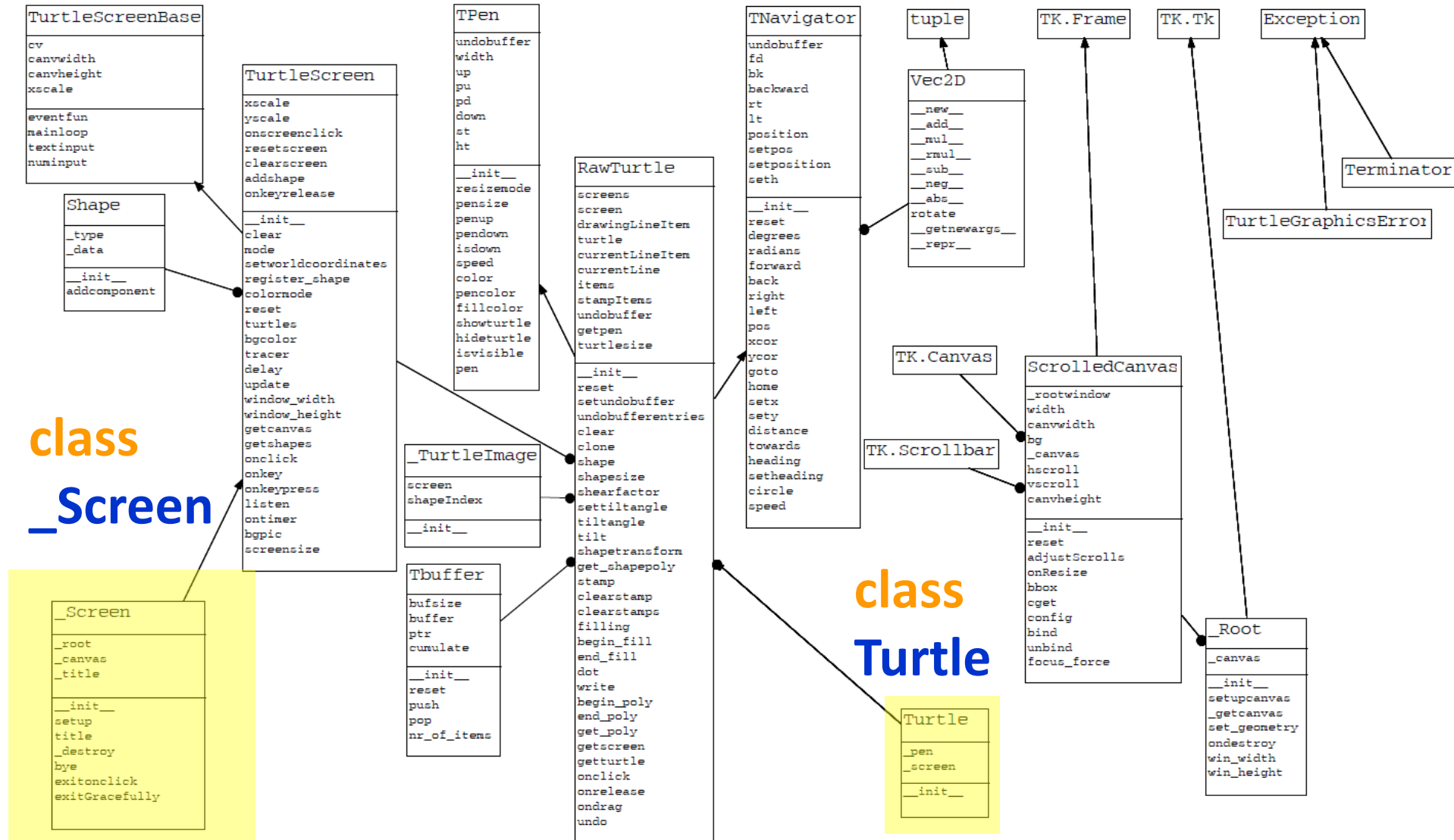
Summary of the turtle module

- File path (@ Windows)   **C:\Python34\Lib\turtle.py**
- Number of lines in source code
  - **About 4000 lines**
  - Rank 2 out of 160 python files in the standard library (Python 3.4.2)
- **2 major classes**
  - With the other 8 supporting classes
  - **class Turtle**
    - 80 methods
    - E.g., **forward , backward , left , right , ...**
  - **class _Screen**
    - 34 methods
    - E.g., **addshape, bgcolor, bgpic, clearscreen, ...**
- **112 Top-level functions**
  - All methods from class Turtle and class _Screen are redefine as the top-level functions with a default turtle and screen objects

Diving into turtle.py improve my Python skills significantly!!

Alias of the turtle module
in Traditional Chinese (**tc**, or **zh-tw**)

- Upon the original turtle module, turtle.py, we create an **associated module** called **turtle_tc.py**, which provides the alias in **traditional Chinese** (thus the subscript "**_tc**") for almost all identifiers (names) in turtle.py

DownLoad @ http://github.com/renyuanL/pythonTurtleInChinese
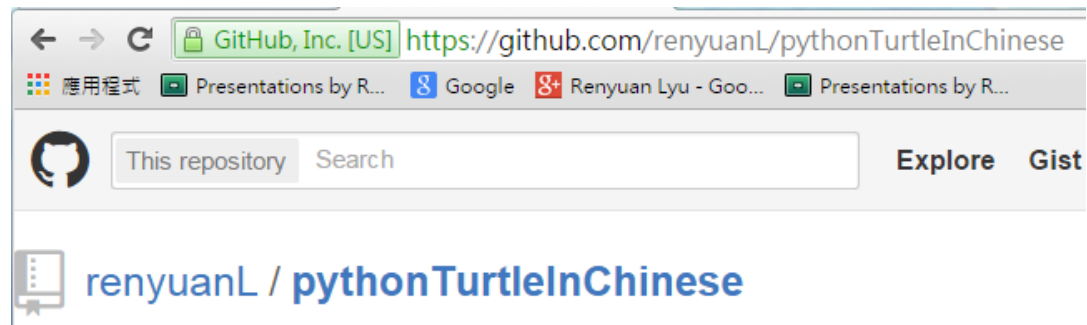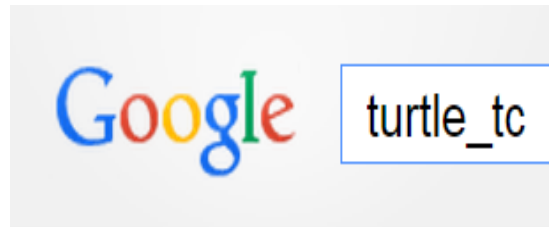
Step 3: after downloading turtle_tc, import it

```
>>> from turtle_tc import *
>>> dir()
```

['Pen', 'RawPen', 'RawTurtle', 'Screen', 'ScrolledCanvas', 'Shape', 'TK', 'Terminator', 'Turtle', 'TurtleScreen', 'Vec2D', '__builtins__', '__doc__', '__loader__', '__name__', '__package__', '__spec__', 'addshape', 'back', 'backward', 'begin_fill', 'begin_poly', 'bgcolor', 'bgpic', 'bk', 'bye', 'circle', 'clear', 'clearscreen', 'clearstamp', 'clearstamps', 'clone', 'color', 'colormode', 'degrees', 'delay', 'distance', 'done', 'dot', 'down', 'end_fill', 'end_poly', 'exitonclick', 'fd', 'fillcolor', 'filling', 'forward', 'get_poly', 'get_shapepoly', 'getcanvas', 'getpen', 'getscreen', 'getshapes', 'getturtle', 'goto', 'heading', 'hideturtle', 'home', 'ht', 'isdown', 'isvisible', 'left', 'listen', 'lt', 'mainloop', 'mode', 'numinput', 'onclick', 'ondrag', 'onkey', 'onkeypress', 'onkeyrelease', 'onrelease', 'onscreenclick', 'ontimer', 'pd', 'pen', 'pencolor', 'pendown', 'pensize', 'penup', 'pos', 'position', 'pu', 'radians', 'register_shape', 'reset', 'resetscreen', 'resizemode', 'right', 'rt', 'screensize', 'seth', 'setheading', 'setpos', 'setposition', 'settiltangle', 'setundobuffer', 'setup', 'setworldcoordinates', 'setx', 'sety', 'shape', 'shapesize', 'shapetransform', 'shearfactor', 'showturtle', 'speed', 'st', 'stamp', 'textinput', 'tilt', 'tiltangle', 'title', 'towards', 'tracer', 'turtles', 'turtlesize', 'undo', 'undobufferentries', 'up', 'update', 'width', 'window_height', 'window_width', 'write', 'write_docstringdict', 'xcor', 'x座標', 'ycor', 'y座標', '下筆', '下筆嗎', '下筆狀態', '中英對照表', '主迴圈', '亂取樣', '亂整數', '亂數', '亂選', '二維向量類', '位置', '假', '做完了', '傾斜', '傾斜角度', '再見', '前往', '前進', '加形狀', '半徑數', '印', '原生筆類', '原生龜類', '去到', '取回復暫存區的長度', '取多邊形', '取幕', '取幕寬', '取幕高', '取形', '取形狀', '取形狀多邊形', '取時間', '取畫布', '取筆', '取龜', '取龜列表', '可捲畫布類', '可見狀態', '右轉', '向上鍵', '向下鍵', '向右鍵', '向左鍵', '向量2D類', '向量類', '回家', '回家鍵', '回復', '回復暫存區的個數', '回復暫存區的長度', '圓', '在幕點擊時', '在拖曳時', '在按下鍵時', '在按著鍵時', '在按鍵時', '在按鍵鬆開時', '在滑鼠拖曳龜時', '在滑鼠釋放龜時', '在滑鼠鍵點擊幕時', '在滑鼠鍵點擊時', '在滑鼠鬆開龜時', '在滑鼠點擊龜時', '在計時器若干毫秒之後', '在計時後', '在釋放時', '在鬆開時', '在點擊幕時', '在點擊時', '在點擊時離開', '在點擊龜時', '填色', '填色狀態', '大小', '寫', '寬', '左轉', '幕大小', '幕寬', '幕類', '幕高', '座標x', '座標y', '座標系統', '延遲', '弧度', '強度', '形', '形狀', '形狀大小', '形狀轉換', '形狀類', '後退', '戳印', '扭曲因子', '提筆', '方形', '是否下筆', '是否可見', '是否正在填色', '時間', '更新', '更新畫面', '朝向', '朝向xy', '標題', '模式', '橙', '橙色', '正在填色', '清除', '清除幕', '清除蓋章', '清除蓋章群', '清除鍵', '灰', '灰色', '烏龜形狀', '無', '生一隻龜', '生龜', '畫圓', '畫點', '登記形狀', '白', '白色', '看時間', '真', '睡', '空白鍵', '窗寬', '窗高', '筆', '筆大小', '筆寬', '筆屬性', '筆粗', '筆粗細', '筆色', '筆類', '等待閉幕', '等時間', '範圍', '紅', '紅色', '紫', '紫色', '結束填', '結束填色', '結束多邊形', '綠', '綠色', '聽', '聽鍵盤', '背景圖', '背景色', '脫離鍵', '色模式', '蓋印', '蓋章', '藍', '藍色', '藏', '藏龜', '複製', '角度', '角度從北開始順時針', '設x座標', '設y座標', '設位置', '設傾斜角度', '設傾角', '設取扭曲因子', '設回復暫存區', '設圓為2pi弧', '設圓為360度', '設座標x', '設座標y', '設座標系統', '設成可伸縮模式', '設標題', '設立', '設角為度', '設角為弧', '設角的單位為半徑數', '設角的單位為角度', '設頭向', '註冊形狀', '距離', '輸入數字', '輸入文字', '追蹤', '追蹤器', '追蹤更新畫面', '速度', '進入主迴圈', '重設', '重設大小模式', '重設幕', '重設幕大小', '重設幕寬高', '重設所有龜', '閉幕', '開始填', '開始填色', '開始多邊形', '開幕', '隨機取樣', '隨機整數', '隨機數', '隨機選', '隱藏', '離開在點擊時', '青', '青色', '頭向', '顏色', '顯', '顯示', '顯龜', '顯龜嗎', '黃', '黃色', '黑', '黑色', '點', '點擊x結束', '龜列表', '龜大小', '龜幕基類', '龜幕類', '龜形', '龜筆類', '龜群', '龜行類', '龜類']
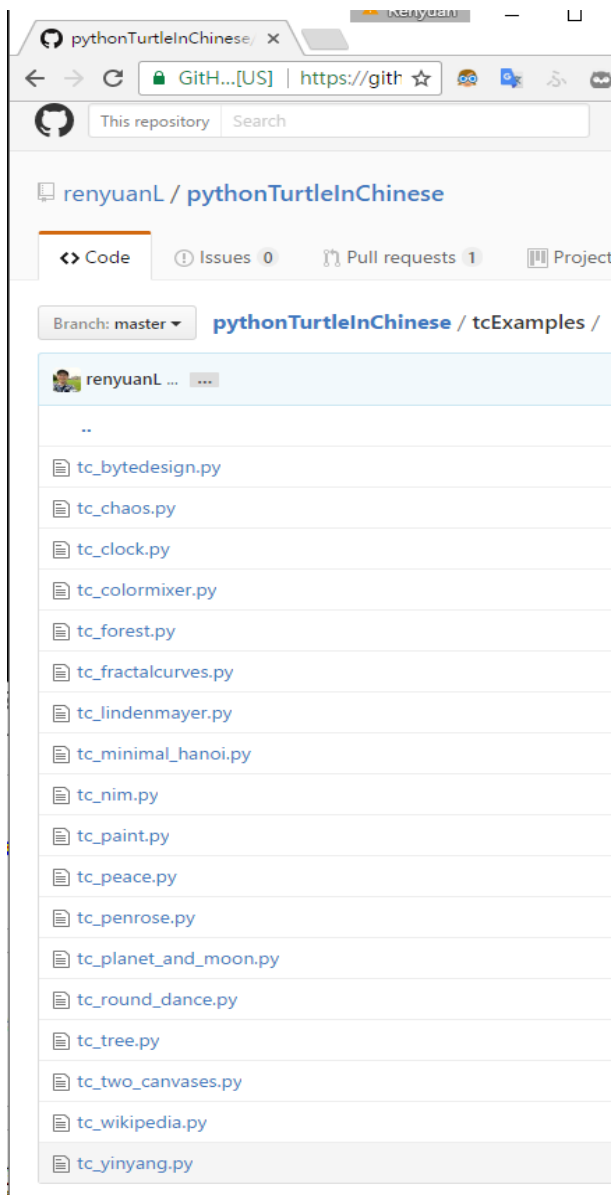
```
>>> len(dir())
368
```

368-128 == **240** , **turtle_tc** added

21

What would a Python program in traditional Chinese look like?


```python
#!/usr/bin/env python3
'''龜作圖範例集：

        tdemo_bytedesign.py

改編自範例集為例
對PythonCard的烏龜圖形。

它是基於在BYTE雜誌上的一篇文章
問題解決標誌：使用龜
圖形重繪設計
1982年11月，頁。 118 - 134


---------------------------------------------


由於該語句

t.delay (0)

在線152 ，這台動畫延遲
為0 ，這個動畫運行在"每行線"
模式盡可能快。=== 以上由 Google 翻譯，請協助改善 ===
'''


import math
from turtle_tc import *; from turtle_tc import 龜類, 主迴圈
from time import clock
```

```python
# 包裝任何額外的繪圖程序
# 這需要了解對方
class 設計師類(龜類):

    def 設計(我, 家的位置, 比例尺度):
        我.提筆()
        for i in 範圍(5):
            我.前進(64.65 * 比例尺度)
            我.下筆()
            我.輪子(我.位置(), 比例尺度)
            我.提筆()
            我.後退(64.65 * 比例尺度)
            我.右轉(72)
        我.提筆()
        我.前往(家的位置)
        我.右轉(36)
        我.前進(24.5 * 比例尺度)
        我.右轉(198)
        我.下筆()
        我.中央塊(46 * 比例尺度, 143.4, 比例尺度)
        我.取幕().追蹤(真)
```

What would a Python program in traditional Chinese look like?

```python
def 輪子(我, 開始位置, 比例尺度):
    我.右轉(54)
    for i in 範圍(4):
        我.五之塊(開始位置, 比例尺度)
    我.下筆()
    我.左轉(36)
    for i in 範圍(5):
        我.三之塊(開始位置, 比例尺度)
    我.左轉(36)
    for i in 範圍(5):
        我.下筆()
        我.右轉(72)
        我.前進(28 * 比例尺度)
        我.提筆()
        我.後退(28 * 比例尺度)
    我.左轉(54)
    我.取幕().更新()


def 三之塊(我, 開始位置, 比例尺度):
    舊頭向 = 我.頭向()
    我.下筆()
    我.後退(2.5 * 比例尺度)
    我.右三邊形(31.5 * 比例尺度, 比例尺度)
    我.提筆()
    我.前往(開始位置)
    我.設頭向(舊頭向)
    我.下筆()
    我.後退(2.5 * 比例尺度)
    我.左三邊形(31.5 * 比例尺度, 比例尺度)
    我.提筆()
    我.前往(開始位置)
    我.設頭向(舊頭向)
    我.左轉(72)
    我.取幕().更新()
```

```python
def 五之塊(我, 開始位置, 比例尺度):
    舊頭向 = 我.頭向()
    我.提筆()
    我.前進(29 * 比例尺度)
    我.下筆()
    for i in 範圍(5):
        我.前進(18 * 比例尺度)
        我.右轉(72)
    我.右五邊形(18 * 比例尺度, 75, 比例尺度)
    我.提筆()
    我.前往(開始位置)
    我.設頭向(舊頭向)
    我.前進(29 * 比例尺度)
    我.下筆()
    for i in 範圍(5):
        我.前進(18 * 比例尺度)
        我.右轉(72)
    我.左五邊形(18 * 比例尺度, 75, 比例尺度)
    我.提筆()
    我.前往(開始位置)
    我.設頭向(舊頭向)
    我.左轉(72)
    我.取幕().更新()


def 左五邊形(我, 邊, 角度, 比例尺度):
    if 邊 < (2 * 比例尺度): return
    我.前進(邊)
    我.左轉(角度)
    我.左五邊形(邊 - (.38 * 比例尺度), 角度, 比例尺度)


def 右五邊形(我, 邊, 角度, 比例尺度):
    if 邊 < (2 * 比例尺度): return
    我.前進(邊)
    我.右轉(角度)
    我.右五邊形(邊 - (.38 * 比例尺度), 角度, 比例尺度)
```

```python
def 右三邊形(我, 邊, 比例尺度):
    if 邊 < (4 * 比例尺度): return
    我.前進(邊)
    我.右轉(111)
    我.前進(邊 / 1.78)
    我.右轉(111)
    我.前進(邊 / 1.3)
    我.右轉(146)
    我.右三邊形(邊 * .75, 比例尺度)


def 左三邊形(我, 邊, 比例尺度):
    if 邊 < (4 * 比例尺度): return
    我.前進(邊)
    我.左轉(111)
    我.前進(邊 / 1.78)
    我.左轉(111)
    我.前進(邊 / 1.3)
    我.左轉(146)
    我.左三邊形(邊 * .75, 比例尺度)


def 中央塊(我, s, a, 比例尺度):
    我.前進(s); 我.左轉(a)
    if s < (7.5 * 比例尺度):
        return
    我.中央塊(s - (1.2 * 比例尺度), a, 比例尺度)


def 主函數():
    t = 設計師類()
    t.速度(0)
    t.藏龜()
    t.取幕().延遲(0)
    t.取幕().追蹤(0)
    at = clock()
    t.設計(t.位置(), 2)
    et = clock()
    return "執行時間: %.2f 秒。" % (et-at)


if __name__ == '__main__':
    訊息 = 主函數()
    印(訊息)
    主迴圈()
# Above: "tc_bytedesign.py", by Renyuan Lyu (呂仁園),
2015-03-06
# Original: "bytedesign.py", by Gregor Lingl.
```

Why not generate many such kinds of programs?

How?

Machine Translation!

GoogleTranslate ?

- No space between identifiers
  - e.g.
    from turtle import * → 從龜進口*

- One English name will be translated into multiple Chinese names
  - e.g.
    circle →圓
    circle →圈
    def →高清
    def →高清的

- Not knowing the preserved "Key words" in Python
  - e.g.
    from, import, def, return, if

- The function name in Chinese may not be defined in the provided modules.
  - e.g.
    reset →重啟
    width →寬度
    up →向上
    down →下
    ht →HT
    begin_fill → begin_fill

- The issues of symbols (半形 vs 全形)
  () → （）
  : → ：
  , → ，

cguTranslate

從英文到中文自動翻譯程式中的標識符之初步探究，以 Python 的 Turtle 模組為例

The Initial Study of Automatic Translation of Identifiers in a Program from English into Chinese ~ Using the Turtle Module in Python as an Example

研究生:郭詠欣， 指導教授:呂仁園老師

@Chang Gung University, Taoyuan, Taiwan

https://github.com/renyuanL/cguTranslate
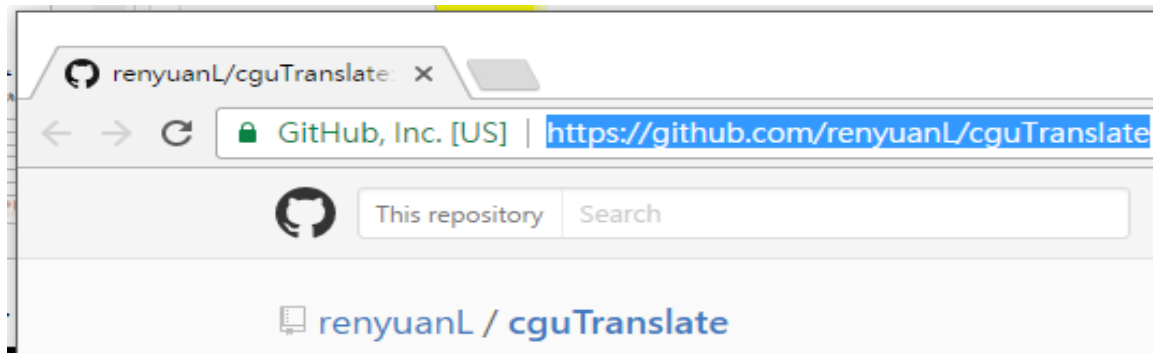
Task Definition

```python
from turtle import *

def yin(radius, color1, color2):
    width(3)
    color("black", color1)
    begin_fill()
    circle(radius/2., 180)
    circle(radius, 180)
    left(180)
    circle(-radius/2., 180)
    end_fill()
    left(90)
    up()
    forward(radius*0.35)
    right(90)
    down()
    color(color1, color2)
    begin_fill()
    circle(radius*0.15)
    end_fill()
    left(90)
    up()
    backward(radius*0.35)
    down()
    left(90)

def main():
    reset()
    yin(200, "black", "white")
    yin(200, "white", "black")
    ht()
    return "Done!"

if __name__ == '__main__':
    main()
    mainloop()
```

Machine Translation

```python
from turtle_tc import *

def 陰(半徑, 顏色1, 顏色2):
    筆寬(3)
    顏色(黑, 顏色1)
    開始填()
    畫圓(半徑/2., 180)
    畫圓(半徑, 180)
    左轉(180)
    畫圓(-半徑/2., 180)
    結束填()
    左轉(90)
    提筆()
    前進(半徑*0.35)
    右轉(90)
    下筆()
    顏色(顏色1, 顏色2)
    開始填()
    畫圓(半徑*0.15)
    結束填()
    左轉(90)
    提筆()
    後退(半徑*0.35)
    下筆()
    左轉(90)

def 主函數():
    重設()
    陰(200, 黑, 白)
    陰(200, 白, 黑)
    藏龜()
    return "Done!"

if __name__ == '__main__':
    主函數()
    主迴圈()
```

28

System architecture Overview
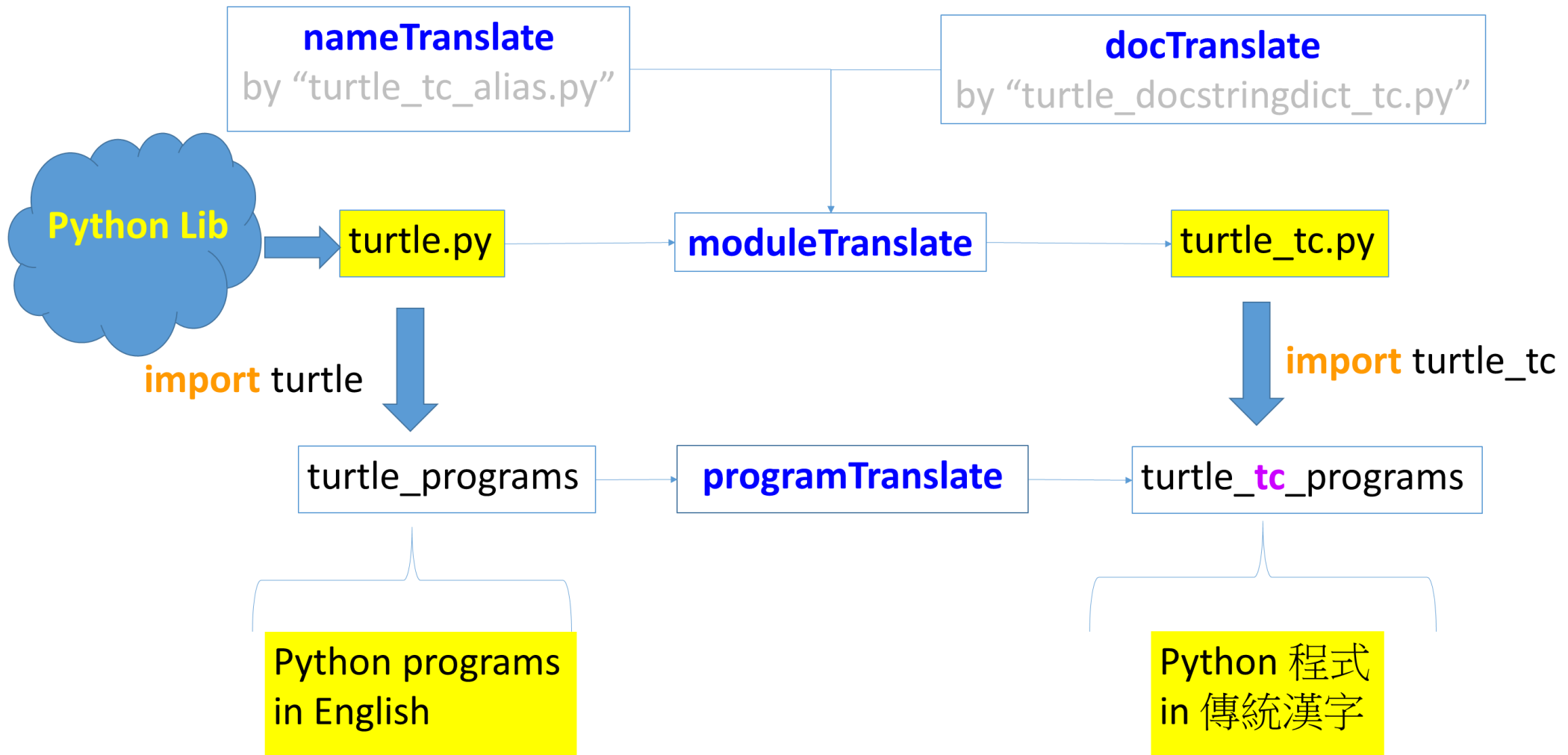
System architecture in more details

中文幫助文件
(turtle_docstri
ngdict_tc)

產生中文模組
(autoTurtle.py)

中文模組
turtle_tc.py

中英對照表
(turtle_tc_alias)

英文範例檔
(TurtleDemo
Files)

中英轉換字典
(ryDic.py)

英中程式自動翻譯
(TurtleTranslate.py)

中英轉換

英中轉換

自然語言翻譯
(ryOuterTrans
late.py)

圖形介面
(Turtle_GUI.py)
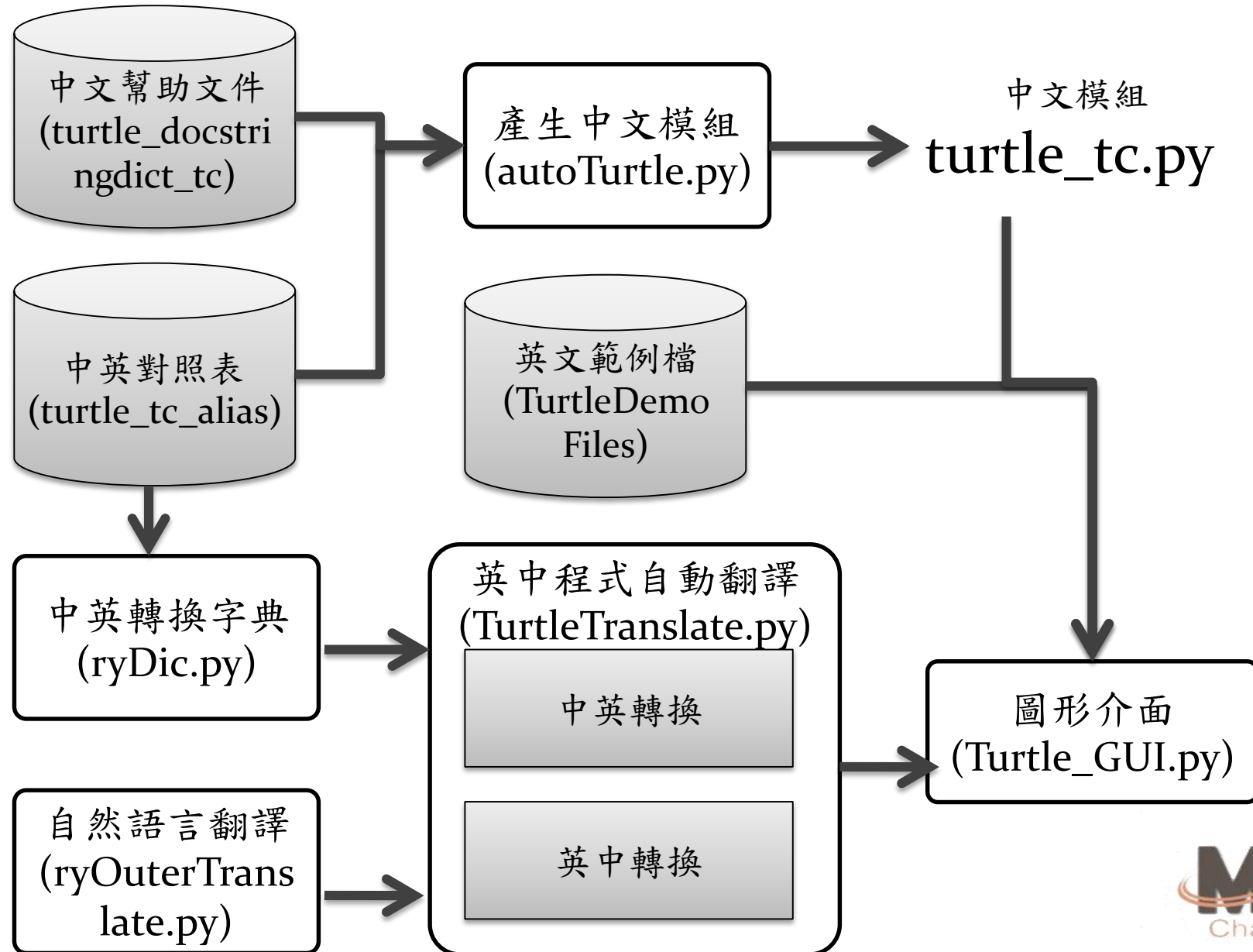
A set of 70+ translated python programs
Put in Github

Folders:
- byte_design
- clocks
- elementary
- fractals
- games
- geometry
- gravitation
- lindenmayer
- mill
- penrose_tiling
- python31_demo
- trees
- wikipedia_example

tc_Demo
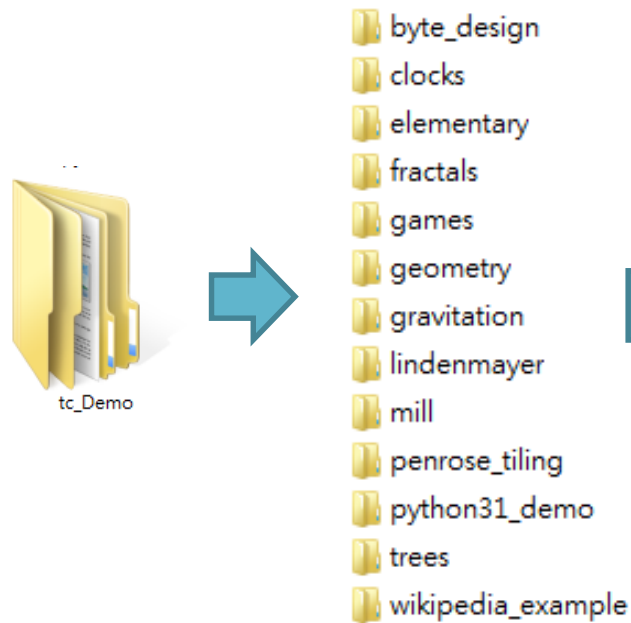
Files:
- tangramdata.py
- tc_2rtrees_generators.py
- tc_animation.py
- tc_another_forest.py
- tc_bytedesign.py
- tc_bytedesign_blue.py
- tc_bytedesign_fast.py
- tc_bytedesign_slow.py
- tc_chaos.py
- tc_clock.py
- tc_clock_2.py
- tc_clock_demo.py
- tc_colormixer.py
- tc_doublestar.py
- tc_forest.py
- tc_fractalcurves.py
- tc_game_of_life.py
- tc_game_of_life_coloured.py
- tc_graviturtle.py
- tc_hanoi_animation.py
- tc_I_dontlike_tiltdemo.py
- tc_illusion_1.py
- tc_illusion_2.py
- tc_illusion_3.py
- tc_illusion_4.py
- tc_illusion_5.py
- tc_indian_kolams.py
- tc_itree.py
- tc_lindenmayer_indian.py
- tc_mill1.py
- tc_mill2.py
- tc_minimal_hanoi.py
- tc_moorhuhn.py
- tc_nested_triangles.py
- tc_nim.py
- tc_paint.py
- tc_peace.py
- tc_peace_2.py
- tc_penrose.py
- tc_planet.py
- tc_planet_and_moon.py
- tc_planet_with_moon.py
- tc_planet_with_moon_2.py
- tc_planets.py
- tc_radioactive.py
- tc_recursive_squares.py
- tc_rhombi.py
- tc_round_dance.py
- tc_rtree1.py
- tc_rtree2_generator.py
- tc_scribble.py
- tc_sierpinski1.py
- tc_sierpinski2.py
- tc_spaceship.py
- tc_sun_and_inner_planets.py
- tc_sun_earth.py
- tc_sun_earth_moon.py
- tc_tangram.py
- tc_teddy.py
- tc_tree.py
- tc_trigeo.py
- tc_twoPlants.py
- tc_wikipedia.py
- tc_wikipedia1.py
- tc_wikipedia2.py
- tc_wikipedia3.py
- tc_wikipedia3_how2.py
- tc_wikipedia4.py
- tc_yinyang.py
- tc_yinyang_circle.py
- tc_yinyang_dot.py

System performance
evaluated over the 70+ programs

	Keyword	**Identifier**
Quantity	1937	12138
Percentage	14%	86%

	Translated	**Untranslated**
Quantity	7584	4554
Percentage	62%	38%

Demo
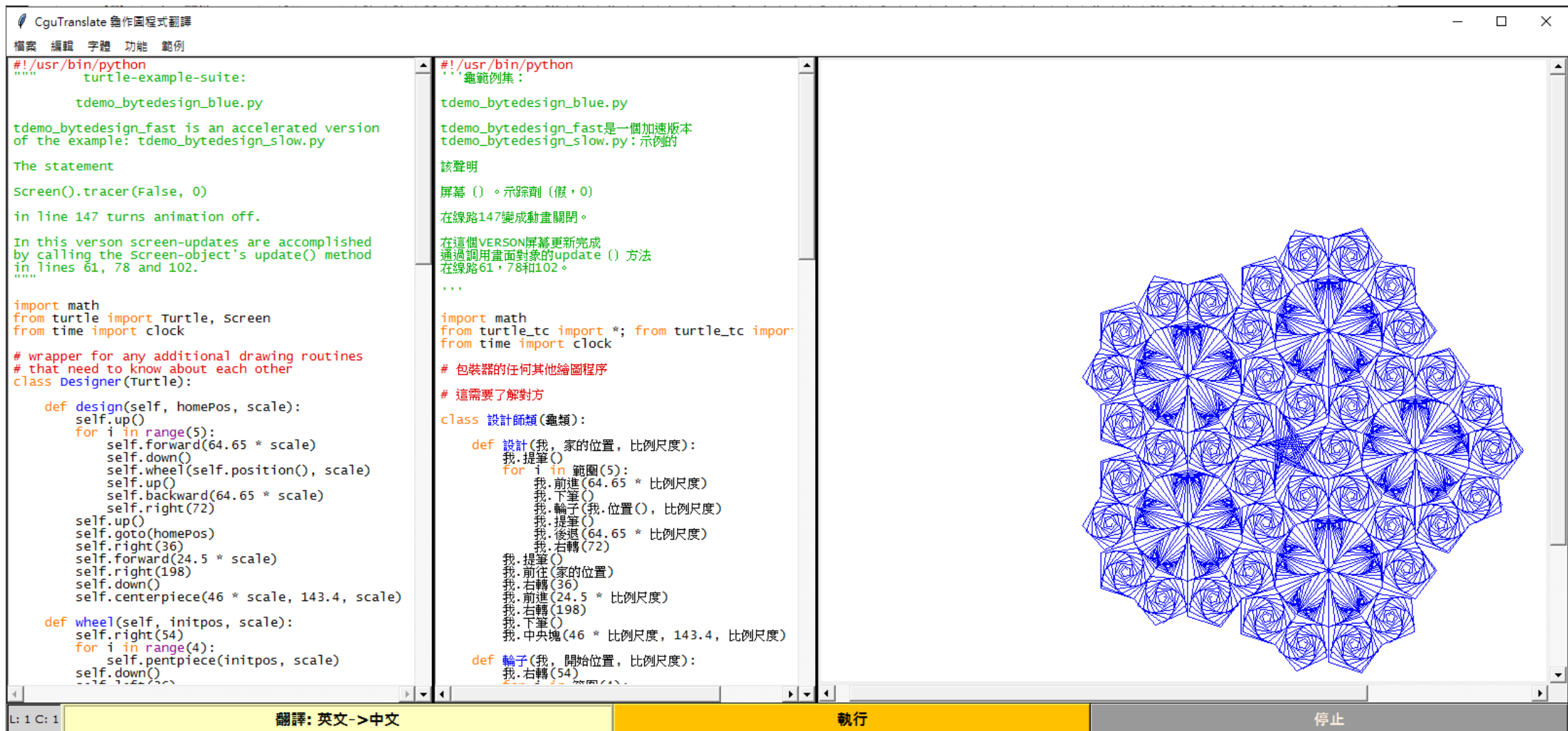
Conclusion

- We teach Reading, Writing, and Arithmetic to **kids** in our native or school educational languages. Why not try to **teach kids programming** in the same **language** with which they have been **natively** familiar.

- By machine translation, we could generate many example programs in native language as soon as possible.

- Although the current status of such a task is still far from perfect, it could provide a good start point to make more efforts in such direction, that helps the program education to the younger generation.

參考文獻

- 中蟒（chinese python）編程語言，http://www.chinesepython.org/
- 周蟒zhpy，https://code.google.com/archive/p/zhpy/
- A Python program written with 90% Traditional Chinese characters，22 May 2014
- Ren-Yuan Lyu,Translation of Python Programs into non-English Languages for Learners without English Proficiency，https://pycon.jp/2015/en/schedule/presentation/43/
- Python，http://www.python.org/
- Tokenizer，https://docs.python.org/3/library/tokenize.html
- Regular expression，https://docs.python.org/3/library/re.html
- Turtle graphics，https://docs.python.org/3/library/turtle.html?highlight=turtle#module-turtle
- Wikipedia，https://zh.wikipedia.org
- Gregor Lingl，Python turtle graphics demo suite for Python 3.1，https://code.google.com/archive/p/python-turtle-demo/downloads
- MyMemory.translated.net ，https://mymemory.translated.net

An International Discussion Forum about this Task

"A Python program written with 90% Traditional Chinese characters"

	Submitted by eah13
1	whoa I love the single-character self
3	我 (wo3) would probably be three keystrokes: two for "w" and "o", then enter/spacebar to confirm substitution into the Chinese character.
7	me too. Japanese (two character) is pretty cool too: 自己 . I'm thinking using characters would be a cool way to slim down a programming language, a bit like how greek and russian characters are used in math.
8	interestingly, that's self in chinese too. wo3 means I, while what you wrote literally means self.
9	I'm going to tattoo those Chinese characters on my arm.

And more ....

http://www.reddit.com/r/Python/comments/268fts/a_python_program_written_with_90_traditional/

謝謝耐心傾聽，歡迎指教。

Machine Translation
of English Identifiers
in **Python Programs**
into Traditional Chinese

Ren-yuan Lyu (呂仁園) , Yung-Hsin Kuo (郭詠欣), Che-Ning Liu (劉哲寧)

Computer Science and Information Engineering

Chang Gung University (長庚大學),

Taoyuan, Taiwan