https://dl.dropboxusercontent.com/u/33089565/_pyconjp2016/01.png

<fig.01> Task Definition

https://dl.dropboxusercontent.com/u/33089565/_pyconjp2016/03.png

https://dl.dropboxusercontent.com/u/33089565/_pyconjp2016/04.png

https://dl.dropboxusercontent.com/u/33089565/_pyconjp2016/06.png

Identifier	translated	untranslated
quantity	7584	4554
ratio	7584/12138	4554/12138
percentage	62%	38%

Automatic Translation of English Identifiers in Python Programs into Traditional Chinese (en)

Ren-yuan Lyu, Che-Ning Liu, ? in Education

DESCRIPTION:

This is a demonstration for a two-year research project sponsored by Taiwan government, which is aimed to translating Python programs into traditional Chinese in order to help those who feel interested in learning computer programming but are not very good at English. The idea could also be helpful for those countries where English is not an official language in elementary or secondary schools.

OBJECTIVES

Abstract

Introduction

Guido van Rossum mentioned in "Computer Programming for Everybody" that everyone should have a chance to write programs. However, in a non-English speaking countries, the lack of English proficiency may be the first obstacle to hinder people from coding. In order to lower down the threshold of English proficiency for potential programming learners, providing a non-English translation for tutorial programming examples was recognized as a significant task. The Scratch programming language   has been an example of success. One of the good points of Scratch is that learners can freely choose their most fluently native languages to learn it.

Thanks to Pep-3131 , starting in Python 3.0, the source coding of Python has been changed into UTF-8. This means that all the identifiers of the Python programs can be named in many international languages. Based on such an infrastructure, we choosed the Turtle module in Python as an example, made an alias list of all classes/functions in the module, and created an automatic translation program to translate a set of 70 more tutorial programs for use in teaching fundamental python programming to those who without English proficiency. All programs along with the translating/browsing program have been put in GitHub for reference.

In this proposal, we will provide the detail in the "**automatic**" translation of　Python programs into another natural languages, taking the Traditional Chinese as an example.　The authors hope to call for participation in translation into the other Asian Languages, such as Japanese, Korean and Simplified Chinese.

去年，我們在 PyCon JP 2015 大會上，發表了一研究報告，題為 "**Translation of Python Programs into non-English Languages for Learners without English Proficiency**" [](https://youtu.be/effsaVIfSws)，獲得很好的回響。在那份報告中，我們提出了用程式師的母語來撰寫程式的優點，並以 Python Turtle 模組為例，製造出傳統中文介面的翻譯模組 turtle_tc.py。使得用中文寫 Python 程式成為可能。我們並提出了 18 支 示範程式以及一個中文化瀏覽程式，以利 Python 程式教學者方便運用這個模組來製作更多教學示範程式。這對初級的 Python 程式教學是很有益處的創舉。

今年，延續這個主題，在台灣的政府(行政院科技部)的支持之下，我們執行了一個一年期的研究計畫，目標是把程式翻譯 (英翻中或中翻英) 的任務 藉由運用自然語言 之機器翻譯 之技術，把過程自動化，期能製造大量中文程式，使得在台灣的程式教學老師，有豐富的中文程式範例可以使用。

本報告即為這項研究的初步成果報告，我們除了完成了自動雙向翻譯 Python 程式 之 Python 程式之外，並成功的翻譯了一組 72 支程式的範例程式集合。對這個程式集合，我們還做了若干有趣的統計，顯示出我們的翻譯任務執行起來的效能，以作為未來改進翻譯引擎的根據。

(01)… Task Definition

- The task looks as that in <fig.01>, where the left column is the input, the middle column is the output, and the right column show their common running results.
- The input is the source code of an original Python program completely in English.
- The desired output is the automatically translated program in almost all Chinese identifiers, such that it looks like a Chinese Python program.
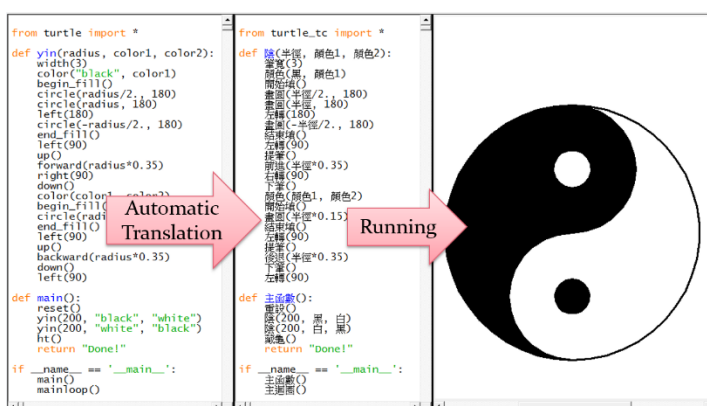
<fig.01> Task Definition

(02)… Block Diagram of the System

https://dl.dropboxusercontent.com/u/33089565/_pyconjp2016/02.png
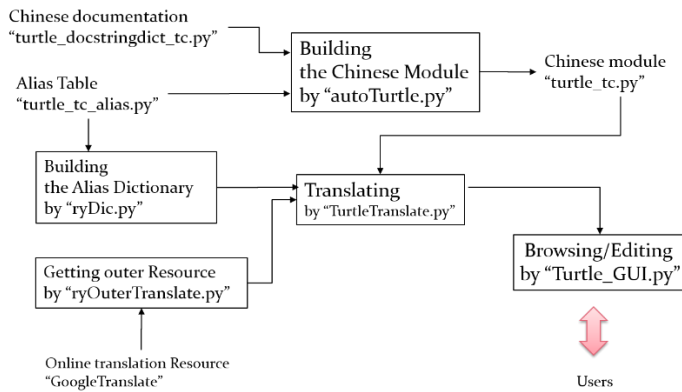
<fig.02> Block Diagram of the System

The whole system can be divided into the following "blocks"，and we will provide some description in more detail in the following

Chinese documentation: "turtle_docstringdict_tc.py"
Alias Table : "turtle_tc_alias.py"
Building the Chinese Module by "autoTurtle.py"
Chinese module: "turtle_tc.py"
Building the Alias Dictionary by "ryDic.py"
Online translation Resource "GoogleTranslate"
Getting outer Resource by "ryOuterTranslate.py"
Translating by "TurtleTranslate.py"
Browsing/Editing by "Turtle_GUI.py"
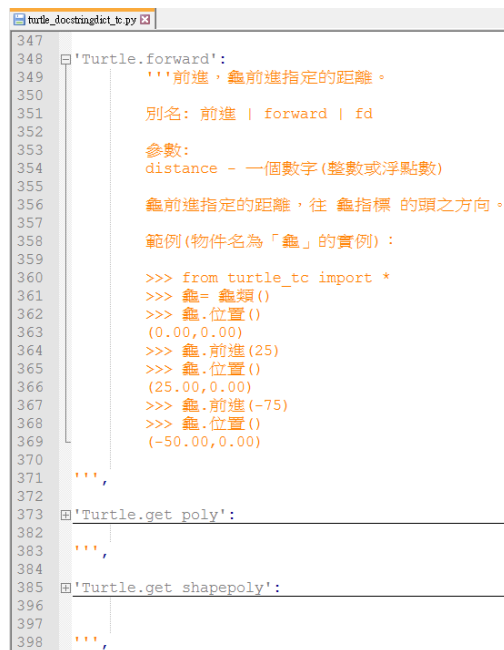
(03)
(04)…

(05)… Chinese documentation: "turtle_docstringdict_tc.py"

For programmers to get the online help, the module should contain the "docstrings" in itself. The original turtle module provided the "turtle_docstringdict.py" as an example to code the docstrings into the program. We use GoogleTranslate to translate it into a Chinese version roughly and then manually correct them into more readable text. It was then named as "turtle_docstringdict_tc.py".　Using "_tc" in the filename as a language code to represent "traditional Chinese".　It turns out to be a near 2000-line-long program. Although this is not much technically attractive, it is important for the whole project to succeed.

Partially of it look as the following

```
347
348  'Turtle.forward':
349        '''前進，龜前進指定的距離。
350
351        別名: 前進 | forward | fd
352
353        參數:
354        distance – 一個數字(整數或浮點數)
355
356        龜前進指定的距離，往 龜指標 的頭之方向。
357
358        範例(物件名為「龜」的實例):
359
360        >>> from turtle_tc import *
361        >>> 龜= 龜類()
362        >>> 龜.位置()
363        (0.00,0.00)
364        >>> 龜.前進(25)
365        >>> 龜.位置()
366        (25.00,0.00)
367        >>> 龜.前進(-75)
368        >>> 龜.位置()
369        (-50.00,0.00)
370
371  ''',
372
373  'Turtle.get_poly':
382
383  ''',
384
385  'Turtle.get_shapepoly':
396
397
398  ''',
```

[fig.101]

Thanks to this effort, the programmers can use online help just by typing in ">>> help(前進)" to get the important information for the function like the following:

```
>>> from turtle_tc import *
>>> help(前進)
Help on function 前進 in module turtle_tc:

前進(distance)
    『0033  中文說明』
    前進，龜前進指定的距離。

    別名: 前進 | forward | fd

    參數:
    distance - 一個數字(整數或浮點數)

    龜前進指定的距離，往 龜指標 的頭之方向。

    範例(物件名為「龜」的實例):

    >>> from turtle_tc import *
    >>> 龜= 龜類()
    >>> 龜.位置()
    (0.00,0.00)
    >>> 龜.前進(25)
    >>> 龜.位置()
    (25.00,0.00)
    >>> 龜.前進(-75)
    >>> 龜.位置()
    (-50.00,0.00)
```

[fig.102]

(06)… Alias Table : "turtle_tc_alias.py"

An alias is a second name for an identifier. When this alias is called, it equals to call the original identifier. Using alias is the major strategy we adopted in this project. It simplifies the whole process for translating English identifiers into Chinese identifiers to a significant degree.

To get the most convenience of such an "alias strategy", we hard-coded the whole table of English-to-Chinese correspondence for all the major identifiers used in the Turtle module, including those of classes, methods, globally accessible variables and functions. This table was put in the module "turtle_tc_alias.py", where the simple Python List structures were used to store the correspondence between 2 languages. A partial list of this module looks like the following figure.

```
turtle_tc_alias.py ☒
12
13  ⊟cListTurtleScreenBase=[
14       ('TurtleScreenBase', '龜幕基類', '烏龜螢幕地
15       ('mainloop'    ,    '主迴圈', '進入主迴圈', '做
16       ('numinput'    ,    '輸入數字'),
17       ('textinput'   ,    '輸入文字'),
18       ]
19
20
21
22  ⊟cListTurtleScreen=[
23       ('TurtleScreen',              '龜幕類', '丿
24
25       ('addshape',                 '加形狀'),
26       ('bgcolor',                  '背景色'),
27       ('bgpic',                    '背景圖'),
28       #('clear',                   '清除'),  #
29       ('clearscreen',              '清除幕'),
30       ('colormode',                '色模式'),
31       ('delay',                    '延遲'),
32       ('getcanvas',                '取畫布'),
33       ('getshapes',                '取形', '取开
34       ('listen',                   '聽', '聽鍵虎
35       ('mode',                     '模式'),
36
37       ('onclick',                  '在點擊時',
38       ('onclick',                  '在滑鼠鍵點
39       ('onkey',                    '在按鍵時',
40
41       ('onkeypress',               '在按著鍵時
42       ('onkeyrelease',             '在按鍵鬆開
43       ('onscreenclick',            '在點擊幕時
44       ('ontimer',                  '在計時後',
45
```

[fig.103]

(07)… Chinese module: "turtle_tc.py"

This is the major python module we built up based on the original turtle module "turtle.py" and the other 2 associated modules discussed in the above, i.e., the Alias Table and the Chinese documentation file. This major module was named as "**turtle_tc.py",** providing the alias in **traditional Chinese** (thus the subscript "**_tc**") for almost all identifiers (names) in "turtle.py". The previous version was described in detail in the Pycon JP 2015, and could be downloaded from Gihub, at http://github.com/renyuanL/pythonTurtleInChinese.

A partial list of the alias identifiers in **turtle_tc.py** looks like this:

```
turtle_tc.py ☒
  17
  18  ⊞def  取筆():
  25    #設置 _getscreen的中文別名
  26  ⊞def  取幕():
  32  ⊞class Vec2D(tuple):
  75    向量類= Vec2D
  76    二維向量類= Vec2D
  77  ⊞class Shape(object):
 123    形狀類= Shape
 124  ⊞class TurtleScreenBase(object):
 573    龜幕基類= TurtleScreenBase
 574    烏龜螢幕地基類= TurtleScreenBase
 575  ⊞class TurtleScreen(TurtleScreenBase):
1562    龜幕類= TurtleScreen
1563    烏龜螢幕類= TurtleScreen
1564  ⊞class TNavigator(object):
2478    龜行類= TNavigator
2479    烏龜航行類= TNavigator
2480  ⊞class TPen(object):
3275    龜筆類= TPen
3276    烏龜畫筆類= TPen
3277  ⊞class RawTurtle(TPen, TNavigator):
4997    原龜類= RawTurtle
4998    粗龜類= RawTurtle
4999    原生龜類= RawTurtle
5000  ⊞class _Screen(TurtleScreen):
5206    _幕類= _Screen
5207    _螢幕類= _Screen
5208  ⊞def Screen():
5215    幕類= Screen
5216    螢幕類= Screen
5217    開幕= Screen
5218  ⊞class Turtle(RawTurtle):
5237    龜類= Turtle
5238    烏龜類= Turtle
5239    生一隻龜= Turtle
5240    清除鍵="Delete"
5241    向下鍵="Down"
```

[fig.104]

You will find that the bi-lingual translation pairs provided in the Alias Table was now carefully aligned in this python module in a perfect Python syntax.

The major improvement in the current version is that this module could be **generated automatically** using another building program "autoTurtle.py", which read in the Alias Table and the Chinese documentation file, and did some routine procedures automatically. Such a building program will be described in the next paragraph.

(08)… Building the Chinese Module by "autoTurtle.py"

The first version of "turtle_tc.py" was hard coded manually. This made it difficult to transfer the code into another language. Therefore, we created a building program named "autoTurtle.py" for building the major Chinese module by reading in the Alias Table of the bi-lingual pairs. The advantage to do this is to make it easier to change into the other languages.

The main steps for "autoTurtle.py" could be summarized in the following steps:
- 1. Reading in the Alias Table and generate the codes for aliasing all classes with methods in the Turtle module. A partial list of the automatically generated code in "turtle_tc.py" looks like the following figure.

[fig.105]

- 2. Reading in the docstrings and insert them into the appropriate places in the code.

  https://dl.dropboxusercontent.com/u/33089565/_pyconjp2016/10.png

[fig.106]

- 3. Generating the other alias, which were not incorporated in the Alias Table.

[fig.107]

- 4. Generating a subset of globally accessible functions, which would be used in procedure-oriented programming paradigm.

[fig.108]

- 5. Writing out the final result as "turtle_tc.py" to be used as the major Chinese module.

  Currently, this is a near **6000-line-long** python module file used as the major import for Chinese Python Turtle Graphics.

(09)…Building an Alias Dictionary for **out-of-Turtle** identifiers by "ryDic.py"

Only using the alias in the turtle module cannot cover all identifiers to a satisfactory percentage. To increase the coverage of the code translation. We needed to add many **out-of-Turtle** identifiers, which were not included in the Turtle module. Those identifiers were collected by looking up all turtle programs we had collected in the web, totally about 70 programs. By doing some statistics, we chosen those identifiers with frequency more than a threshold, translated them using GoogleTranslate with human correction. All of these were put in another Alias Dictionary, named "ryDic.py"
The partial list looks like the following figure.

```
ryDic.py ☒  turtle_tc.py ☒  autoTurtle.py ☒
  6     from turtle_tc_alias import *
  7  增加的中英對照表=[
  8
  9         ('self', '我'),
 10         ('main', '主函數'),
 11
 12         ('scale', '比例尺度'),
 13         ('game', '遊戲'),
 14         ('row', '列'),
 15         ('writer', '寫手'),
 16         ('col', '行'),
 17         ('size', '尺寸'),
 18         ('side', '邊'),
 19         ('msg', '訊息'),
 20         ('level', '等級'),
 21         ('sticks', '棒子們'),
 22         ('state', '狀態'),
 23         ('radius', '半徑'),
 24         ('tiledict', '瓦片字典'),
 25         ('start', '開始'),
 26
 27         ('parity','對等性'), # 一�

 28
 29         ('initpos','開始位置'),
 30         ('gravSys','重力系統'),
 31         ('dt','dt'),
 32         ('Nim','捻遊戲類'), # 有大

 33
 34         ('tree','樹'),
 35         ('ta','ta'),
 36         ('sun','日'),
 37         ('player','玩家'),
 38         ('model','模型'),
```

[fig.109]

(10)…Online translation Resource "GoogleTranslate"

In order to further improve the translation coverage in arbitrary Turtle program, the GoogleTranslate was considered in translating the part of code in natural English, including the text in string variables or in the comment lines. Although it is not perfect at present for arbitrary English sentences, there are

indeed valuable for reference for experienced users, who can help polish the machine translation for younger learners. And in this project, we will use a module to deal with communication to the GoogleTranslate. It will be described in the next paragraph.

[fig.110]

(11)… Getting outer Resource by "ryOuterTranslate.py"

- This is a small module adapted from "terryyin/google-translate-python" in GitHub.
- It was originally based on GoogleTranslate, but currently on MyMemory-Translation (http://mymemory.translated.net/), because of GoogleTranslate no longer providing free API.
- It uses Python's Standard Library–"urllib.request" and "urllib.parse".
- It sends HTTP GET Request to MyMemory with parameters like that in the following: "http://mymemory.translated.net/api/get?q=An%20example& langpair=en|zh-TW"
- It returns string of JSON format, which looks as the following.

  {"responseData":{"translatedText":"舉個例子
  ","match":0.85},"responseDetails":"","responseStatus":200,"responderId":"235","matches":[{"id":
 0,"segment":"An example adapted from the example-suite",...............}

- All we need to do is to parse the string and get what we need as the translated text.

(12)… Translating by "TurtleTranslate.py"

By integrating the submodules described in the above paragraphs, the main process of translating could be set up. The block diagram in [fig.201] shows the process in detail. The whole process could be divided into 5 steps:

    (1) Parsing the input program

    (2) Translating the identifiers

    (3) Translating the comments

    (4) Post processing

    (5) Outputing

https://dl.dropboxusercontent.com/u/33089565/_pyconjp2016/201.png
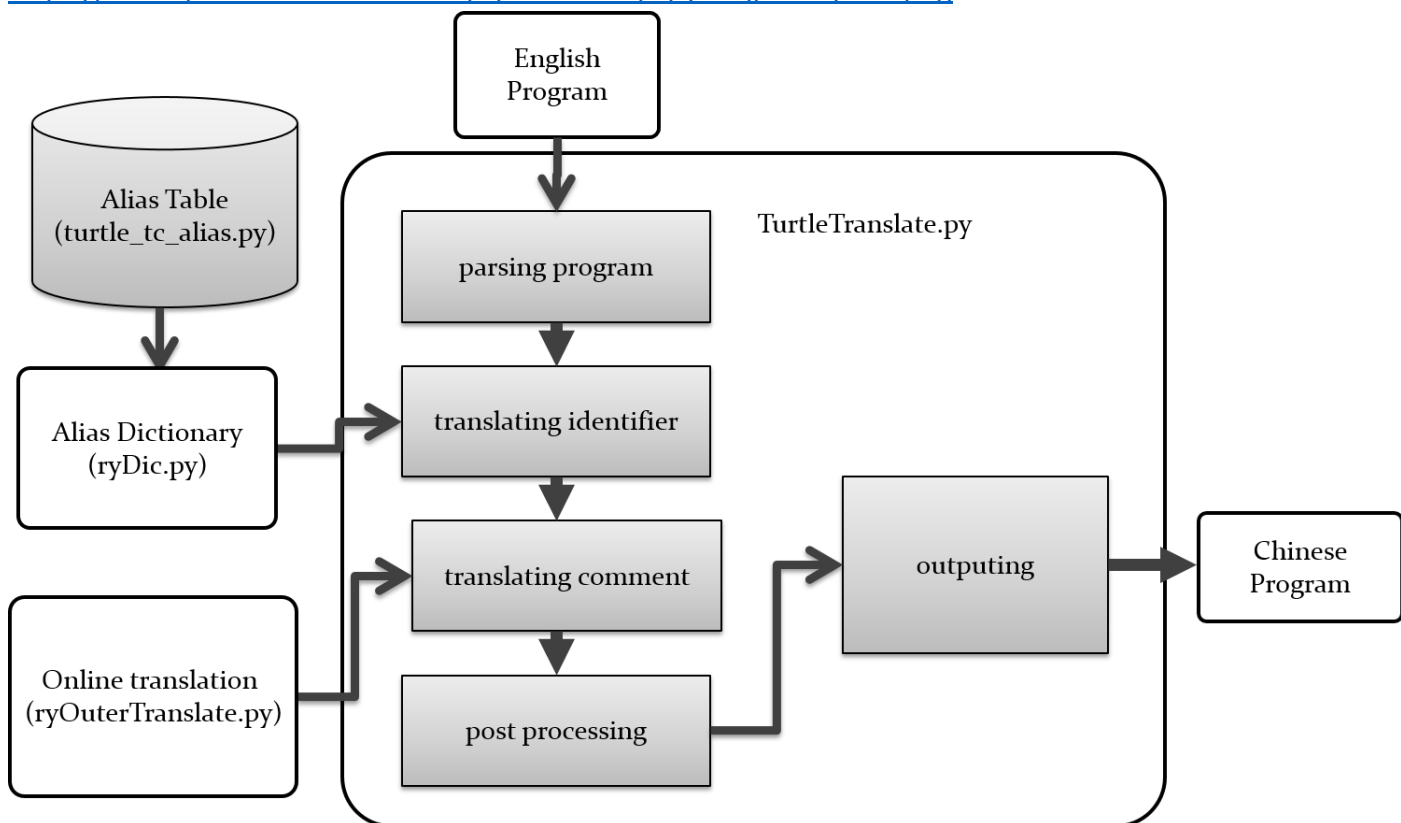
[fig.201]

Let's describe more in detail about each sub-processing block.

    (1) Parsing the input program

This process uses the "**generate_tokens()**", a method from the **"tokenize"** module, one of the critically important modules used in this project. This process will generate type, string and position as tokens. Then they will be stored in a Python List. Taking a short segment of python code as an example to be parsed. After tokenization, we could get the results as shown in the following. Among all types of tokens, 3 of them were chosen to be further processed, i.e,

NAME, STRING, and COMMENT.

https://dl.dropboxusercontent.com/u/33089565/_pyconjp2016/202.png

Python Code

```
from turtle import *

def yin(radius, color1, color2):
    width(3)
    color("black", color1)
    begin_fill()
    circle(radius/2., 180)
    circle(radius, 180)
    left(180)
```

Tokenization →

Result of tokenization

index	Position	Type	String
1	0,0-0,0	ENCODING	'utf-8'
2	1, 1-1, 5	NAME	'from'
3	1, 6-1, 12	NAME	'turtle'
4	1, 13-1, 19	NAME	'import'
5	1, 20-1, 21	OP	'*'
6	1, 21-1, 22	NEWLINE	'\n'
7	2, 0-2, 1	NL	'\n'
8	3,0-3,3	NAME	'def'
9	3,4-3,7	NAME	'yin'
10	3,7-3,8	OP	'('
11	3,8-3,14	NAME	'radius'
12	3,14-3,15	OP	','
13	3,16-3,22	NAME	'color1'
14	3,22-3,23	OP	','
15	3,24-3,30	NAME	'color2'
16	3,30-3,31	OP	','

[fig.202]

(2) Translating the names of identifiers

As long as the names of identifiers were spot, we did a table lookup for translating English words into Chinese characters. However, there was one exception. We do not translate those names used in the ***named arguments*** inside a function which was imported from those modules not created by ourselves. Because those ones were usually in English inside those modules, where we decided not to alter anything in the source code. The figures listed below help clarify this strategy.

[fig.203] The usual cases we did the translations for names of identifiers

Before Translation	After Translation

def **exercise** (run=0, jump=0):
  pass
 	
exercise (run=1)
	def 運動 (跑=0, 跳=0):
	  pass
	
	運動 (跑=1)

[fig.204] An exceptional name ("jump"), which was not translated.

Before Translation	After Translation
from outside *import* walk	*from* outside *import* walk as 步行
walk(jump=0)	步行(jump=0)

The detail flow chart for doing the translating task could be then summarized as follows:
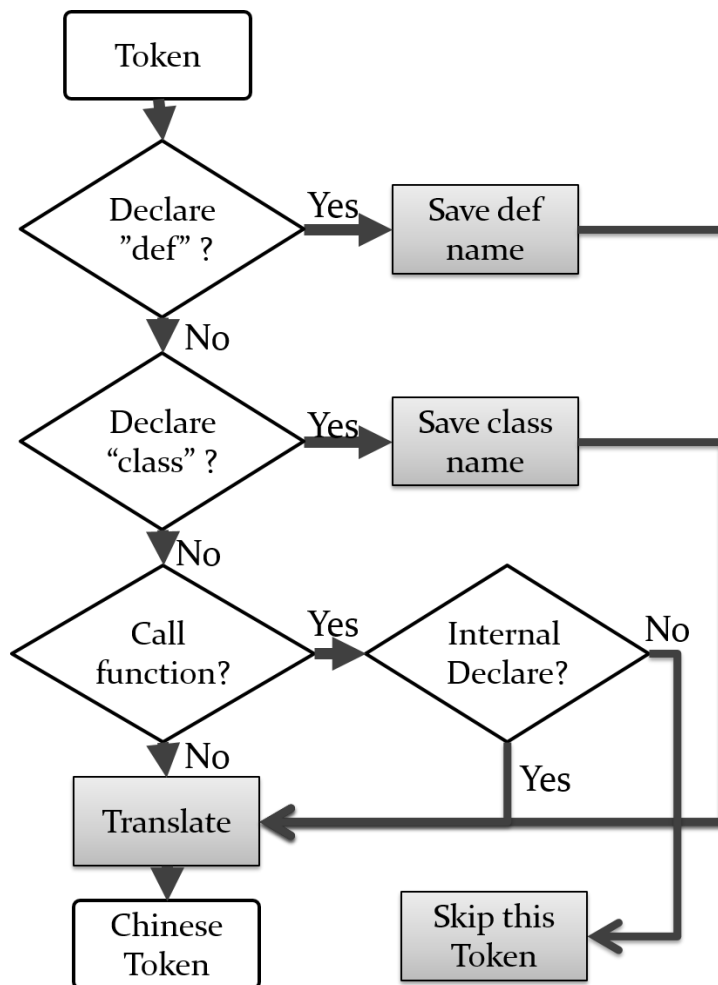
[fig.205] the flow chart for doing the translating task for names

https://dl.dropboxusercontent.com/u/33089565/_pyconjp2016/205.png

(3) Translating the comments

This part of sub-processing deals with the comments or text strings in the code. They are usually in natural language (English) syntax, not in the Python syntax. We call the outside machine translation engine to perform such a task.

The flowchart in the following figure show the detail to do it.
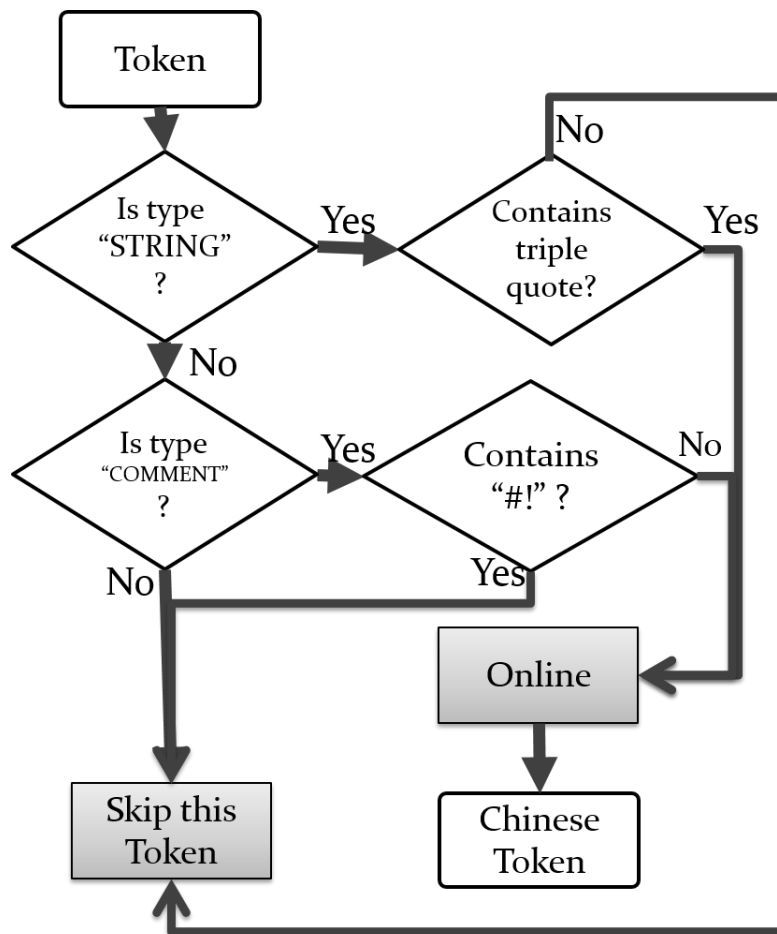
[fig.206] the flow chart for doing the translating task for comments and text strings
https://dl.dropboxusercontent.com/u/33089565/_pyconjp2016/206.png

(4) Post processing

There exists an untokenize() function in the tokenize module. It was used to combine all translated tokens into a syntactically correct Python codes. Furthermore, several post processing steps should be done, including modifying the head of the program, changing the import module from "turtle" to "turtle_tc", … etc.

(5) Outputing

Finally, the task of translation from a English program into an Chinese program was done！ Here is a good example.

```
from turtle import *                          from turtle_tc import *
def switchupdown(x=0, y=0):                    def 切換提筆下筆(x=0, y=0):
    if pen()["pendown"]:                           if 筆()["pendown"]:
        end_fill()                                     結束填()
        up()                                           提筆()
    else:                                          else:
        down()                                         下筆()
        begin_fill()                                   開始填()

def changecolor(x=0, y=0):                     def 改變顏色(x=0, y=0):
    global colors                                  global 顏色們
    colors = colors[1:]+colors[:1]                 顏色們 = 顏色們[1:]+顏色們[:1]
    color(colors[0])                               顏色(顏色們[0])

def main():                                    def 主函數():
    global colors                                  global 顏色們
    shape("circle")                                形狀("circle")
    resizemode("user")                             重設大小模式("user")
    shapesize(.5)                                  形狀大小(.5)
    width(3)                                       筆寬(3)
    colors=["red", "green", "blue", "yellow"]      顏色們=[紅, 綠, 藍, 黃]
    color(colors[0])                               顏色(顏色們[0])
    switchupdown()                                 切換提筆下筆()
    onscreenclick(goto,1)                          在點擊幕時(前往,1)
    onscreenclick(changecolor,2)                   在點擊幕時(改變顏色,2)
    onscreenclick(switchupdown,3)                  在點擊幕時(切換提筆下筆,3)
    return "EVENTLOOP"                             return "EVENTLOOP"

if __name__ == "__main__":                     if __name__ == "__main__":
    msg = main()                                   訊息 = 主函數()
    print(msg)                                     印(訊息)
    mainloop()                                     主迴圈()
```

(13)… Browsing/Editing by "Turtle_GUI.py"

(14)

(15)…

(16)…

(17)…

(18)…

(19)…

(20)…

(21)…

(22)…

(23)…

(24)