**Modules**

April 29, 2017 · 1,294 views

ในการเขียนโปรแกรมในภาษา Python โมดูล (Module) คือไฟล์ของโปรแกรมที่กำหนดตัวแปร ฟังก์ชัน หรือคลาสโดยแบ่งย่อยออกไปจากโปรแกรมหลัก และสามารถนำมาใช้งานได้โดยการนำเข้ามาในโปรแกรม (Import) กล่าวอีกนัยหนึ่ง โมดูลก็คือไลบรารี่ที่สร้างไว้และนำมาใช้งานในโปรแกรม ในบทนี้ เราจะพูดถึงความหมายของโมดูล การสร้าง และการใช้งานโมดูลในการเขียนโปรแกรม นอกจากนี้ เราจะแนะนำให้คุณรู้จักกับ Package ซึ่งเป็นแนวคิดในการจัดการกับโมดูลในรูปแบบของ Namespace

**โมดูล คืออะไร**

โมดูล (Module) คือไฟล์หรือส่วนของโปรแกรมที่ใช้สำหรับกำหนดตัวแปร ฟังก์ชัน หรือคลาสโดยแบ่งย่อยอีกหน่วยหนึ่งจากโปรแกรมหลัก และในโมดูลยังสามารถประกอบไปด้วยคำสั่งประมวลผลการทำงานได้ ยกตัวอย่างเช่น เมื่อคุณเขียนโปรแกรมในภาษา Python คุณอาจจะมีฟังก์ชันสำหรับทำงานและจัดการกับตัวเลขเป็นจำนวนมาก และในขณะเดียวกัน คุณไม่ต้องการให้โปรแกรมหลักนั้นมีขนาดใหญ่เกินไป นั่นหมายความว่าคุณสามารถนำฟังก์ชันเหล่านี้มาสร้างเป็นโมดูล และในการใช้งานนั้นจะต้องนำเข้ามาในโปรแกรมโดยวิธีที่เรียกว่า Import



คุณจะเห็นว่าโมดูลก็คือการแยกส่วนของโปรแกรมออกไปเป็นอีกส่วนและสามารถเรียกใช้ได้เมื่อต้องการ หรือกล่าวอีกนัยหนึ่ง โมดูลก็เหมือนไลบรารีของฟังก์ชันและคลาสต่างๆ นั่นเป็นเพราะว่าเมื่อโปรแกรมของคุณมีขนาดใหญ่ คุณสามารถแบ่งส่วนต่างๆ ของโปรแกรมออกเป็นโมดูลย่อยๆ เพื่อให้ง่ายต่อการจัดการและการใช้งาน ในภาษา Python โมดูลที่ถูกสร้างขึ้นมานั้นจะเป็นไฟล์ในรูปแบบ module\_name.py และนอกจากนี้ Python ยังมี Built-in module เป็นจำนวนมาก เช่น math เป็นโมดูลเกี่ยวกับฟังก์ชันทางคณิตศาสตร์ หรือ random เป็นโมดูลเพื่อจัดการและสุ่มตัวเลข เป็นต้น

**การสร้างโมดูลในภาษา Python**

ในการสร้างโมดูลในภาษา Python คุณต้องนำโค้ดของโปรแกรม โดยทั่วไปแล้วจะประกอบไปด้วย ตัวแปร ฟังก์ชัน หรือคลาส ไปรวมไว้ในไฟล์ใหม่ที่ไม่ใช้ไฟล์หลักของโปรแกรม ในรูปแบบ module\_name.py โดยที่module\_name นั้นเป็นชื่อของโมดูลเพื่อนำไปใช้งานในการเขียนโปรแกรม โดยการเรียกใช้ด้วยคำสั่ง import ต่อไปมาดูตัวอย่างการสร้างโมดูลในภาษา Python

# number.py
def factorial(n): # return factorial value of n
 if n == 0 or n == 1:
 return 1
 else:
 return n \* factorial(n - 1)

def fibonacci(n): # return Fibonacci series up to n
 result = []
 a, b = 0, 1
 while b < n:
 result.append(b)
 a, b = b, a + b
 return result

ในตัวอย่าง เป็นการสร้างโมดูลโดยไฟล์ของโมดูลนั้นมีชื่อว่า number.py นั่นหมายความว่าโมดูลนี้มีชื่อว่า number ซึ่งนี่เป็นสิ่งที่เราจะใช้สำหรับเรียกใช้งานโมดูลในการเขียนโปรแกรม ภายในโมดูลประกอบไปด้วย 2 ฟังก์ชันที่ทำงานเกี่ยวกับตัวเลข ฟังก์ชัน factorial() เป็นฟังก์ชันสำหรับหาค่า Factorial ของตัวเลขจำนวนเต็ม n ซึ่งเป็น *Recursive function* และฟังก์ชัน fibonacci() ใช้หาลำดับของ Fibonacci จนถึงจำนวนเต็ม n

**การนำเข้าโมดูลด้วยคำสั่ง import**

หลังจากที่เราได้สร้างโมดูลไปแล้ว ต่อไปจะเป็นการนำโมดูลดังกล่าวมาใช้งาน ในภาษา Python นั้นจะใช้คำสั่ง import เพื่อนำเข้าโมดูลเพื่อนำมาใช้งานในโปรแกรม มาดูตัวอย่างการใช้งานโมดูล number ในตัวอย่างก่อนหน้า นี่เป็นโค้ดของโปรแกรม

import number

print('5! = ', number.factorial(5))
print(number.fibonacci(100))

ในตัวอย่าง เป็นการนำโมดูล number เข้ามาใช้งานที่โปรแกรมหลัก (Main program) โดยการใช้คำสั่ง import และตามด้วยชื่อของโมดูล นี่เป็นการนำเข้าสมาชิกทั้งหมดเข้ามาในโปรแกรม ในการใช้งานฟังก์ชันในโมดูลนั้นสามารถทำได้โดยใช้ชื่อของโมดูล ตามด้วยเครื่องหมายจุด และหลังจากนั้นเป็นฟังก์ชันที่ต้องการ ในรูปแบบ module\_name.object\_name เราได้เข้าถึงฟังก์ชันทั้งสองในโมดูล

5! = 120
[1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]

นี่เป็นผลลัพธ์การทำงานของโปรแกรม ซึ่งนี่แสดงให้เห็นอย่างชัดเจนว่า เราสามารถใช้โมดูลในการแยกโค้ดของโปรแกรมออกไปต่างหากได้ ซึ่งจะทำให้โปรแกรมสั้นลงหรือทำเป็นไลบรารี่เพื่อนำไว้ใช้เมื่อต้องการ

**การนำเข้าโมดูลด้วยคำสั่ง from ... import**

ในการใช้งานคำสั่ง import นั้นจะเป็นการนำเข้าออบเจ็คทั้งหมดในโมดูลเข้ามายังโปรแกรม และการใช้งานฟังก์ชันหรือออบเจ็คภายในโมดูลจะต้องนำหน้าด้วยชื่อโมดูลเสมอ ในภาษา Python นั้นมีคำสั่ง from import สำหรับนำเข้าข้อมูลบางส่วนภายในโมดูล และสามารถใช้งานออบเจ็คได้โดยตรงโดยไม่ต้องมี Prefix ชื่อของโมดูล มาดูตัวอย่างการใช้งาน

from number import factorial

print('5! = ', factorial(5))
print('3! = ', factorial(3))

ในตัวอย่าง เป็นการใช้งานคำสั่ง from ... import เพื่อนำเข้าฟังก์ชันภายในโมดูล numberจะเห็นได้ว่าเราได้นำเข้าเพียงฟังก์ชัน factorial() เข้ามาในโปรแกรมและในตอนใช้งานนั้นสามารถใช้ได้โดยที่ไม่ต้องใช้ Prefix ในการเรียกใช้

from number import factorial, fibonacci
# or
from number import \*

ในอีกทางหนึ่ง คุณสามารถนำเข้าหลายออบเจ็ค (ฟังก์ชัน) ในโมดูลโดยการใช้เครื่องหมายคอมมา (,) เป็นตัวคั่น หรือนำเข้าทั้งหมดโดยการใช้เครื่องหมายสตาร์ (\*) เหมือนในตัวอย่างด้านบน แต่อย่างไรก็ตามนี่เป็นวิธีที่ไม่แนะนำในการใช้งาน เพราะว่าวิธีดังกล่าวนั้นจะเป็นการนำเข้าที่อาจจะเกิดความขัดแย้งกัน ดังนั้นการใช้งานในรูปแบบของ Prefix จึงเป็นวิธีฝึกปฏิบัติที่ดีสำหรับการ Import โมดูลเข้ามาในโปรแกรม และนอกจากนี้ โมดูลยังสามารถ Import เป็นแบบลำดับชั้นได้ ยกตัวอย่างเช่น โมดูล B นำเข้า โมดูล A และหลังจากนั้นโมดูล C นำเข้าโมดูล B (A -> B -> C) เป็นต้น

ในการ Import โมดูลเข้ามาในโปรแกรมนั้น Python จะทำการค้นหาไฟล์ของ module\_name.py จาก Built-in module ก่อน แล้วหลังจากนั้นโปรแกรมจะทำการค้นหาโมดูลภายในลิสต์ของ Directory ภายในตัวแปร sys.path

**Packages**

Package คือการกำหนดโครงสร้างของโมดูลในภาษา Python ที่เรียกว่า Namespace เพื่อจัดระเบียบของโมดูลต่างๆ ให้เป็นหมวดหมู่เดียวกัน แนวคิดของ Package เหมือนกับระบบจัดการไฟล์ในระบบปฏิบัติการซึ่งจะประกอบไปด้วยด้วยไฟล์และโฟล์เดอร์ โดยไฟล์ที่อยู่ในหมวดหมู่เดียวกันจะถูกเก็บไว้ในโฟล์เดอร์เดียวกัน เช่นเดียวกัน Package ใช้สำหรับจัดหมวดหมู่ให้กับโมดูล โดยโมดูลที่มีฟังก์ชันและคลาสการทำงานที่เหมือนกันจะอยู่ใน Package เดียวกัน อย่างไรก็ตาม นี่จะขึ้นกับการออกแบบของโปรแกรมเมอร์

ในการสร้าง Package คุณต้องสร้างโฟลด์เดอร์ให้มีโครงสร้างตามที่ต้องการ เนื่องจากในภาษา Python นั้น Package ก็คือโฟล์เดอร์ที่ใช้เก็บไฟล์โมดูลของโปรแกรม และสามารถซ้อนกันได้แบบลำดับชั้น นี่เป็นตัวอย่างของ Package ที่เราได้สร้างขึ้นโดยมี image เป็นรูทของ Package ภายใน Package นี้จะแบ่งย่อยออกเป็นอีกสาม Package คือ formats filters และ editและแต่ละ Package จะมีโมดูลอยู่ภายใน

image/ Top-level package
 \_\_init\_\_.py Initialize the image package
 formats/ Subpackage for file format
 \_\_init\_\_.py
 jpeg.py
 gif.py
 png.py
 ...
 filters/ Subpackage for image filters
 \_\_init\_\_.py
 blur.py
 noise.py
 render.py
 ...
 edit/ Subpackage for editing images
 \_\_init\_\_.py
 crop.py
 grayscale.py
 invert.py
 resize.py
 ...

ในโฟล์เดอร์ของใน Package มีจะมีไฟล์พิเศษที่ชื่อว่า \_\_init\_\_.py ซึ่งเป็นตัวกำหนดโมดูลภายใน Package สำหรับเพื่อให้ Python ใช้ในการค้นหา Package ภายในโฟล์เดอร์ดังกล่าวเมื่อมีการ Import ในรูปแบบ import \* และไฟล์นี้สามารถที่จะไม่มีก็ได้ มาดูตัวอย่างของไฟล์ \_\_init\_\_.py สำหรับ Package image/formats

\_\_all\_\_ = ["jpeg", "gif", "png"]

ในตัวอย่าง เราได้กำหนดค่าให้กับไฟล์ \_\_init\_\_.py สำหรับ Package image/formats ในตัวแปร \_\_all\_\_ เป็นรายการของโมดูลหรือ Package ย่อยที่จะอนุญาติให้ Python ทำการค้นหาและโหลดเข้ามาในโปรแกรม ซึ่งนี่เป็นการบอก Python ว่าโมดูลดังกล่าวนั้นจะถูก Import เมื่อมีการใช้คำสั่ง import \* และต่อไปมาดูตัวอย่างการใช้งานและการ Import โมดูลจาก Package ในภาษา Python โดยในไฟล์ image/formats/jpeg.py นั้นมีโค้ดดังต่อไปนี้

class JPEG:

 def \_\_init\_\_(self, w, h):
 self.w = w
 self.h = h
 print('JPEG image created')

 def dimension(self):
 print('Image dimension:', self.w, 'x', (self.h))

ในโมดูล jpeg ได้มีคลาส JPEG สำหรับสร้างรูปภาพประเภท jpeg เพื่อที่จะใช้งานคลาสนี้ เราจะต้องทำการ Import โมดูลดังกล่าวเข้มาในโปรแกรม ด้วยคำสั่งดังนี้

from image.formats import jpeg
from image.formats import \*

ในตัวอย่าง เป็นสองวิธีที่คุณสามารทำได้สำหรับการ Import โมดูล jpeg เข้ามาใช้งานในโปรแกรม ในแบบแรกเป็นการ Import เพียงเฉพาะโมดูล jpeg ที่กำหนด และในแบบที่สองนั้นเป็นการเลือกทั้งหมด ซึ่งนี่จะทำให้ Python ทำการ Import โมดูลที่ถูกกำหนดไว้ในไฟล์ \_\_init\_\_.py ถ้าหากมีไฟล์ดังกล่าว อย่างไรก็ตาม ในวิธีที่สองนั้นไม่แนะนำในทางปฏิบัติ เพราะคุณควรจะ Import เพียงโมดูลที่ต้องการใช้งานจริงๆ เท่านั้น ซึ่งนี่จะช่วยประหยัดหน่วยความจำได้

import image.formats.jpeg

g = jpeg.JPEG(400, 100)
g.dimension()

นี่เป็นตัวอย่างของการ Import โมดูลจาก Package และสร้างออบเจ็คจากคลาส JPEG คุณจะเห็นว่าในการเข้าถึงคลาสนั้นเรายังคงต้อง Prefix กับชื่อของโมดูลเสมอ

ในบทนี้ คุณได้เรียนรู้เกี่ยวกับโมดูลในภาษา Python และได้ทราบว่าการใช้งานโมดูลนั้นสามารถช่วยแบ่งโค้ดออกเป็นส่วนๆ และเรียกใช้งานได้เมื่อต้องการ เราได้พูดถึงการสร้างและการใช้งานโมดูลโดยการนำเข้าโมดูลด้วยคำสั่ง import และคำสั่ง from import การจัดหมวดหมู่ของโมดูลด้วย Package นี่เป็นสิ่งที่สำคัญเมื่อโปรแกรมของคุณมีขนาดใหญ่ขึ้น คุณอาจจะแบ่งมันออกเป็นส่วนๆ โดยแยกเป็นโมดูล และจัดกลุ่มของโมดูลด้วยการใช้ Package และนอกจากนี้ คุณยังสามารถสร้างไลบรารี่ของคุณ เพื่อให้นักพัฒนาคนอื่นๆ ได้ใช้งาน