บทนำ

การเขียนหรือการพัฒนาโปรแกรมคอมพิวเตอร์ในเพื่อให้ผู้ใช้งานได้ใช้งานคอมพิวเตอร์ได้อย่างง่ายดาย สมัยเริ่มแรกนักพัฒนาโปรแกรมจะพัฒนาในรูปแบบของ Text mode หรือเป็นโปรแกรมที่แสดงผลในลักษณะตัวอักษร ข้อความ แล้วให้ผู้ใช้งานโปรแกรมป้อนข้อมูลตามบรรทัดข้อความเพื่อสั่งให้คอมพิวเตอร์ทำการประมวลผล ผลลัพธ์ออกมา ซึ่งเป็นรูปแบบโปรแกรมที่มีความยุ่งยากต่อการใช้งานสำหรับผู้ใช้งานคอมพิวเตอร์ที่ไม่คุ้นเคยกับคอมพิวเตอร์มากนัก ดังนั้นต่อมาจึงได้มีการพัฒนารูปแบบโปรแกรมใหม่ เป็นโปรแกรมที่มีการแสดงผลด้วยลักษณะกราฟิกจึงทำให้สามารถสื่อสารกับผู้ใช้งานโปรแกรมได้ง่าย โดยอยู่ในรูปแบบกราฟิก และนักโปรแกรมเมอร์นิยมพัฒนาโปรแกรมโดยใช้เครื่องมือการพัฒนาในรูปแบบของ Visual ซึ่งหมายความว่า นักพัฒนาโปรแกรมต้องการให้รูปแบบหน้าจอโปรแกรมแสดงผลอย่างไร ก็สามารถใช้เครื่องมือในการพัฒนาโปรแกรมลาก และวางได้ (Drag and Drop) ตามที่ต้องการ

ฟังก์ชัน (Function) เป็นการทำงานที่คล้ายกับโปรแกรมย่อย คือ เป็นส่วนย่อยของโปรแกรมหลัก สร้างขึ้นเพื่อให้โปรแกรมหลักเรียกใช้งาน แต่แตกต่างจากโปรแกรมย่อยในส่วนของการส่งค่ากลับออกมา (Return Value) อธิบายคือ เมื่อโปรแกรมย่อยประมวลผลเสร็จ จะไม่มีการส่งค่ากลับออกมาจากโปรแกรม ย่อยนั้น แต่ฟังก์ชันเมื่อมีการประมวลผลเสร็จแล้ว จะมีการส่งค่ากลับออกมาจากฟังก์ชันนั้น

การเขียนโปรแกรมในภาษา Python โมดูล (Module) คือไฟล์ของโปรแกรมที่กำหนดตัวแปร ฟังก์ชัน หรือคลาสโดยแบ่งย่อยออกไปจากโปรแกรมหลัก และสามารถนำมาใช้งานได้โดยการนำเข้ามาในโปรแกรม (Import) กล่าวอีกนัยหนึ่ง โมดูลก็คือไลบรารี่ที่สร้างไว้และนำมาใช้งานในโปรแกรม ในบทนี้ เราจะพูดถึงความหมายของโมดูล การสร้าง และการใช้งานโมดูลในการเขียนโปรแกรม

Package คือการกำหนดโครงสร้างของโมดูลในภาษา Python ที่เรียกว่า Namespace เพื่อจัดระเบียบของโมดูลต่าง ๆ ให้เป็นหมวดหมู่เดียวกัน แนวคิดของ Package เหมือนกับระบบจัดการไฟล์ในระบบปฏิบัติการซึ่งจะประกอบไปด้วยด้วยไฟล์ และโฟล์เดอร์ โดยไฟล์ที่อยู่ในหมวดหมู่เดียวกันจะถูกเก็บไว้ในโฟล์เดอร์เดียวกัน เช่นเดียวกัน Package ใช้สำหรับจัดหมวดหมู่ให้กับโมดูล โดยโมดูลที่มีฟังก์ชันและคลาสการทำงานที่เหมือนกันจะอยู่ใน Package เดียวกัน อย่างไรก็ตาม นี่จะขึ้นกับการออกแบบของโปรแกรมเมอร์ และในภาษา Python คุณสามารถสร้างฟังก์ชันของคุณเองเพื่อให้ทำงานที่ต้องการ ในการเขียนโปรแกรมเรามักจะแยกโค้ดที่มีการทำงานเหมือนๆ กันเป็นฟังก์ชันเอาไว้และเรียกใช้ฟังก์ชันนั้นซ้ำ ๆ ซึ่งเป็นแนวคิดของการ reuse โค้ด นี่เป็นรูปแบบของการประกาศฟังก์ชันในภาษา Python

1.Built-in Function in Python

ฟังก์ชัน (Function) คือส่วนของโค้ดหรือโปรแกรมที่ทำงานเพื่อวัตถุประสงค์บางอย่าง ในภาษา Python คุณสามารถสร้างฟังก์ชันของคุณเองเพื่อให้ทำงานที่ต้องการ ในการเขียนโปรแกรม เรามักจะแยกโค้ดที่มีการทำงานเหมือนๆ กันเป็นฟังก์ชันเอาไว้ และเรียกใช้ฟังก์ชันนั้นซ้ำ ๆ ซึ่งเป็นแนวคิดของการ reuse โค้ด นี่เป็นรูปแบบของการประกาศฟังก์ชันในภาษา Python

```
def function_name(args...):
    # statements

def function_name(args...):
    # statements
    return value
```

ในรูปแบบของการประกาศฟังก์ชันในภาษา Python นั้นจะใช้คำสั่ง def และหลังจากนั้น function_name เป็นชื่อของฟังก์ชัน และในวงเล็บ () เป็นการกำหนดพารามิเตอร์ของฟังก์ชัน พารามิเตอร์ของฟังก์ชันนั้นสามารถมีจำนวนเท่าไหร่ก็ได้หรือไม่มีก็ได้ และเช่นเดียวกับภาษาอื่น ๆ ฟังก์ชันอาจจะมีหรือไม่มีการส่งค่ากลับ สำหรับฟังก์ชันที่ไม่มีการ return ค่ากลับนั้น เรามักจะเรียกว่าโพรซีเยอร์ (Procedure) ต่อไปมาดูตัวอย่างการประกาศและใช้งานฟังก์ชันในภาษา Python

```
def hello(name):
    print('Hello %s' % name)

def count_vowel(str):
    vowel = 0
    for c in str:
        if c in ('A', 'E', 'I', 'O', 'U', 'a', 'e', 'i', 'o', 'u'):
            vowel = vowel + 1
    return vowel

def area(width, height):
    c = width * height
    return c
```

```
def count_vowel(str):
    vowel = 0
    for c in str:
        if c in ('A', 'E', 'I', 'O', 'U', 'a', 'e', 'i', 'o', 'u'):
            vowel = vowel + 1
    return vowel
```

ในตัวอย่าง เราได้สร้างฟังก์ชันจำนวน 3 ฟังก์ชัน ฟังก์ชันแรกมีชื่อว่า hello() เป็นฟังก์ชันสำหรับแสดงข้อความทักทายจากที่ชื่อส่งเข้ามา ฟังก์ชันนี้มีหนึ่งพารามิเตอร์คือ name สำหรับรับชื่อที่ส่งเข้ามาในฟังก์ชัน

ต่อมาฟังก์ชัน count_vowel() เป็นฟังก์ชันสำหรับนับจำนวนสระใน String ฟังก์ชันนี้มีหนึ่ง String พารามิเตอร์ ในการทำงานของฟังก์ชันนั้นเราใช้คำสั่ง For loop ในการวนอ่านค่าทีละตัวอักษรเพื่อตรวจสอบว่าเป็นสระหรือไม่ด้วยคำสั่ง in และตัวแปร vowel นั้นใช้สำหรับนับจำนวนสระที่พบใน String ในตอนท้ายเราได้ส่งค่าของจำนวนสระที่นับได้กลับไปด้วยคำสั่ง return

```
def area(width, height):
    c = width * height
    return c
```

และฟังกชันสุดท้ายคือฟังก์ชัน area() เป็นฟังก์ชันสำหรับหาพื้นที่ของรูปสี่เหลี่ยมด้านขนานและฟังก์ชันมีพารามิเตอร์สองตัวสำหรับความกว้างและความยาวของสี่เหลี่ยม และฟังก์ชันทำการ return ผลลัพธ์ที่เป็นพื้นที่กลับไปด้วยคำสั่ง return

การเรียกใช้งานฟังก์ชันในภาษา Python

หลังจากเราได้สร้างฟังก์ชันในก่อนหน้าแล้ว ต่อไปเราจะมาเรียกใช้งานฟังก์ชันเหล่านั้น โดยการเรียกใช้ฟังก์ชันนั้นเราจะใช้ชื่อของฟังก์ชันและส่งอาร์กิวเมนต์ให้สอดคล้องกับพารามิเตอร์ที่กำหนดไว้ในฟังก์ชัน ดังนั้นอาร์กิวเมนต์คือค่าที่ส่งเข้าไปในฟังก์ชันตอนใช้งาน ส่วนพารามิเตอร์นั้นคือตัวแปรที่กำหนดไว้ในฟังก์ชันเพื่อรับค่าจากอาร์กิวเมนต์ มาดูตัวอย่างการเรียกใช้งานฟังก์ชันในภาษา Python

```
def hello(name):
    print('Hello %s' % name)

def count_vowel(str):
    vowel = 0
    for c in str:
        if c in ('A', 'E', 'I', 'O', 'U', 'a', 'e', 'i', 'o', 'u'):
            vowel = vowel + 1
    return vowel

def area(width, height):
    c = width * height
    return c

# calling functions
hello('Danny')
hello('Mateo')
print('Vowel in string = %d' % count_vowel('marcuscode.com'))
print('Vowel in string = %d' % count_vowel('Python'))
print('Area = %d' % area(8, 4))
```

ในตัวอย่าง เป็นการเรียกใช้งานฟังก์ชันที่เราสร้างขึ้น เราได้เรียกใช้ฟังก์ชัน hello() และส่งอาร์กิวเมนต์ที่เป็น String เข้าไปยังฟังก์ชัน เราเรียกใช้ฟังก์ชันนี้สองครั้ง ซึ่งนี่เองเป็นการ reuse โค้ดในการเขียนโปรแกรม

หลังจากนั้นเราเรียกใช้ฟังก์ชัน count_vowel() และฟังก์ชัน area() และส่งพารามิเตอร์ที่ถูกต้องไปยังฟังก์ชัน และเพราะว่าฟังก์ชันเหล่านี้มีการ return ค่ากลับ เราสามารถนำค่าเหล่านี้ไปใช้งานได้ต่อไป เราได้นำไปใช้กับฟังก์ชัน print() เพื่อจัดรูปแบบการแสดงผล

```
Hello Danny
Hello Mateo
Vowel in string = 5
Vowel in string = 1
Area = 32
```

เป็นผลลัพธ์การทำงานของโปรแกรม จากการเรียกใช้งานฟังก์ชันในภาษา Python

Default Argument Values

ในภาษา Python เราสามารถสร้างฟังก์ชันโดยการกำหนด Default Argument ให้กับฟังก์ชันพารามิเตอร์ได้ Default Argument เป็นการการกำหนดค่าเริ่มต้นให้กับอาร์กิวเมนต์ที่ส่งเข้ามายังฟังก์ชัน นั่นทำให้เราสามารถเรียกใช้งานฟังก์ชันโดยการส่งอาร์กิวเมนต์น้อยกว่าจำนวนที่กำหนดไว้ในฟังก์ชันได้ ซึ่งอำนวยความสะดวกในการใช้งานมากขึ้น มาดูตัวอย่างการสร้างและใช้งานฟังก์ชันกับ Default Argument

```
def show_info(name, salary = 84360, lang = "Python"):
    print('Name: %s' % name)
    print('Salary: %d' % salary)
    print('Language: %s' % lang)
    print()
```

ในตัวอย่าง เราได้สร้างฟังก์ชัน show_info() สำหรับแสดงข้อมูลของโปรแกรมเมอร์ ข้อมูลที่จำเป็นต้องการจะแสดงนั้นมีชื่อ เงินเดือน และภาษาที่เขียน ในฟังก์ชันของเรานั้นมี 3 พารามิเตอร์ พารามิเตอร์แรก name นั้นเป็นพารามิเตอร์แบบปกติ และสองพารามิเตอร์นั้นเป็น Default Argument ซึ่งเรากำหนดค่าเริ่มต้นให้กับพารามิเตอร์โดยใช้เครื่องหมาย = ในการกำหนดพารามิเตอร์นั้น Default Argument ต้องอยู่หลังพารามิเตอร์แบบปกติเสมอ

```
# calling function
show_info('Mateo')
show_info('Mateo', 105000)
show_info('Danny', 120000, 'Java')
```

ในการเรียกใช้งานฟังก์ชันนั้น เราต้องทำการส่งค่าอาร์กิวเมนต์สำหรับพารามิเตอร์แบบปกติเสมอ ส่วนพารามิเตอร์แบบ Default Argument นั้นเป็นทางเลือก ในตัวอย่าง คำสั่งเราเรียกใช้ฟังก์ชันโดยอาร์กิวเมนต์เพียงหนึ่งตัวเข้าไป ทำให้สองอาร์กิวเมนต์ที่เหลือที่เป็น Default Argument ใช้ค่าเริ่มต้นของมันแทน คือ 84360 สำหรับเงินเดือน และ "Python" สำหรับภาษาเขียนโปรแกรม

ต่อมาเราเรียกใช้ฟังก์ชันโดยการส่งสองอาร์กิวเมนต์เข้าไป ทำให้มีเพียงพารามิเตอร์สุดท้ายเท่านั้นที่ใช้ค่าเริ่มต้น และในคำสั่งสุดท้ายเป็นการส่งค่าครบจำนวนให้กับทุกอาร์กิวเมนต์

```
Name: Mateo
Salary: 84360
Language: Python

Name: Mateo
Salary: 105000
Language: Python

Name: Danny
Salary: 120000
Language: Java
```

นี่เป็นผลลัพธ์การทำงานของโปรแกรม ในการเรียกใช้งานฟังก์ชันกับ Default Argument

Keyword Arguments

ในภาษา Python เราสามารถเรียกใช้งานฟังก์ชันในรูปแบบของ Keyword Argument โดยการใช้ชื่อของพารามิเตอร์สำหรับส่งอาร์กิวเมนต์ ในการใช้งานนั้น พารามิเตอร์ต้องมีการกำหนดในรูปแบบของ Default Argument ก่อน มาดูตัวอย่างการใช้งาน Keyword Arguments ในภาษา Python

```python
def create_button(id, color = '#ffffff', text = 'Button', size = 16):
    print('Button ID: %d' % id)
    print('Attributes:')
    print('Color: %s' % color)
    print('Text: %s' % text)
    print('Size: %d px' % size)
    print()

create_button(10)
create_button(11, color = '#4286f4', text = 'Sign up')
create_button(id = 12, color = '#323f54', size = 24)
create_button(color = '#1cb22b', text = 'Log in', size = 32, id = 13)
```

ในตัวอย่าง เราได้สร้างฟังก์ชันสำหรับการสร้างปุ่ม ในการเรียกใช้งานฟังก์ชันนั้น เราสามารถเรียกโดยวิธีการส่งแบบ Keyword Argument ได้ในรูปแบบของ argument = value และสามารถสลับตำแหน่งของอาร์กิวเมนต์ได้ และในฟังก์ชันนั้นเรามีอาร์กิวเมนต์ id แบบซึ่งเป็นอาร์กิวเมนต์แบบปกติ ในการส่งค่านั้นต้องส่งเป็นลำดับแรกเสมอ เหมือนในคำสั่งการเรียกใช้งานสองอันแรก หรือสามารถส่งแบบ Keyword Argument ก็ได้เช่นกันเหมือนในคำสั่งที่สามและสี่

นี่เป็นผลลัพธ์การทำงานของโปรแกรม เราได้เรียกใช้งานฟังก์ชันเพื่อสร้างปุ่ม 4 ปุ่มในรูปแบบต่างๆ ของการใช้ Keyword Argument

```
Button ID: 10
Attributes:
Color: #ffffff
Text: Button
Size: 16 px

Button ID: 11
Attributes:
Color: #4286f4
Text: Sign up
Size: 16 px

Button ID: 12
Attributes:
Color: #323f54
Text: Button
Size: 24 px

Button ID: 13
Attributes:
Color: #1cb22b
Text: Log in
Size: 32 px
```

ตัวอย่างของฟังก์ชันที่มีการใช้งานในรูปแบบของ Keyword Argument ก็คือฟังก์ชัน print() เราสามารถเปลี่ยนตัวคั่นระหว่างอาร์กิวเมนต์ และการแสดงผลลัพธ์ในตอนท้ายของฟังก์ชันได้ โดยการใช้ Keyword sep และ end ตามลำดับ

```
print(1, 2, 3)
print(1, 2, 3, sep = '-', end = '/')
```

Lambda Expressions

Lambda Expressions คือ anonymous function ที่เป็นฟังก์ชันที่มีการทำงานขนาดเล็กอยู่ภายในที่สามารถมีได้เพียง Expression เดียวเท่านั้น เราสามารถสร้างโดยใช้คำสั่ง lambda เราสามารถใช้ Lambda Expressions สร้างออบเจ็คของฟังก์ชันได้ และค่า return จะเป็นค่าที่ได้จากผลลัพธ์ของ Expression ของฟังก์ชัน มาดูตัวอย่างการใช้งาน

```
f = lambda x: x + 1
print(f(2))
print(f(8))

g = lambda a, b: (a + b) / 2
print(g(3, 5))
print(g(10, 33))

def make_incrementor(n):
    return lambda x: x + n

f = make_incrementor(13)
print(f(0))
print(f(1))
print(f(5))
```

ในตัวอย่าง เราได้สร้าง Lambda Expressions เป็นจำนวนสามฟังก์ชัน ฟังก์ชันแรกเป็นฟังก์ชันสำหรับเพิ่มตัวเลขขึ้น 1 และฟังก์ชันที่สองเป็นฟังก์ชันสำหรับหาค่าเฉลี่ยของตัวเลขสองจำนวน คุณจะสังเกตุได้ว่าฟังก์ชันแรกนั้นมี 1 อาร์กิวเมนต์และฟังก์ชันที่สองนั้นมี 2 อาร์กิวเมนต์ และฟังก์ชันสุดท้ายนั้นเป็นการ return ฟังก์ชันกลับภายในฟังก์ชันอีกที และเป็นฟังก์ชันสำหรับเพิ่มตัวเลขขึ้นจำนวน n จากอาร์กิวเมนต์ที่ใส่เข้าไป

```
3
9
4.0
21.5
13
14
18
```

นี่เป็นผลลัพธ์การทำงานของโปรแกรม

นอกจากนี้ Lambda Expressions ยังมีประโยชน์เพื่อใช้งานกับ built-in function เช่น ฟังก์ชัน filter() และฟังก์ชัน map() ในภาษา Python มันใช้เป็นอาร์กิวเมนต์ส่งเข้าไปในฟังก์ชัน เพื่อสร้าง Expression ให้กับฟังก์ชัน มาดูตัวอย่างการใช้งาน

```
numbers = [2, 15, 5, 7, 10, 3, 28, 30]
print(list(filter(lambda x: x % 5 == 0, numbers)))
print(list(map(lambda x: x * 2, numbers)))
```

ในตัวอย่าง เรามีลิสต์ของตัวเลข Integer และเราได้ใช้ฟังก์ชัน filter() และฟังก์ชัน map() ซึ่งเป็นฟังก์ชันที่มีอาร์กิวเมนต์ตัวแรกเป็นฟังก์ชัน และตัวที่สองเป็นลิสต์ ในการทำงานของฟังก์ชัน filter() เราได้ใช้ฟังก์ชันกรองเอาตัวเลขที่ตรงกันกับ Lambda Expressions ซึ่งก็คือตัวเลขในลิสต์ที่หารด้วย 5 ลงตัว และในการใช้ฟังก์ชัน map() เป็นการเชื่อมโยงค่าในลิสต์ให้ตรงกับ Lambda Expressions คือการเพิ่มตัวเลขให้เป็นสองเท่า ซึ่งทั้งสองฟังก์ชันนี้ส่งค่ากลับเป็นออบเจ็ค และเราใช้ฟังก์ชัน list() เพื่อแปลงออบเจ็คให้เป็นสิสต์

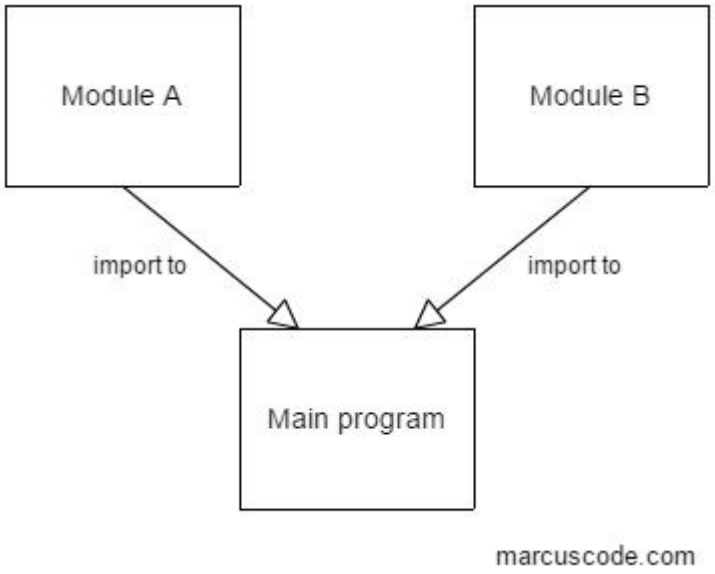
```
[15, 5, 10, 30]
[4, 30, 10, 14, 20, 6, 56, 60]
```

นี่เป็นผลลัพธ์การทำงานของโปรแกรม

2. Module in Python

ในการเขียนโปรแกรมในภาษา Python โมดูล (Module) คือไฟล์ของโปรแกรมที่กำหนดตัวแปร ฟังก์ชัน หรือคลาสโดยแบ่งย่อยออกไปจากโปรแกรมหลัก และสามารถนำมาใช้งานได้โดยการนำเข้ามาในโปรแกรม (Import) กล่าวอีกนัยหนึ่ง โมดูลก็คือไลบรารี่ที่สร้างไว้และนำมาใช้งานในโปรแกรม ในบทนี้ เราจะพูดถึงความหมายของโมดูล การสร้าง และการใช้งานโมดูลในการเขียนโปรแกรม

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



คุณจะเห็นว่าโมดูลก็คือการแยกส่วนของโปรแกรมออกไปเป็นอีกส่วนและสามารถเรียกใช้ได้เมื่อต้องการ หรือกล่าวอีกนัยหนึ่ง โมดูลก็เหมือนไลบรารีของฟังก์ชันและคลาสต่าง ๆ นั่นเป็นเพราะว่าเมื่อโปรแกรมของคุณมีขนาดใหญ่ คุณสามารถแบ่งส่วนต่าง ๆ ของโปรแกรมออกเป็นโมดูลย่อย ๆ เพื่อให้ง่ายต่อการจัดการและการใช้งาน ในภาษา Python โมดูลที่ถูกสร้างขึ้นมานั้นจะเป็นไฟล์ในรูปแบบ module_name.py และนอกจากนี้ Python ยังมี Built-in module เป็นจำนวนมาก เช่น math เป็นโมดูลเกี่ยวกับฟังก์ชันทางคณิตศาสตร์ หรือ random เป็นโมดูลเพื่อจัดการและสุ่มตัวเลข เป็นต้น

Modules ใน Python

โมดูลใน ไพทอน คือกลุ่มของ ตัวแปล ฟังก์ชั่น หรือ คลาส ที่ทำงานคล้ายๆ กันแล้วเอามารวมกันไว้ในไฟล์ไฟล์เดียว ในไพทอน เวลาเราจะเรียกใช้ โมดูลเราต้อง import โมดูลเข้ามาก่อนถึงจะสามารถเรียกใช้งาน ฟังก์ชั่น หรือ คลาสที่อยู่ ภายในโมดูลได้มีโมดูลอยู่สองประเภทคือ

1. โมดูลที่ไพทอนเตียมไว้ให้สามารถ import เข้ามาใช้งานได้เลย
2. โมดูลที่เราสร้างขึ้นมาเอง

การสร้างโมดูลในภาษา Python

ในการสร้างโมดูลในภาษา Python คุณต้องนำโค้ดของโปรแกรม โดยทั่วไปแล้วจะประกอบไปด้วย ตัวแปร ฟังก์ชัน หรือคลาส ไปรวมไว้ในไฟล์ใหม่ที่ไม่ใช้ไฟล์หลักของโปรแกรม ในรูปแบบ module_name.py โดยที่ module_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 ในการใช้งานฟังก์ชันในโมดูลนั้นสามารถทำได้โดยใช้ชื่อของโมดูลตามด้วยเครื่องหมายจุด 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 นั้น

มีคำสั่ง from import สำหรับนำเข้าข้อมูลบางส่วนภายในโมดูล และสามารถใช้งานออบเจ็คได้โดยตรงโดยไม่ต้องมี Prefix ชื่อของโมดูล มาดูตัวอย่างการใช้งาน

```
from number import factorial

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

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

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

```
from number import factorial, fibonacci

# or

from number import *
```

ในอีกทางหนึ่ง คุณสามารถนำเข้าหลายออบเจ็ค (ฟังก์ชัน) ในโมดูลโดยการใช้เครื่องหมายคอมม่า (,) 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

3.Packages in Python

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 Importในรูปแบบimport * 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 โมดูลดังกล่าวเข้ามาในโปรแกรม 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 ถ้าหากมี

ไฟล์ดังกล่าว อย่างไรก็ตาม ในวิธีที่สองนั้นไม่แนะนำในทางปฏิบัติ เพราะคุณควรจะ Import เพียงโมดูลที่ต้องการใช้งานจริงๆ เท่านั้น ซึ่งนี่จะช่วยประหยัดหน่วยความจำได้

```
import image.formats.jpeg
g = jpeg.JPEG(400, 100)
g.dimension()
```

นี่เป็นตัวอย่างของการ Import โมดูลจาก Package และสร้างออปเจ็กจากคลาส JPEG คุณจะเห็นว่าในการเข้าถึงคลาสนั้นเรายังคงต้อง Prefix กับชื่อของโมดูลเสมอ

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

บรรณานุกรม

1. http://marcuscode.com/lang/python/functionsการสร้างฟังก์ชันในภาษา Python
2. http://marcuscode.com/lang/pythonภาษาPython
3. http://marcuscode.com/lang/python/modules
4. https://sites.google.com/site/karkheiynpormkaerm/fangkchan-function
5. https://sites.google.com/site/dotpython/
6. https://sites.google.com/site/pythonclassroom/module-package-python/module-python-is โมดูลไพธอน และแพคเกจไพธอน
7. https://www.mindphp.com/
8. https://sites.google.com/site/introductiontoprogrammingc/hnathi-5---fangkchan-functions-laea-porkaerm-yaek-pen-modul-modular-programming
9. https://www.mindphp.com/-python.html