



รายงาน

วิชา การโปรแกรมคอมพิวเตอร์

เรื่อง 1.Built-in Function in Python

2.Module and Package in Python

โดย

นายเกรียงไกร เพ็ญจันทร์	รหัสนักศึกษา 362516232004
น.ส จิราวรรณ ชัยนอก	รหัสนักศึกษา 362516232006
นายวัลลภ พุดบุญ	รหัสนักศึกษา 362516232012
นายสรเรณู แสงบุญนำ	รหัสนักศึกษา 362516232016
นายภัทรพงษ์ ทองย้อย	รหัสนักศึกษา 362516232025

กลุ่มเรียน EE36231E ระดับ ปริญญาตรี สมทบ

เสนอ

อาจารย์ ดร.สุรพล ไรจนประดิษฐ์

คณะวิศวกรรมศาสตร์ สาขาวิศวกรรมไฟฟ้า ปีการศึกษา 2562

มหาวิทยาลัยเทคโนโลยีราชมงคลสุวรรณภูมิ

คำนำ

รายงานเล่มนี้มีความเกี่ยวข้องกับวิชาการเขียนโปรแกรมคอมพิวเตอร์ และเนื้อหาภายในเล่มนำเสนอความรู้เกี่ยวกับเรื่อง Built-in Function in Python และ Module and Package in Python ซึ่งมีที่เกิเกิดขึ้นใหม่อย่าง ภาษาไพธอน จึงทำให้การเขียนโปรแกรมมีความซับซ้อน และยังใช้เวลาในการพัฒนามากขึ้นอีกด้วยเมื่อเปรียบเทียบกับภาษาคอมพิวเตอร์ทั่ว ๆ ไปที่นำมาใช้ในการเรียนการสอนในระดับวิทยาลัยหรือมหาวิทยาลัย เช่น ภาษา Pascal ภาษาซี ภาษาจาวา เป็นต้น

ซึ่งทางคณะผู้จัดทำหวังเป็นอย่างยิ่งว่ารายงานฉบับนี้จะเป็นประโยชน์สำหรับผู้สนใจศึกษาค้นคว้า และประสงค์ที่จะเรียนรู้ด้วยตนเองผ่านรายงานเล่มนี้

คณะผู้จัดทำ

1 ตุลาคม 2562

สารบัญ

เรื่อง	หน้า
คำนำ	
สารบัญ	
บทนำ	1
1. Built-in Function in Python	2
2. Python – modules	9
3. Python – Package	18
บรรณานุกรม	22

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 โมดูลก็คือไลบรารีที่สร้างไว้และนำมาใช้งานในโปรแกรม ในบทนี้ เราจะพูดถึงความหมายของโมดูล การสร้าง และการใช้งานโมดูลในการเขียนโปรแกรม

Package คือการกำหนดโครงสร้างของโมดูลในภาษา Python ที่เรียกว่า Namespace เพื่อจัดระเบียบของโมดูลต่าง ๆ ให้เป็นหมวดหมู่เดียวกัน แนวคิดของ Package เหมือนกับระบบจัดการไฟล์ในระบบปฏิบัติการซึ่งจะประกอบไปด้วยด้วยไฟล์ และโฟลเดอร์ โดยไฟล์ที่อยู่ในหมวดหมู่เดียวกันจะถูกเก็บไว้ในโฟลเดอร์เดียวกัน เช่นเดียวกัน Package ใช้สำหรับจัดหมวดหมู่ให้กับโมดูล โดยโมดูล ที่มีฟังก์ชันและคลาสการทำงานที่เหมือนกันจะอยู่ใน Package เดียวกัน อย่างไรก็ตาม นี่จะขึ้นกับการออกแบบของโปรแกรมเมอร์ และในภาษา Python คุณสามารถสร้างฟังก์ชันของคุณเองเพื่อให้ทำงาน ที่ต้องการ ในการเขียนโปรแกรมเรามักจะแยกโค้ดที่การทำงานเหมือนกันเป็นฟังก์ชันเอาไว้ และเรียกใช้ฟังก์ชันนั้นซ้ำ ๆ ซึ่งเป็นแนวคิดของการ reuse โค้ด นี่เป็นรูปแบบของการประกาศฟังก์ชันในภาษา Python

1. Built-in function in Python

เป็นกลุ่มฟังก์ชันที่มากับ Python สามารถเรียกใช้งานได้เลย See a list of the functions next page.

User defined

เป็นฟังก์ชันที่ผู้ใช้สร้างขึ้นมาเองสำหรับงานที่เฉพาะเจาะจงมากขึ้นซึ่ง ในบทนี้เราจะเรียนรู้การเรียกใช้ฟังก์ชันแบบ built-in ก่อน

โดยทั่วไปแล้ว Function specification มีข้อกำหนดคุณสมบัติการทำงานของโปรแกรมที่มีหน้าตาต่างได้ตอบ และกล่องโต้ตอบกับผู้ใช้จะแสดงลักษณะที่ปรากฏของส่วนต่อประสานผู้ใช้ (UI) และอธิบายการกระทำของผู้ใช้ และการตอบสนองของโปรแกรมซึ่งฟังก์ชันนั้นที่ต้องการ argument จำนวนกี่ตัว และมีลำดับการส่ง argument อย่างไร

ค่า return แบ่งเป็น 2 แบบคือ

1. ค่า return ของ function เช่น function absolute(\$value) เป็น function ที่ทำให้ค่า \$value กลายเป็นค่า สัมบูรณ์ ดังนั้น function absolute() จะต้อง return ค่าที่เป็นค่าสัมบูรณ์เพื่อเอาไปใช้ต่อ

2. ค่า return ของโปรแกรมตัวโปรแกรมทุกโปรแกรมจะ return ค่าเมื่อทำงานเสร็จสิ้น เช่น rasdial บน windows ถ้าทำการเชื่อมต่อไปยังปลายทางสำเร็จจะ return ค่า 0 ถ้าไม่สำเร็จจะ return ค่าอื่น เช่น 678 Return value เมื่อฟังก์ชันทำงานเสร็จสิ้นจะส่งผลลัพธ์อะไรกลับมา

Built-in Functions				
abs()	dict()	help()	min()	setattr()
all()	dir()	hex()	next()	slice()
any()	divmod()	id()	object()	sorted()
ascii()	enumerate()	input()	oct()	staticmethod()
bin()	eval()	int()	open()	str()
bool()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	
delattr()	hash()	memoryview()	set()	

ฟังก์ชันสำหรับแปลงชนิดข้อมูล ใน ไพทอน Python Data Type Conversion

ในภาษา Python คุณสามารถใช้ built-in ฟังก์ชันที่มีอยู่สำหรับแปลงประเภทข้อมูล โดยฟังก์ชันเหล่านั้นจะมีชื่อที่เหมือนกับประเภทของมัน (ซึ่งเป็นชื่อของคลาส) ยกตัวอย่างเช่น ฟังก์ชัน `int()` ใช้แปลงข้อมูลประเภทใดๆ ให้เป็นตัวเลขจำนวนเต็ม และ `str()` ใช้แปลงข้อมูลประเภทใดๆ ให้เป็น String นี่เป็นตารางของฟังก์ชันสำหรับแปลงข้อมูลในภาษา Python

Function	Description
<code>int(x [,base])</code>	แปลงออบเจ็กต์ x จากฐานที่กำหนด base ให้เป็น Integer
<code>long(x [,base])</code>	แปลงออบเจ็กต์ x จากฐานที่กำหนด base ให้เป็น Long
<code>float(x)</code>	แปลงออบเจ็กต์ x ให้เป็น Floating point number
<code>complex(real [,im])</code>	สร้างตัวเลขจำนวนเชิงซ้อนจากค่า real และค่า imagine
<code>str(x)</code>	แปลงออบเจ็กต์ x ให้เป็น String
<code>repr(x)</code>	แปลงออบเจ็กต์ x ให้เป็น String expression
<code>eval(str)</code>	ประเมินค่าของ String
<code>tuple(s)</code>	แปลง Sequence ให้เป็น Tuple
<code>list(s)</code>	แปลง Sequence ให้เป็น List
<code>set(s)</code>	แปลง Sequence ให้เป็น Tuple
<code>dict(d)</code>	แปลงออบเจ็กต์ให้เป็น Dictionary
<code>frozenset(s)</code>	แปลงออบเจ็กต์ให้เป็น Frozen set
<code>chr(x)</code>	แปลงค่าของ Integer ให้เป็น Unicode Char
<code>ord(x)</code>	แปลง Character ให้เป็นค่า Integer
<code>hex(x)</code>	แปลง Integer ให้เป็น Hex string
<code>oct(x)</code>	แปลง Integer ให้เป็น Oct string

ฟังก์ชัน dir()

ฟังก์ชัน dir() ในไพทอน เป็น built-in การสร้างฟังก์ชัน ในไพทอน Python Defining function ฟังก์ชัน

ซึ่ง ฟังก์ชัน dir() จะคืนค่าเป็น string ที่เก็บชื่อที่ถูกกำหนดขึ้นใน module ชื่อที่คืนค่าออกมา ประกอบด้วย ชื่อ module , ตัวแปร, ฟังก์ชันที่ถูกกำหนดขึ้นภายใน module

ตัวอย่าง

```
# ตัวอย่าง สอน Python ในเว็บ Mindphp.com
# ทำความรู้จักฟังก์ชัน dir()
# Import built-in module math
import math

content = dir(math)

print (content)
```

ผลที่ได้

```
['__doc__', '__name__', '__package__', 'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'copysign',
'cos', 'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum', 'gamma',
'hypot', 'isfinite', 'isinf', 'isnan', 'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'modf', 'pi', 'pow', 'radians', 'sin', 'sinh',
'sqrt', 'tan', 'tanh', 'trunc']
```

ฟังก์ชัน globals() locals() ในไพทอน

ฟังก์ชัน globals(), locals() จะคืนค่าจะคืนค่า global และ local namespaces ซึ่งจะขึ้นอยู่กับตำแหน่งที่เรียก

ถ้าฟังก์ชัน locals() ถูกเรียก ภายในฟังก์ชัน จะคืนค่าชื่อทั้งหมดที่สามารถเรียกใช้ได้ภายในฟังก์ชัน

ถ้าฟังก์ชัน globals() ถูกเรียก ภายในฟังก์ชัน จะคืนค่าชื่อทั้งหมดที่สามารถเรียกใช้ได้ภายในฟังก์ชันนั้น

ตัวอย่างไฟล์ function_locals_globals.py

```
# ตัวอย่าง สอน Python ในเว็บ Mindphp.com
# การใช้งาน locals และ globals เพื่อหา namespaces
varpython = 2000
def Addvarmindpython():
    # Uncomment the following line to fix the code:
    global varpython
    b = 'test'
    g = globals()
    print(g)
    print('-----')
    l = locals()
    print(l)
    varpython = varpython + 1

print (varpython)
Addvarmindpython ()
print (varpython)
```

ผลที่ได้

2000

```
{'__builtins__': <module 'builtins' (built-in)>, '__file__': 'D:\\function_locals_globals.py', 'Addvarmindpython':
<function Addvarmindpython at 0x0000000002677248>, '__package__': None, 'varpython': 2000, '__cached__':
None, '__name__': '__main__', '__doc__': None}
```

```
{'b': 'test', 'g': {'__builtins__': <module 'builtins' (built-in)>, '__file__': 'D:\\function_locals_globals.py',
'Addvarmindpython': <function Addvarmindpython at 0x0000000002677248>, '__package__': None,
'varpython': 2000, '__cached__': None, '__name__': '__main__', '__doc__': None}}
```

2001

ฟังก์ชัน len()

เป็นคำสั่งสำหรับใช้วัดความยาวของตัวอักษร ใช้ได้กับทั้งสตริง, ไลสต์, ทูเพิล, List, หรือ range มีรูปแบบไวยากรณ์ดังนี้

len(ตัวแปรหรือข้อมูล)

ตัวอย่างเช่น

```
>>> a = "123456"
```

```
>>> len(a)
```

```
6
```

max

เป็นคำสั่งสำหรับใช้หาค่าที่มากที่สุดในชนิดข้อมูลนั้น ๆ มีรูปแบบไวยากรณ์ดังนี้

max(ตัวแปรหรือข้อมูล)

ตัวอย่างเช่น

```
>>> max(1,2,3)
```

```
3
```

```
>>> max([1,2,3,4,5,6,-1])
```

```
6
```

```
>>> max('abca') # ตัวอักษร c มีค่ามากที่สุดถ้าเรียงตามลำดับตัวอักษร
```

```
'c'
```

min

เป็นคำสั่งสำหรับใช้หาค่าน้อยที่สุดในชนิดข้อมูลนั้น ๆ มีรูปแบบไวยากรณ์ดังนี้

min(ตัวแปรหรือข้อมูล)

ตัวอย่างเช่น

```
>>> min('abca')
```

```
'a'
```

```
>>> min([1,2,3,4,5,6,-1])
```

```
-1
```

```
>>> min(1,2,3)
```

```
1
```


zip

zip(ทูเพิล1,ทูเพิล2) เป็นคำสั่งสำหรับใช้รวมทูเพิล (tuples) 2 อัน จับคู่เป็นทูเพิลเดียวกันตัวอย่างเช่น

```
a = ["a", "b", "c"]
```

```
b = [1, 2, 3]
```

```
c = zip(a,b)
```

```
print(list(c))
```

ผลลัพธ์

```
[('a', 1), ('b', 2), ('c', 3)]
```

lambda

lambda สามารถใช้คำสั่งนี้สร้างฟังก์ชันไม่ระบุชื่อได้ มักใช้ในการสร้างฟังก์ชันที่ใช้งานเพียงครั้งเดียว สามารถใช้พารามิเตอร์ และส่งกลับค่าของนิพจน์ได้

มีรูปแบบไวยากรณ์ดังนี้

lambda <input>: <expression>ตัวอย่างเช่น

```
f = lambda x: x**2 + 2*x - 5
```

```
print(f(1)) # มาจาก 1**2 + 2*1 - 5 = -2
```

map

เป็นคำสั่งสำหรับใช้ดำเนินการฟังก์ชันกับ list ของ iterable (ชนิดข้อมูลของ iterable ต้องเป็น List) มีรูปแบบไวยากรณ์ดังนี้

map(function, iterable)ตัวอย่างเช่น

```
a = map(lambda a: a+1, [1,2,3,4])
```

```
print(list(a))
```

ผลลัพธ์

```
[2, 3, 4, 5]
```

len(ตัวแปรหรือข้อมูล)

ตัวอย่างเช่น

```
a = "123456"
len(a)
```

ผลลัพธ์ 6

ฟังก์ชัน `chr()` เพื่อหาอักขระ ตัวอย่างเช่น

```
>>> chr(65)    #แปลงตัวเลขเป็นอักขระ
'A'
```

ฟังก์ชัน `print()` แสดงข้อความตัวอย่างเช่น

```
>>>print('Python programming')    #พิมพ์แสดงข้อความบนจอภาพ
Python programming
```

คำสั่ง **max()**

เป็นคำสั่งสำหรับใช้หาค่าที่มากที่สุดในชนิดข้อมูลนั้น ๆ มีรูปแบบดังนี้

```
max(ตัวแปรหรือข้อมูล)
```

ตัวอย่างเช่น

```
max(1,2,3)
```

ผลลัพธ์ 3

```
max([1,2,3,-1])
```

ผลลัพธ์ 3

```
max('abcdb') # ตัวอักษร d มีความสุดถ้าเรียงตามลำดับตัวอักษร
```

ผลลัพธ์ d

ฟังก์ชัน `min()` เป็นคำสั่งสำหรับใช้หาค่าที่น้อยที่สุดในชนิดข้อมูลนั้น ๆ มีรูปแบบไวยากรณ์ดังนี้
`min(ตัวแปรชนิดข้อมูล)`

```
min('abca')
```

ผลลัพธ์ 'a'

```
min([1,2,3,-1])
```

ผลลัพธ์ -1

```
min(1,2,3)
```

ผลลัพธ์ 1

ตัวอย่างเช่น

ฟังก์ชัน `zip()` เป็นคำสั่งสำหรับใช้รวมทูเพิล (tuples) 2 ตัว จับคู่เป็นทูเพิลเดียวกัน

ตัวอย่างเช่น

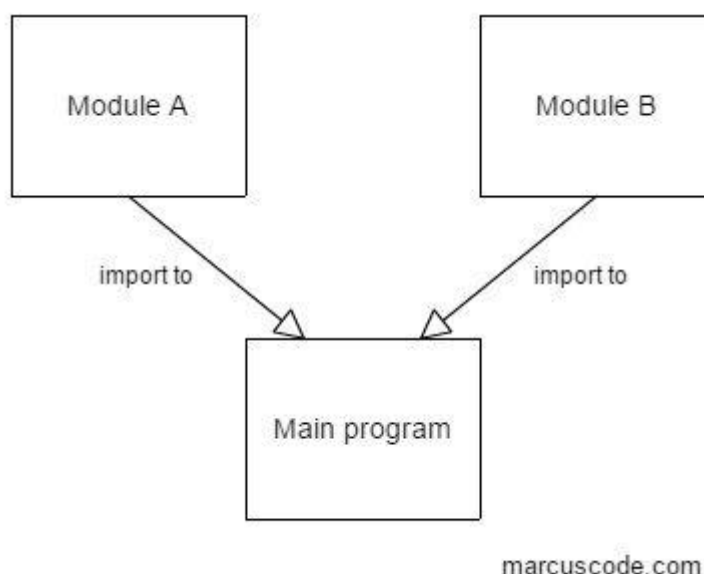
```
a = ["a", "b", "c"]  
b = [1, 2, 3]  
c = zip(a,b)  
print(list(c))
```

ผลลัพธ์ [('a', 1), ('b', 2), ('c', 3)]

2. Python – modules

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

โมดูล (Module) คือ ไฟล์หรือส่วนของโปรแกรมที่ใช้สำหรับกำหนดตัวแปร ฟังก์ชัน หรือคลาสโดยแบ่งย่อยอีกหน่วยหนึ่งจากโปรแกรมหลัก และในโมดูลยังสามารถประกอบไปด้วยคำสั่งประมวลผลการทำงานได้ ยกตัวอย่างเช่น เมื่อคุณเขียนโปรแกรมในภาษา Python คุณอาจจะมีฟังก์ชันสำหรับทำงานและจัดการกับตัวเลขเป็นจำนวนมาก และในขณะเดียวกัน คุณไม่ต้องการให้โปรแกรมหลักนั้นมีขนาดใหญ่เกินไป นั่นหมายความว่า คุณสามารถนำฟังก์ชันเหล่านั้นมาสร้างเป็นโมดูล 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

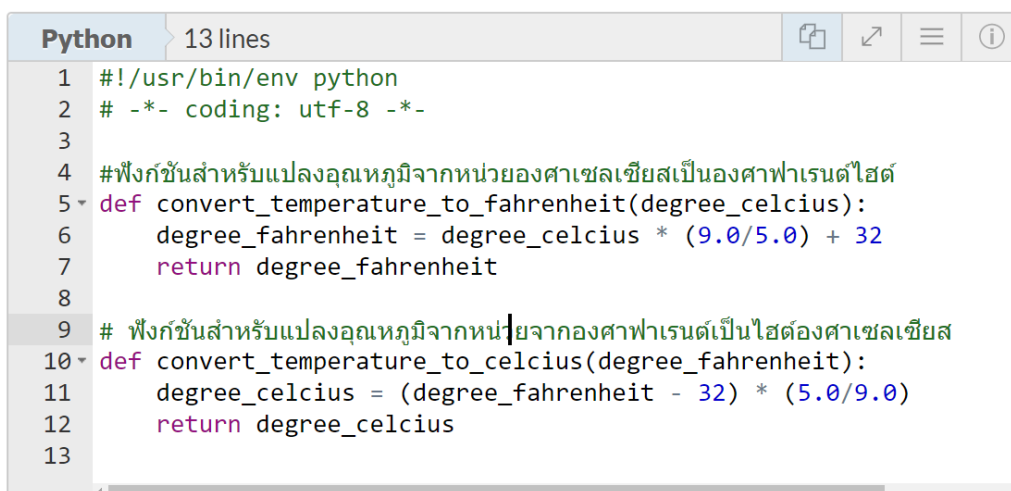
ในการออกแบบโปรแกรมที่มีขนาดใหญ่ขึ้น การจัดระบบของตัวแปร, คำสั่ง, ฟังก์ชันให้อยู่ในรูปแบบที่สืบค้นได้สะดวกนั้นก็มีส่วนสำคัญ การแยกส่วนต่างๆ ของการทำงานที่ซับซ้อนออกเป็นส่วนย่อยๆ แล้วจัดกลุ่มการทำงานเหล่านั้นให้เหมาะสม จะทำให้ลดความซ้ำซ้อนในการเขียนโปรแกรมและใช้เวลาน้อยในการสืบค้นชุดคำสั่งที่ทำงานแบบเดียวกัน

โมดูล(modules) ใช้อ้างอิงไฟล์ที่บรรจุชุดคำสั่ง ซึ่งชุดคำสั่งเหล่านั้นจะถูกรวบรวมไว้ให้อยู่ในที่เดียวกันเพื่อทำงานและเพื่อประมวลผลร่วมกัน ตัวอย่างเช่น ทำการสร้างไฟล์ที่ชื่อ “degree_converter.py” เพื่อให้ไฟล์ดังกล่าวทำการรวบรวมคำสั่งและฟังก์ชันสำหรับการแปลงข้อมูลอุณหภูมิในหน่วยต่างๆ โดยไฟล์ดังกล่าวนี้สามารถจัดเป็นหนึ่งโมดูลได้ โดยชื่อของโมดูลก็คือ “degree_converter” ชื่อเดียวกับชื่อไฟล์

การที่เราทำการแบ่งโปรแกรมใหญ่ออกเป็นโมดูลย่อยๆ ทำให้เราสามารถจัดกลุ่มฟังก์ชันเหล่านั้น ให้การทำงานประเภทเดียวกันอยู่ในกลุ่มเดียวกัน การจัดระเบียบดังกล่าวนี้ นอกจากจะทำให้ผู้พัฒนาเอง สามารถค้นหาคำสั่งหรือฟังก์ชันที่ต้องการเรียกใช้ได้ง่ายและเร็วขึ้นแล้ว ยังทำให้สามารถสามารถนำเข้าโมดูลและจะเรียกใช้ฟังก์ชันเหล่านั้นซ้ำได้เรื่อยๆ โดยไม่ต้องเขียนฟังก์ชันหรือคำสั่งแบบเดิมซ้ำใหม่ทุกครั้ง โดยเราสามารถจะประกาศชุดฟังก์ชันที่ซับซ้อนๆ ลงในโมดูล และเมื่อต้องการใช้โมดูลเหล่านั้นก็สามารถที่จะใช้คีย์เวิร์ด import เพื่อนำเข้าโมดูลนั้นและเรียกใช้ตัวแปรหรือฟังก์ชันที่มีโดยไม่ต้องทำการเขียนชุดฟังก์ชันดังกล่าวซ้ำลงไปใหม่ในโปรแกรมใหม่ที่เรากำลังสร้างขึ้นมา

ตัวอย่างการเรียกใช้งานโมดูลทั่วไป

1.สร้างโมดูลสำหรับแปลงอุณหภูมิ โดยสร้างไฟล์ “degree_converter.py”



```

Python 13 lines
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3
4  #ฟังก์ชันสำหรับแปลงอุณหภูมิจากหน่วยองศาเซลเซียสเป็นองศาฟาเรนไฮต์
5  def convert_temperature_to_fahrenheit(degree_celcius):
6      degree_fahrenheit = degree_celcius * (9.0/5.0) + 32
7      return degree_fahrenheit
8
9  # ฟังก์ชันสำหรับแปลงอุณหภูมิจากหน่วยองศาฟาเรนไฮต์เป็นองศาเซลเซียส
10 def convert_temperature_to_celcius(degree_fahrenheit):
11     degree_celcius = (degree_fahrenheit - 32) * (5.0/9.0)
12     return degree_celcius
13

```

จากตัวอย่างข้างต้น เราได้ทำการสร้างฟังก์ชันชื่อ

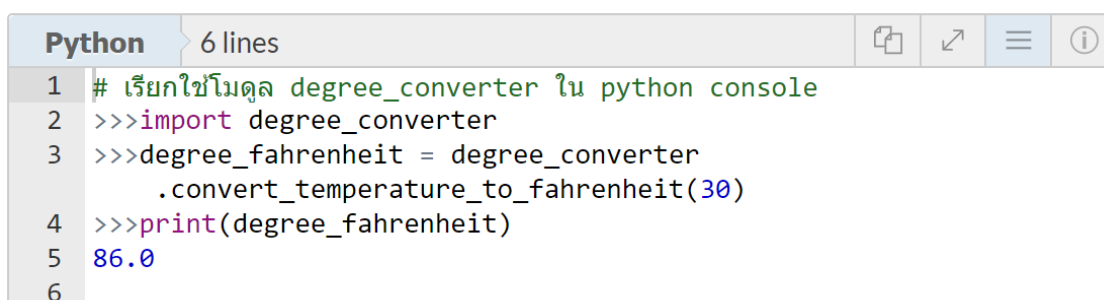
- convert_temperature_to_fahrenheit
- convert_fahrenheit_to_celcius

โดยฟังก์ชันดังกล่าวจะอยู่ในโมดูลชื่อ “degree_converter” ซึ่งเป็นโมดูลที่ทำการบรรจุคำสั่งสำหรับแปลงอุณหภูมิเป็นหน่วยต่างๆ

2. การเรียกใช้โมดูลใน python console

เราสามารถนำเข้าตัวแปรและฟังก์ชันของโมดูลใน python console ด้วยคีย์เวิร์ด “import” ในกรณีไฟล์ของโมดูลดังกล่าวอยู่ในโฟลเดอร์เดียวกันกับโฟลเดอร์ที่ทำงานอยู่ใน python console

ตัวอย่างการใช้งาน



```

Python 6 lines
1 # เรียกใช้โมดูล degree_converter ใน python console
2 >>>import degree_converter
3 >>>degree_fahrenheit = degree_converter
  .convert_temperature_to_fahrenheit(30)
4 >>>print(degree_fahrenheit)
5 86.0
6
  
```

จากตัวอย่างข้างต้น การนำเข้าโมดูลโดยใช้คำสั่ง import ในรูปแบบดังกล่าวไม่สามารถอ้างถึงชื่อฟังก์ชันที่ระบุในโมดูล “degree_converter” ได้เลยโดยตรง แต่จะทำการเรียกใช้ฟังก์ชันนั้นได้ต้องมีการทับทากับด้วยชื่อโมดูล “degree_converter” ด้วย การเรียกใช้งานฟังก์ชันในโมดูลจึงต้องทำการอ้างถึงในรูปแบบ “ชื่อโมดูล.ชื่อฟังก์ชัน”

การเรียกใช้โมดูลจากโมดูลอื่น

เราสามารถนำเข้าตัวแปรและฟังก์ชันของโมดูลจากโมดูลอื่น ด้วยคีย์เวิร์ด “import”

ตัวอย่างการใช้งาน

```

Python 17 lines
1 #!/usr/bin/env python
2 # -*- coding: utf-8 -*-
3
4 # ทำการเรียกใช้โมดูล degree_converter จากในโมดูลอื่น
5 import degree_converter
6
7 # กำหนดตัวแปรเก็บข้อมูลอุณหภูมิในหน่วยองศาเซลเซียส
8 degree_celcius = 30
9
10 # ทำการเรียกใช้ฟังก์ชันแปลงอุณหภูมิจากหน่วยเซลเซียสเป็นฟาเรนไฮต์ จากโมดูล
    degree_converter ซึ่งเป็นชื่อเต็มของโมดูลนั้น
11 degree_fahrenheit = degree_converter.convert_temperature_to_fahrenheit(
    degree_celcius )
12
13 print("Temperature %d degree celcius is equal to : %d" % (degree_celcius,
    degree_fahrenheit) )
14
15 # โปรแกรมทำการปรีนต์ค่า
16 # Temperature 30 degree celcius is equal to : 86
17

```

การเรียกใช้ในโมดูลอื่นนั้น ก็ใช้วิธีนำเข้าเช่นเดียวกับที่ใช้ใน python console โดยจากตัวอย่างการนำเข้าโมดูลรูปแบบนี้ไม่สามารถอ้างถึงชื่อฟังก์ชันที่ระบุในโมดูล “degree_converter” ได้เลยโดยตรง แต่จะทำการเรียกใช้ฟังก์ชันได้พร้อมกับกำกับด้วยชื่อโมดูลด้วยการเรียกใช้งานฟังก์ชันในโมดูลจึงต้องทำการอ้างถึงในรูปแบบ “ชื่อโมดูล.ชื่อฟังก์ชัน”

นอกจากโมดูลที่สร้างขึ้นเองเช่น โมดูล “degree_converter” จากตัวอย่างข้างต้นแล้ว ใน python เองนั้นมีโมดูลที่ถูกพัฒนาขึ้นมาไว้แล้วและถูกติดตั้งมาพร้อมกับตอนติดตั้งให้เลือกใช้ค่อนข้างมาก โดยโมดูลส่วนใหญ่จะอยู่ในโฟลเดอร์ Lib ของตำแหน่งที่เราได้ทำการติดตั้งภาษา python เอาไว้ วิธีการเรียกใช้โมดูลเหล่านี้ ก็ใช้วิธีนำเข้าแบบเดียวกันกับที่เราทำการเรียกใช้โมดูลที่เราเขียนขึ้นเองใหม่ทีกล่าวถึงไว้ข้างต้น

การเรียกใช้โมดูลในแบบต่างๆ

1.การเรียกใช้โมดูลโดยใช้เฉพาะคีย์เวิร์ด import

เราสามารถนำเข้าโมดูลโดยใช้เฉพาะคีย์เวิร์ด “import” เพื่อนำเข้าตัวแปรและฟังก์ชันของโมดูลนั้น

ตัวอย่างการใช้งาน

```

Python 17 lines
1 #!/usr/bin/env python
2 # -*- coding: utf-8 -*-
3
4 # ทำการเรียกใช้โมดูล degree_converter
5 import degree_converter
6
7 # กำหนดตัวแปรเก็บข้อมูลอุณหภูมิในหน่วยองศาเซลเซียส
8 degree_celcius = 30
9
10 # ทำการเรียกใช้ฟังก์ชันแปลงอุณหภูมิจากหน่วยเซลเซียสเป็นฟาเรนไฮต์ จากโมดูล
    degree_converter ซึ่งเป็นชื่อเต็มของโมดูลนั้น
11 degree_fahrenheit = degree_converter.convert_temperature_to_fahrenheit(
    degree_celcius )
12
13 print("Temperature %d degree celcius is equal to : %d" % (degree_celcius,
    degree_fahrenheit) )
14
15 # โปรแกรมทำการปรี้นต์ค่า
16 # Temperature 30 degree celcius is equal to : 86
17

```

การนำเข้าโมดูลด้วยการใช้เฉพาะคีย์เวิร์ด `import` ในรูปแบบนี้ไม่สามารถอ้างถึงชื่อฟังก์ชันที่ระบุในโมดูล “`degree_converter`” ได้เลยโดยตรง แต่จะทำการเรียกใช้ฟังก์ชันได้พร้อมกับกำกับด้วยชื่อโมดูลด้วย การเรียกใช้งานฟังก์ชันในโมดูลจึงต้องทำการอ้างถึงในรูปแบบ “ชื่อโมดูล.ชื่อฟังก์ชัน”

2. วิธีการเรียกใช้โมดูลโดยคีย์เวิร์ด `import...as...` และทำการตั้งชื่อย่อใหม่ให้โมดูลนั้น

เราสามารถนำเข้าโมดูลและตั้งชื่อย่อแบบย่อให้ใหม่ เพื่อการเรียกใช้โมดูลในบริบทนั้นจะถูกเรียกโดยใช้ชื่อย่อแทน โดยใช้คำสั่ง `import...as...` เพื่อนำเข้าตัวแปรและฟังก์ชันของโมดูลนั้น

ตัวอย่างการใช้งาน

```

Python 17 lines
1 #!/usr/bin/env python
2 # -*- coding: utf-8 -*-
3
4 # ทำการเรียกใช้โมดูล degree_converter และตั้งชื่อย่อของโมดูลใหม่ว่า dc
    ซึ่งจะช่วยให้สะดวกในการเรียกใช้ภายในโปรแกรม
5 import degree_converter as dc
6
7 # กำหนดตัวแปรเก็บข้อมูลอุณหภูมิในหน่วยองศาเซลเซียส
8 degree_celcius = 30
9
10 # ทำการเรียกใช้ฟังก์ชันแปลงอุณหภูมิจากหน่วยเซลเซียสเป็นฟาเรนไฮต์ จากโมดูล
    degree_converter ซึ่งถูกตั้งชื่อย่อว่า dc
11 degree_fahrenheit = dc.convert_temperature_to_fahrenheit( degree_celcius )
12
13 print("Temperature %d degree celcius is equal to : %d" % (degree_celcius,
    degree_fahrenheit) )
14
15 # โปรแกรมทำการปรี้นต์ค่า
16 # Temperature 30 degree celcius is equal to : 86
17

```

เราได้ทำการเรียกใช้โมดูลและทำการตั้งชื่อย่อแบบย่อให้ใหม่เพื่อทำให้การเขียนโปรแกรมสั้นขึ้น โดยชื่อโมดูล `degree_converter` เราตั้งชื่อย่อให้ว่า `dc` และเราจะเรียกใช้งานโมดูลดังกล่าวด้วย

ชื่อ dc แทน โดยหากเราทำการตั้งชื่อแบบย่อแล้ว ในโปรแกรมของเราก็ต้องเข้าถึงฟังก์ชันของโมดูลแบบใช้ชื่อย่อ หากเราทำการเรียกแบบใช้ชื่อเต็ม โปรแกรมจะเกิดข้อผิดพลาดขึ้น

3.วิธีการเรียกใช้โมดูลโดยคำสั่ง `from ... import...` เพื่อนำเข้าเฉพาะบางฟังก์ชันของโมดูล

เราสามารถทำการเรียกใช้เฉพาะบางฟังก์ชันภายในโมดูล โดยไม่ต้องนำเข้าฟังก์ชันทั้งหมดของโมดูลได้ด้วยคำสั่ง `from ... import...` ซึ่งการนำเข้าแบบนี้จะทำให้เราสามารถจะอ้างอิงถึงชื่อฟังก์ชันของโมดูลได้โดยตรงโดยไม่ต้องระบุชื่อของโมดูล

ตัวอย่างการใช้งาน

```

Python 17 lines
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3
4  # ทำการเรียกใช้โมดูล degree_converter โดยนำเข้าเฉพาะฟังก์ชัน
   convert_degree_to_fahrenheit
5  from degree_converter import convert_temperature_to_fahrenheit
6
7  # กำหนดตัวแปรเก็บข้อมูลอุณหภูมิในหน่วยองศาเซลเซียส
8  degree_celcius = 30
9
10 # ทำการเรียกใช้ฟังก์ชันแปลงอุณหภูมิจากหน่วยเซลเซียสเป็นฟาเรนไฮต์
   ซึ่งการนำเข้าแบบนี้ สามารถเรียกใช้ฟังก์ชันของโมดูลได้เลย
   โดยไม่ต้องระบุชื่อโมดูล
11 degree_fahrenheit = convert_temperature_to_fahrenheit( degree_celcius )
12
13 print("Temperature %d degree celcius is equal to : %d" % (degree_celcius,
   degree_fahrenheit) )
14
15 # โปรแกรมทำการปรี้นต์ค่า
16 # Temperature 30 degree celcius is equal to : 86
17

```

จากตัวอย่างข้างต้น เมื่อเราใช้คำสั่ง `from...import...` เพื่อนำเข้าเฉพาะบางฟังก์ชันของโมดูลใดๆแล้วในที่นี้ก็คือเฉพาะฟังก์ชัน “`convert_temperature_to_fahrenheit`” เมื่อเวลาอ้างอิงถึงฟังก์ชันในโมดูลนั้น เราไม่จำเป็นต้องระบุชื่อโมดูลอีก แต่จะเรียกใช้ตามชื่อฟังก์ชันนั้นได้โดยตรง แต่เราจะสามารถเข้าถึงเฉพาะฟังก์ชันที่เราทำการ `import` มาเท่านั้น ไม่สามารถเรียกใช้ฟังก์ชันอื่นในโมดูลได้

4.วิธีการเรียกใช้โมดูลโดยคำสั่ง `from ... import *` เพื่อนำเข้าตัวแปรและฟังก์ชันทั้งหมดของโมดูล

การเรียกใช้โมดูลแบบนี้จะสามารถเข้าถึงตัวแปรและฟังก์ชันทั้งหมดที่ไม่นำหน้าด้วย “`_`” และยังสามารถเรียกใช้ตัวแปรและฟังก์ชันดังกล่าวโดยไม่ต้องอ้างอิงชื่อโมดูลอีกด้วย

ตัวอย่างการใช้งาน

```

Python 17 lines
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3
4  # ทำการเรียกใช้โมดูล degree_converter
   โดยนำเข้าตัวแปรและฟังก์ชันทั้งหมดจากโมดูล
5  from degree_converter import *
6
7  # กำหนดตัวแปรเก็บข้อมูลอุณหภูมิในหน่วยองศาเซลเซียส
8  degree_celcius = 30
9
10 # ทำการเรียกใช้ฟังก์ชันแปลงอุณหภูมิจากหน่วยเซลเซียสเป็นฟาเรนไฮต์
   ซึ่งการนำเข้าแบบนี้ สามารถเรียกใช้ฟังก์ชันของโมดูลได้เลย
   โดยไม่ต้องระบุชื่อโมดูล
11 degree_fahrenheit = convert_temperature_to_fahrenheit( degree_celcius )
12
13 print("Temperature %d degree celcius is equal to : %d" % (degree_celcius,
   degree_fahrenheit) )
14
15 # โปรแกรมทำการปรี้นค่า
16 # Temperature 30 degree celcius is equal to : 86
17

```

จากตัวอย่าง โปรแกรมจะนำเข้าตัวแปรและฟังก์ชันทั้งหมดของโมดูล และสามารถเรียกใช้ตัวแปรและฟังก์ชันโดยไม่ต้องอ้างถึงชื่อโมดูลด้วย

ตำแหน่งของไฟล์ที่เก็บโมดูลของ python

เมื่อเรียกใช้คำสั่ง import นั้น ตัวประมวลผลจะทำการค้นหาโมดูลทั้งหมดที่ถูกสร้างเอาไว้แล้วจากที่ตั้งของไฟล์จากหลายตำแหน่ง โดยตัวแปลภาษาจะเริ่มทำการค้นหาโมดูลจากโมดูลที่ติดตั้งมาพร้อม python ก่อนแล้วจึงตรวจสอบตำแหน่งที่ระบุจากค่าที่อยู่ในตัวแปร “sys.path” โดยจะค้นข้อมูลไปตามลำดับดังนี้

1. จากตำแหน่งปัจจุบัน
2. จากตัวแปร “PYTHONPATH”
3. ตำแหน่งที่ติดตั้ง python

```

Python 11 lines
1  import sys
2  print(sys.path)
3  # โปรแกรมทำการปรี้นค่า
4  # [' ',
5  # 'C:\\Python33\\Lib\\idlelib',
6  # 'C:\\Windows\\system32\\python33.zip',
7  # 'C:\\Python33\\DLLs',
8  # 'C:\\Python33\\lib',
9  # 'C:\\Python33',
10 # 'C:\\Python33\\lib\\site-packages']
11

```

เราสามารถทำการเพิ่มหรือเปลี่ยนแปลงลิสต์ของตำแหน่งที่บรรจุค่าเหล่านี้ได้เช่นกัน

การเรียกคำสั่งนำเข้าโมดูลเดิมหลายครั้ง

ตัวแปลภาษานั้น จะทำการนำเข้าโมดูลหนึ่งๆ เพียงแค่ครั้งเดียวใน session เดียวกัน หากเราเขียนคำสั่งนำเข้าโมดูลตัวเดียวกันหลายครั้ง ตัวแปลภาษาก็จะทำการนำเข้าเพียงโมดูลนั้นเพียงครั้งเดียวเท่านั้น

- สร้างโมดูลชื่อ "MyTest.py"

```

Python 6 lines
1 #!/usr/bin/env python
2 # -*- coding: utf-8 -*-
3
4 # บรรจุนำเข้าของโมดูล "MyTest.py"
5 print("my test exam")
6

```

- ทำการเรียกใช้โมดูล "MyTest.py" ใน python console

```

Python 6 lines
1 # ทำการเรียกใช้โมดูล "MyTest.py" ใน python console
2 >>>import MyTest
3 my test exam
4 >>>import MyTest
5 >>>import MyTest
6

```

จากตัวอย่างข้างต้น เราทำการเรียกคำสั่งเพื่อนำเข้าโมดูล "MyTest.py" 3 ครั้ง แต่จะเห็นว่า มีการนำเข้าโมดูลดังกล่าวจริงเพียงแค่ครั้งแรกครั้งเดียว ด้วยการพิมพ์ข้อมูลที่บรรจุอยู่ในโมดูลซึ่งเป็นคำสั่งของโมดูลนั้นออกมา และการนำเข้าครั้งอื่นๆ หลังจากนั้นก็จะไม่ถูกดำเนินการ

การเรียกคำสั่งเพื่อนำเข้าโมดูลเดิมใหม่(reload)

ในกรณีเรามีการทำการแก้ไขโมดูลต้นทางซึ่งเรานำเข้ามาใช้งานนั้น ตัวโปรแกรมจะไม่ทำการอัปเดตข้อมูลโมดูลต้นทางที่มีการแก้ไขให้อัตโนมัติ แต่เราสามารถสั่งให้โหลดโมดูลใหม่อีกครั้งได้ด้วยฟังก์ชัน "reload" จากโมดูล "imp"

ตัวอย่างการใช้งานใน python console

```

Python 21 lines
1 # ทำการเรียกใช้โมดูลใน python console
2
3 # นำเข้าโมดูล "imp"
4 >>>import imp
5
6 # นำเข้าโมดูล "MyTest"
7 >>>import MyTest
8 my test exam
9
10 # ทำการแก้ไขคำสั่งในโมดูล "MyTest" ให้ทำการปรี้นคำว่า "Hello world!"
11 >>>import MyTest
12
13 # ลองเรียกคำสั่งนำเข้าโมดูล "MyTest" ใหม่
14   โดยโปรแกรมยังคงเรียกใช้โมดูลเดิมและยังไม่รับรู้ถึงการเปลี่ยนแปลง
15 >>>import MyTest
16
17 # ทำการเรียกคำสั่ง reload เพื่อนำเข้าโมดูลใหม่อีกครั้ง
18   และโปรแกรมจะสามารถเรียกใช้คำสั่งใหม่ที่ถูกรับการแก้ไข
19 >>>imp.reload(MyTest)
20 Hello world!
21 <module 'MyTest' from './MyTest.py'>

```

การใช้ฟังก์ชัน dir() เพื่อดูรายละเอียดของโมดูล

เราสามารถดูฟังก์ชัน dir() เพื่อจะดูรายละเอียดของชื่อตัวแปรและฟังก์ชันที่ถูกประกาศเอาไว้ให้เรียกใช้ภายในโมดูล เช่นในโมดูลชื่อ “degree_converter” เราได้ทำการสร้างฟังก์ชันเพิ่มคือ “convert_celcius_to_fahrenheit” และฟังก์ชัน “convert_fahrenheit_to_celcius” เอาไว้

ตัวอย่างการใช้งาน

```

Python 14 lines
1 # ทำการเรียกใช้คำสั่ง dir เพื่อดูตัวแปรและฟังก์ชันในโมดูล "degree_converter" ใน
  python console
2 >>>dir(degree_converter)
3 # โปรแกรมทำการปรีด์ค่า
4 # ['__builtins__',
5 # '__cached__',
6 # '__doc__',
7 # '__file__',
8 # '__initializing__',
9 # '__loader__',
10 # '__name__',
11 # '__package__',
12 # 'convert_celcius_to_fahrenheit',
13 # 'convert_fahrenheit_to_celcius']
14

```

จากตัวอย่าง เมื่อเรียกใช้ฟังก์ชัน dir เพื่อดูรายชื่อตัวแปรและคำสั่งภายในโมดูล degree_converter แล้ว โปรแกรมจะทำการลิสต์ชื่อและฟังก์ชันที่สามารถเรียกใช้ได้ โดยชื่อที่เริ่มต้นด้วย “_” นั้นจะเป็น attribute ของโมดูลที่สร้างขึ้นมาให้เองอัตโนมัติตอนสร้างโมดูลใหม่ ตัวอย่างเช่น attribute “__name__” จะเป็น attribute ของโมดูลที่ใช้ระบุข้อมูลของชื่อโมดูล

ตัวอย่างการใช้งาน

```

Python 4 lines
1 >>> import degree_converter as dc
2 >>> dc.__name__
3 'degree_converter'
4

```

จากตัวอย่าง เป็นการเรียกใช้โมดูล degree_converter แล้วตั้งชื่อย่อให้โมดูลว่า dc ใน python console ซึ่งจากตัวอย่างก่อนหน้านี้ เราทราบแล้วว่าทุกโมดูลที่ถูกสร้างขึ้นของ python จะมี attribute ชื่อว่า “__name__” ซึ่งจะระบุชื่อเต็มของโมดูลนั้น แล้วจึงทำการเรียกดูข้อมูลของชื่อโมดูลดังกล่าว

3. Python – packages

Package คือการจัดเก็บข้อมูลลงคอมพิวเตอร์ก็เหมือนการจัดเก็บข้อมูลในรูปแบบอื่น เมื่อมีข้อมูลเยอะขึ้น มีทรัพยากรเยอะขึ้น เราก็จำเป็นต้องมีการจัดหมวดหมู่ของข้อมูลให้เป็นระเบียบ เพื่อที่จะช่วยให้เราสามารถทำงานได้สะดวกและรวดเร็วขึ้น การจัดหมวดหมู่และแยกย่อยไฟล์ที่เรา มีอยู่ ทำให้เราสามารถสืบค้นไฟล์เหล่านั้นได้รวดเร็วขึ้นและไม่จำเป็นต้องเสียเวลาในการสร้างข้อมูล ใหม่หากมีข้อมูลดังกล่าวจัดเก็บอยู่แล้ว

เมื่อเราจะต้องทำการจัดระเบียบข้อมูล เราก็สามารถเริ่มต้นจากการจัดรูปแบบโดยการ จัดกลุ่มของไฟล์แยกย่อยเป็นโฟลเดอร์และภายในโฟลเดอร์ดังกล่าวก็อาจจะมีไฟล์หรือโฟลเดอร์อื่นๆ ซ้อนกันไปเรื่อยๆ ตามความเกี่ยวเนื่องกันของข้อมูล

สำหรับไฟล์ที่ทำงานในลักษณะใกล้เคียงกันก็ถูกจัดวางไว้ในโฟลเดอร์เดียวกัน ตัวอย่างเช่น เราเก็บไฟล์ที่บันทึกรูปภาพไว้ในโฟลเดอร์ชื่อว่า “pictures” และภายในโฟลเดอร์ดังกล่าวเราอาจจะมี โฟลเดอร์ย่อย ที่ระบุชื่อปีที่ทำการบันทึกรูปภาพ

ในการสร้าง Package คุณต้องสร้างโฟลด์เดอร์ให้มีโครงสร้างตามที่ต้องการ เนื่องจากใน ภาษา Python นั้น Package ก็คือโฟลด์เดอร์ที่ใช้เก็บไฟล์โมดูลของโปรแกรม และสามารถซ้อนกันได้ แบบลำดับชั้น นี่เป็นตัวอย่างของ Package ที่เราได้สร้างขึ้น โดยมี image เป็นรูทของ Package ภายใน Package นี้จะแบ่งย่อยออกเป็นอีกสาม Package คือ formats filters และ edit และแต่ละ Package จะมี โมดูลอยู่ภายใน

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
    ...

```

ในไฟล์เดอร์ของใน Package จะมีไฟล์พิเศษที่ชื่อว่า `__init__.py` ซึ่งเป็นตัวกำหนดโมดูลภายใน Package สำหรับเพื่อให้ Python ใช้ในการค้นหา Package ภายในไฟล์เดอร์ดังกล่าวเมื่อมีการ 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 Packageย่อยที่จะอนุญาตให้ 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 เป็นสองวิธีที่คุณสามารถทำได้สำหรับการ Import โมดูล jpeg เข้ามาใช้งานในโปรแกรม ในแบบแรกเป็นการ Import เพียงเฉพาะโมดูล jpeg ที่กำหนด และในแบบที่สองนั้นเป็นการเลือกทั้งหมด ซึ่งนี้จะทำให้ Python ทำการ Import โมดูลที่ถูกกำหนดไว้ในไฟล์ `__init__.py` ถ้าหากมีไฟล์ดังกล่าว อย่างไรก็ตาม ในวิธีที่สองนั้นไม่แนะนำในทางปฏิบัติ เพราะคุณควรจะ Import เพียงโมดูลที่ต้องการใช้งานจริงๆ เท่านั้น ซึ่งนี้จะช่วยประหยัดหน่วยความจำได้

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

นี่เป็นตัวอย่างของการ Import โมดูลจาก Package และสร้างออปเจ็กต์จากคลาส JPEG คุณจะเห็นว่าในการเข้าถึงคลาสนั้นเรายังคงต้อง Prefix กับชื่อของโมดูลเสมอ

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

1.การเรียกใช้โมดูลใน package

เราสามารถเรียกใช้โมดูลจาก package โดยค้นข้อมูลของชื่อ package และชื่อโมดูลโดยใช้เครื่องหมาย “.” แต่ก็อาจจะมีรูปแบบการเรียกต่างกันไปขึ้นอยู่กับวิธีที่เรานำเข้า package ด้วย

ตัวอย่างการใช้งาน

สร้างไฟล์ “`degree_converter.py`” แล้วทำการจัดเก็บให้อยู่ในโครงสร้างของ package ตามตัวอย่างข้างต้น `room > utils > degree_converter.py`

```

Python 13 lines
1 #!/usr/bin/env python
2 # -*- coding: utf-8 -*-
3
4 # ฟังก์ชันสำหรับแปลงอุณหภูมิจากหน่วยองศาเซลเซียสเป็นองศาฟาเรนไฮต์
5 def convert_temperature_to_fahrenheit(degree_celcius):
6     degree_fahrenheit = degree_celcius * (9.0/5.0) + 32
7     return degree_fahrenheit
8
9 # ฟังก์ชันสำหรับแปลงอุณหภูมิจากหน่วยจากองศาฟาเรนไฮต์เป็นองศาเซลเซียส
10 def convert_temperature_to_celcius(degree_fahrenheit):
11     degree_celcius = (degree_fahrenheit - 32) * (5.0/9.0)
12     return degree_celcius
13

```

ในตัวอย่างเป็นการสร้างโมดูลชื่อ “degree_converter” โดยโมดูลนี้เป็นส่วนหนึ่งของ package ชื่อ room และอยู่ใน sub-package ชื่อ util

2. การเรียกใช้ฟังก์ชันของโมดูลใน package แบบเต็ม

```

Python 3 lines
1 import room.utils.degree_converter
2 room.utils.degree_converter.convert_temperature_to_fahrenheit(30)
3

```

ในตัวอย่างเป็นการเรียกใช้งานฟังก์ชัน convert_temperature_to_fahrenheit ซึ่งเป็นส่วนหนึ่งของ package room โดยหากเราทำการนำเข้าโมดูลโดยใช้คีย์เวิร์ด import เวลาเราเรียกใช้ เราก็ต้องอ้างถึงชื่อ package, sub-package ย่อยไปจนถึงตำแหน่งของโมดูล แล้วจึงเรียกใช้งานฟังก์ชัน

3. การเรียกใช้โมดูลโดยไม่ต้องมี prefix

```

Python 3 lines
1 from room.utils import degree_converter
2 degree_converter.convert_temperature_to_fahrenheit(30)
3

```

ในตัวอย่างเป็นการเรียกใช้งานฟังก์ชัน convert_temperature_to_fahrenheit ซึ่งเป็นส่วนหนึ่งของ package room โดยหากเราทำการนำเข้าโมดูลโดยใช้คีย์เวิร์ด from...import... ในรูปแบบนี้ เวลาเราเรียกใช้งานฟังก์ชัน เราก็เพียงแค่อ้างถึงชื่อของโมดูลและฟังก์ชันที่ต้องการใช้งาน

4. การเรียกใช้ฟังก์ชันโดยไม่ต้องมี prefix

```

Python 3 lines
1 from room.utils.degree_converter import convert_temperature_to_fahrenheit
2 convert_temperature_to_fahrenheit(30)
3

```

ในตัวอย่างเป็นการเรียกใช้งานฟังก์ชัน convert_temperature_to_fahrenheit ซึ่งเป็นส่วนหนึ่งของ package room โดยหากเราทำการนำเข้าฟังก์ชันโดยใช้คีย์เวิร์ด from...import... ในรูปแบบนี้ เวลาเราเรียกใช้งานฟังก์ชัน เราก็เพียงแค่อ้างถึงชื่อของฟังก์ชันที่ต้องการใช้งาน แต่การเรียกใช้ชื่อฟังก์ชันอย่างเดียวนี้อาจทำให้เกิดปัญหาได้ ในกรณีที่ชื่อฟังก์ชันนั้นมีชื่อเหมือนกันในหลายๆ โมดูลที่ถูกนำมาใช้ในโปรแกรม

บรรณานุกรม

1. <http://marcuscode.com/lang/python/functions>การสร้างฟังก์ชันในภาษา Python
 2. <http://marcuscode.com/lang/python>ภาษาPython
 3. <http://marcuscode.com/lang/python/modules>
 4. <https://sites.google.com/site/karkheinyormkaerm/fangkchan-function>
 5. <https://sites.google.com/site/dotpython/>
 6. <https://sites.google.com/site/pythonclassroom/module-package-python/module-python-is-module-python>
- โมดูลไพธอน และแพ็คเกจไพธอน
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>
 10. <https://searchsoftwarequality.techtarget.com/definition/functional-specification>
 11. <https://python3.wannaphong.com/2015/06/python-1-len-max-min-zip-lambda-map.html>
 12. <https://www.mindphp.com/%E0%B8%9A/83-python/2508-dir-function-python.html>
 13. <http://marcuscode.com/lang/python/type-conversions>