การแนะนำและสภาพแวดล้อมของไพธอน

ไพธอน(Python) เป็นภาษาโปรแกรมระดับสูง(High level) อยู่ในลักษณะภาษาอินเทอร์พรีเตอร์โปรแกรมมิ่ง(Interpreter Programming) คือการประมวลผลทีละบรรทัด ไพธอนถูกสร้างในช่วงศตวรรษที่ 80 โดย Guido van Rossum ที่National Research Institute for Mathematics and Computer Science ประเทศเนเธอร์แลนด์ พัฒนาและดัดแปลงขึ้นมาจากหลายภาษาได้แก่ ABC, Modula-3, C, C++, Algor-68, Smalltalk Unix Shell และ Script language อื่นๆ

สภาพแวดล้อมของไพธอน

ไพธอนเป็น open source ที่ไม่ยึดติดกับอุปกรณ์ฮาร์ดแวร์แบบใดแบบหนึ่ง จึงสามารถใช้ได้กับทุก platform ไม่ว่าจะเป็น Windows, Linux, Mac OS X เป็นต้น

Python Environment Variable

ตัวแปร	คำอธิบาย
PYTHONPATH	ตัวแปรชนิดนี้จะบอกเพื่อให้อินเทอร์พรีเตอร์สามารถทำการดึงไฟล์เข้าสู่โปรแกรมได้ โดย PYTHONPATH จะมี library ในการเก็บรวบรวมคำสั่งต่างๆ ซึ่งจะถูกติดตั้งมาในตัวโปรแกรมแล้ว
PYTHONSTARTUP	ในการกำหนดค่าเริ่มต้น จะเกิดขึ้นขณะเรียกใช้อินเทอร์พรีเตอร์
PYTHONCASEOK	โปรแกรมจะทำกาตรวจสอบ case-sensitive ในการกำหนดค่าให้กับตัวแปรด้วย
PYTHONHOME	ในการเลือก PYTHONPATH หรือ PYTHONSTARTUP ในการเข้าใช้ library

การทำงานของ PYTHON

- Interactive Interpreter โดยโปรแกรมจะถูกสร้างขึ้นเมื่อเรียกใช้อินเทอร์พรีเตอร์บน command line
- Script from the Command line การทำงานนั้น command line จะเรียกใช้อินเทอร์พรีเตอร์
- Integrated Development Environment สามารถ run โปรแกรมบน GUI ซึ่งจะถูกติดตั้งบนโปรแกรมอยู่แล้ว

Python Basic Syntax

ภาษา Python มีความคล้ายคลึงกับหลายภาษา เช่น Perl, C และ Java อย่างไรก็ตาม แต่ละภาษาย่อมมีความแตกต่างระหว่างภาษาอย่างชัดเจน ซึ่งในบทนี้จะเป็นการยกตัวอย่างเพื่อให้เข้าใจไวยกรณ์อย่างง่ายๆของภาษา Python

First Python Program :

หากใช้โปรแกรมนี้บนระบบปฏิบัติการ windows จะพบข้อความลักษณะนี้อยู่บนจอแสดงผล

```
Python 3.4.2 (v3.4.2:ab2c023a9432, Oct  6 2014,
22:15:05) [MSC v.1600 32 bit (Intel)] on win32

Type "help", "copyright", "credits" or "license"
for more information.

>>>
```

พิมพ์ข้อความต่อไปนี้ที่ด้านขวาของหน้าจอ Python และกด Enter :

```
>>> print ("Hello, Python!");
```

จะให้ผลดังนี้

```
Hello, Python!
```

การตั้งชื่อตัวแปร (Python Identifiers)

การตั้งชื่อตัวแปรใน python จะเป็นชื่อที่ใช้ในการตั้งชื่อตัวแปร,ฟังก์ชัน,คลาส,โมดูลหรือวัตถุอื่นๆ โดยการตั้งชื่อจะขึ้นต้นด้วยตัวอักษร A-Z หรือ a-z หรือเครื่องหมายขีดล่าง (_) ตามด้วย 0 หรือตัวอักษรอื่นๆ,เครื่องหมายขีดล่าง (_) และตัวเลข (0-9)

ภาษา python ไม่อนุญาตให้ใช้เครื่องหมายวรรคตอน หรือเครื่องหมายอื่นๆ เช่น @, $ และ % ในการตั้งชื่อตัวแปร ภาษา python เป็นภาษาโปรแกรมที่เป็น case sensitive นั่นคือ ตัวอักษรพิมพ์เล็กและตัวอักษรพิมพ์ใหญ่ จะถือว่าเป็นคนละตัวกันในการตั้งชื่อตัวแปร

ตัวอย่าง

```
>>> Test = "Hello"; test = "Python"
```

```
>>> print (Test)
Hello                    #result
>>> print(test)
Python                   #result
>>> print(Test+test)
HelloPython              #result
```

คำสงวน(Reserve Words) :

ตารางต่อไปนี้เป็นคำสงวนในภาษา python ซึ่งคำสงวนเหล่านี้ไม่สามารถนำไปตั้งชื่อค่าคงที่ หรือตัวแปร หรือการตั้งชื่ออื่นๆ

Keywords in Python programming language

False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	
break	except	in	raise	

Note : ตารางนี้อ้างอิงจาก python เวอร์ชั่น 3.4

เส้นและการเยื้อง(Lines and Indentation) :

จำนวนช่องว่างการเยื้องของ statement ทั้งหมด จะต้องเยื้องในจำนวนที่เท่ากัน ดังตัวอย่างทั้งสองบล็อกต่อไปนี้ :

```
>>> n=20
>>> if n<0:
     print("Answer")
     print("False")
else:
     print("Answer")
     print("True")
```

ในบล็อกที่สองนี้ มีการเยื้องที่ไม่เท่ากัน จึงทำให้เกิดข้อผิดพลาด :

```
>>> n=20
>>> if n<0:
     print("Answer")
     print("False")
else:
     print("Answer")
  print("True")
```

ดังนั้น หากคำสั่งนั้นอยู่ในบล็อกของเงื่อนไขเดียวกัน จะต้องเยื้องให้เท่ากัน ไม่เช่นนั้นจะเกิดข้อผิดพลาด

Multi-Line Statement

Statement ใน python มักจะจบลงด้วยการขึ้นบรรทัดใหม่ แต่ในที่นี้เราสามารถใช้ความต่อเนื่องของคำสั่งได้ โดยใช้ เครื่องหมาย backslash (\) เพื่อแสดงว่ามีคำสั่งที่ต้องการดำเนินการต่อ ตัวอย่างเช่น :

```
total = item_one + \
     item_two + \
     item_three
```

แต่ถ้า statement ที่อยู่ภายใน (),{} หรือ [] ไม่จำเป็นต้องใช้เครื่องหมาย backslash (\) เช่น :

```
days = ['Monday', 'Tuesday',
'Wednesday',
'Thursday', 'Friday']
```

การกล่าวอ้างในภาษา python (Quotation in Python) :

ภาษา python ยอมรับการใช้เครื่องหมาย single quote(‘), double quote(“) และ triple quote(‘’’ or “””) เพื่อแสดงข้อความหรือสตริง ดังที่จะแสดงให้เห็นต่อไปนี้

```
word = 'word'
sentence = "This is a sentence."
paragraph = """This is a paragraph. It is
made up of multiple lines and sentences."""
```

Comment ใน ภาษา python

ในภาษา python จะใช้เครื่องหมาย (#) ในการเริ่มการ comment ทุกตัวอักษรหลังจาก # จะถือว่าเป็น comment และจะไม่แสดงให้เห็นเมื่อรันโปรแกรมตัวอย่าง

```
# First comment
print ("Hello, Python!"); # second comment
```

ได้ผลลัพธ์ดังนี้ :

```
Hello, Python!
```

Comment อาจจะอยู่ในบรรทัดเดียวกัน หลังจากคำสั่งก็ได้ เช่น :

```
name = "Wipa" # This is again comment
```

หรืออาจจะ comment ได้หลายบรรทัด ดังนี้

```
# This is a comment.
# This is a comment, too.
# This is a comment, too.
# I said that already.
```

การรอผู้ใช้ (Waiting for the User) :

บรรทัดต่อไปนี้ของโปรแกรมจะแสดงให้กดปุ่ม Enter เพื่อออกและเป็นการรอผู้ใช้

```
input("\n\nPress the enter key to exit.")
```

Note: ใน Python2 จะใช้ raw_input() แต่เวอร์ชั่น Python3.0 ขึ้นไป จะมีการเปลี่ยนชื่อจาก raw_input() เป็น input()

จะเห็นได้ว่า “\n\n” ถูกนำมาใช้ในการสร้างสองบรรทัดก่อนที่จะทำคำสั่งอื่น และเป็นการรอรับค่าจากผู้ใช้ นี่เป็นเคล็ดลับในการที่จะทำให้หน้าต่างคอนโซลเปิดรอจนกว่าผู้ใช้จะกระทำการใดๆกับโปรแกรม

การสร้างหลายๆคำสั่งในบรรทัดเดียว

เราจะใช้เครื่องหมาย semicolon (;) ในการจบคำสั่งแต่ละคำสั่ง จึงทำให้ในหนึ่งบรรทัดสามารถมีได้หลายคำสั่ง ดังตัวอย่าง :

```
var = 12; day = 'Monday'; print(day); print(var)
```

นอกจากนี้ยังมีเครื่องหมาย colon (:) ที่ใช้ตามหลังคำสั่งจำพวก if, while, def และ คลาส แล้วตามด้วยชุดคำสั่งอื่นๆ เช่น :

```
if expression :
    suite
elif expression :
    suite
else :
    suite
```

บันทึกข้อความ

ชนิดของตัวแปร

ในภาษาไพธอนนั้นไม่จำเป็นต้องสนใจในการประกาศชนิดข้อมูลของตัวแปรเหมือนภาษาอื่นๆ เพราะไพธอนเป็นภาษาแบบอินเตอร์พรีเตอร์ที่ไม่สนใจชนิดของตัวแปร

การกำหนดค่าตัวแปร

ในการกำหนดค่าตัวแปรในไพธอนไม่จำเป็นต้องประกาศล่วงหน้าเพื่อจองพื้นที่ในหน่วยความจำ เนื่องจากเมื่อมีการกำหนดค่าของตัวแปรจะมีการจองพื้นที่ในหน่วยความจำโดยอัตโนมัติ ในการกำหนดค่านั้นจะกำหนดโดยการใช้เครื่องหมายเท่ากับ (=) โดยที่ด้านซ้ายของเครื่องหมายเท่ากับคือชื่อตัวแปรและด้านขวาคือค่าที่ต้องการกำหนดให้ตัวแปร

เช่น

```
number = 100              // กำหนดค่าของ Integer
km     = 23.5             // กำหนดค่าของ Floating Point
name   = “Somchai”        // กำหนดค่าของสตริง
```

การกำหนดค่าตัวแปรหลายค่า

ไพธอนสามารถกำหนดค่าของตัวแปรหลายตัวพร้อมกันได้ โดยสามารถกำหนดได้ทั้งที่เป็นชนิดข้อมูลแบบเดียวกันหรือชนิดข้อมูลต่างกันก็ได้

เช่น a = b = c = 1 หรือ a, b, c = 1, 2.5, ”Somchai”

หากมีการประกาศตัวแปรซ้ำ ค่าที่จะแสดงผลจะเป็นค่าล่าสุดที่มีการประกาศ

ชนิดข้อมูลแบบมาตรฐาน

ไพธอนมีชนิดของข้อมูลแบบมาตรฐานอยู่ 5 ชนิด ได้แก่

1. ตัวเลข (Numbers)
2. สายอักขระ (String)
3. ลิสต์ (Lists)
4. ทับเปิ้ล (Tuples)
5. ดิกชันนารี (Dictionary)

ตัวเลข (Numbers)

ชนิดข้อมูลแบบตัวเลขจะใช้สำหรับเก็บข้อมูลที่เป็นจำนวนจริง

ตัวเลขเชิงวัตถุจะถูกสร้างขึ้นเมื่อมีการกำหนดค่าให้ตัวแปร เช่น

```
var1 = 1
var2 = 10
```

สามารถลบค่าของตัวแปรได้โดยใช้โดยการประกาศ **del** หน้าตัวแปรที่ต้องการลบ เช่น

```
del   (var1[,var2[,var3[…,varN]]])
```

สามารถลบค่าเดียวหรือหลายค่าพร้อมกันได้

```
del    (var)
del    (var_a,var_b)
```

เมื่อลบค่า var1 แล้วทำการแสดงผลจะได้

```
NameError: name 'var1' is not
defined
```

ไพธอนสนับสนุนชนิดข้อมูลแบบตัวเลข 4 ชนิดข้อมูลที่แตกต่างกัน

1. int (integer แบบมีเครื่องหมายกำกับ)
2. long (สามารถเก็บข้อมูลได้มากกว่า int และสามารถแสดงผลในรูปแบบของเลขฐานแปดและฐานสิบหกได้)
3. float (ค่าทศนิยม)
4. complex (จำนวนเชิงซ้อน)

ไพธอนอนุญาตให้สามารถใช้ตัวแอลพิมพ์เล็กกับชนิดข้อมูลแบบ long ได้ แต่โดยทั่วไปจะนิยมใช้ตัวแอลพิมพ์ใหญ่เพื่อหลีกเลี่ยงการสับสนระหว่างตัวแอลพิมพ์เล็กกับเลขหนึ่งอารบิก (1)

int	long	float	complex
10	51924361L	0.0	3.14j
100	-0x19323L	15.20	45.j
-786	0122L	-21.9	9.322e-36j
080	0xDEFABCECBDAECBFBAE1	32.3+e18	.876j
-0490	535633629843L	-90.	-.6545+0J
-0x260	-052318172735L	-32.54e100	3e+26J
0x69	-4721885298529L	70.2-E12	4.53e-7j

สายอักขระ (String)

สายอักขระในไพธอนจะถูกกำหนดอยู่ภายในเครื่องหมายคำพูดสามารถใช้ได้ทั้ง ('...') และ ("...") โดยสามารถสร้างสายอักขระย่อยได้จากการใช้ slice operator ([] และ [:]) เพื่อเข้าถึงมีค่าดัชนีเริ่มต้นที่ 0 ใช้เครื่องหมายบวก(+)ในการนำสายอักขระมาต่อกันก่อนแสดงผลและใช้เครื่องหมายดอกจันทร์(*)ในการทำซ้ำข้อมูลตามจำนวนครั้งที่กำหนดในการแสดงผล ดังตัวอย่าง

```
str = 'Silpakorn'

print (str)                          # Prints complete string
print (str[0])                       # Prints first character of the string
print (str[3:5])                     # Prints characters starting from 4rd to 5th
print (str[2:])                      # Prints string starting from 3rd character
print (str * 2)                      # Prints string two times
print (str + " University")          # Prints concatenated string
```

ถ้าต้องการพิมพ์ให้อยู่บรรทัดเดียวกัน ให้พิมพ์ Semicolon (;) ต่อท้ายเมื่อจบคำสั่งแล้วจึงพิมพ์คำสั่งถัดไป

แสดงผล

```
Silpakorn
S
pa
lpakorn
SilpakornSilpakorn
Silpakorn University
```

ลิสต์ (Lists)

ลิสต์ในไพธอนนั้นเหมือนกับอาเรย์ในภาษาซีแต่แตกต่างกันที่ในลิสต์ของไพธอน 1 ลิสต์สามารถมีได้หลายชนิดข้อมูลภายในลิสต์เดียว การกำหนดค่าจะให้อยู่ใน square braces ([]) และคั่นด้วยลูกน้ำ(,) โดยลิสต์สามารถเลือกแสดงผลข้อมูลภายในลิสต์ได้โดยการใช้ slice operator ([] และ [:]) มีค่าเริ่มต้นที่ดัชนีเท่ากับ 0 ใช้เครื่องหมายบวก(+)ในการนำสายอักขระมาต่อกันก่อนแสดงผลและใช้เครื่องหมายดอกจันทร์(*)ในการทำซ้ำข้อมูลตามจำนวนครั้งที่กำหนดในการแสดงผล ดังตัวอย่าง

```
list = [ 'comsci', 2911 , 9.98, 'Satit', 70.2 ]
tinylist = [1150, 'jubjang']

print (list)              # Prints complete list
print (list[0])           # Prints first element of the list
print (list[1:3])         # Prints elements starting from 2nd till 3rd
print (list[2:])          # Prints elements starting from 3rd element
print (tinylist * 2)      # Prints list two times
print (list + tinylist)   # Prints concatenated lists
```

ถ้าต้องการพิมพ์ให้อยู่บรรทัดเดียวกัน ให้พิมพ์ Semicolon (;) ต่อท้ายเมื่อจบคำสั่งแล้วจึงพิมพ์คำสั่งถัดไป

แสดงผล

```
['comsci', 2911 , 9.98, 'Satit', 70.2]
comsci
[2911, 9.98]
[9.98, 'Satit', 70.2]
[1150, 'jubjang', 1150, 'jubjang']
['comsci', 2911 , 9.98, 'Satit', 70.2, 1150, 'jubjang']
```

ทับเปิ้ล (Tuples)

ทับเปิ้ลคือชนิดข้อมูลที่มีความคล้ายคลึงกับลิสต์ มีแตกต่างกันที่ลิสต์อยู่ภายในเครื่องหมาย square braces [] และสามารถเพิ่มเติมหรือแก้ไขข้อมูลหรือขนาดได้ แต่ทับเปิ้ลมีการประกาศค่าอยู่ในวงเล็บ () และไม่สามารถเปลี่ยนแปลงข้อมูลภายในได้ ดังตัวอย่าง

```
tuple = ('comsci', 2911 , 9.98, 'Satit', 70.2)
tinytuple = (1150, 'jubjang')

print (tuple)                    # Prints complete list
print (tuple[0])                 # Prints first element of the list
print (tuple[1:3])               # Prints elements starting from 2nd till 3rd
print (tuple[2:])                # Prints elements starting from 3rd element
print (tinytuple * 2)            # Prints list two times
print (tuple + tinytuple)        # Prints concatenated lists
```

ถ้าต้องการพิมพ์ให้อยู่บรรทัดเดียวกัน ให้พิมพ์ Semicolon (;) ต่อท้ายเมื่อจบคำสั่งแล้วจึงพิมพ์คำสั่งถัดไป

แสดงผล

```
('comsci', 2911 , 9.98, 'Satit', 70.2)
comsci
(2911, 9.98)
(9.98, 'Satit', 70.2)
(1150, 'jubjang', 1150, 'jubjang')
('comsci', 2911 , 9.98, 'Satit', 70.2, 1150, 'jubjang')
```

ทับเปิ้ลไม่สามารถปรับเปลี่ยนขนาดและข้อมูลได้

```
tuple = ('comsci', 2911 , 9.98, 'Satit', 70.2)
list = ['comsci', 2911 , 9.98, 'Satit', 70.2]
tuple[2] = 1000    # Invalid syntax with tuple
list[2] = 1000     # Valid syntax with list (ค่าตัวที่ 3 ใน list จะเปลี่ยนจาก 9.98 เป็น 1000)
```

เมื่อมีการปรับเปลี่ยนค่าของ Tuple โปรแกรมจะแสดงผลว่าไม่สามารถกระทำได้

```
TypeError: 'tuple' object does not support item
```

ดิกชันนารี (Dictionary)

ดิกชันนารีเป็นข้อมูลอีกชนิดหนึ่งที่มีการทำงานที่เหมือน Accosiative Array มีการจับคู่ระหว่าง keys และ values โดยดิกชันนารีสามารถใช้ได้กับทุกชนิดข้อมูลของไพธอน แต่โดยปกติแล้วจะใช้ในรูปแบบของตัวเลขและสายอักขระ ในบางกรณียังสามารถมองเป็นวัตถุในไพธอนได้อีกด้วย ดิกชันนารีจะมีการประกาศค่าของตัวแปรให้อยู่ภายในปีกกา {} และสามารถใช้ square braces [] ในการกำหนดค่าอ้างอิงได้อีกด้วย ดังตัวอย่าง

```
dict = {}
dict['one'] = "NUMBER ONE"
dict[2]     = "NUMBER TWO"
tinydict = {'name': 'johnifer','code':5732, 'sales': '15 Percent'}


print (dict['one'])               # Prints value for 'one' key
print (dict[2])                   # Prints value for 2 key
print (tinydict)                  # Prints complete dictionary
print (tinydict.keys())           # Prints all the keys
print (tinydict.values())         # Prints all the values
```

ถ้าต้องการพิมพ์ให้อยู่บรรทัดเดียวกัน ให้พิมพ์ Semicolon (;) ต่อท้ายเมื่อจบคำสั่งแล้วจึงพิมพ์คำสั่งถัดไป

แสดงผล (ค่า keys และ values จะแสดงผลตามลำดับอักษรของ key)

```
NUMBER ONE
NUMBER TWO
{'code': 5732, 'name': 'johnifer', 'sales': '15 Percent'}
['code', 'name', 'sales']
[5732, 'johnifer', '15 Percent']
```

การแปลงชนิดข้อมูล

ในการทำงานโดยทั่วไปมักจะมีการดำเนินการแปลงชนิดของข้อมูล ซึ่งในไพธอนนี้จะใช้ชื่อชนิดข้อมูลแทนฟังก์ชัน มีหลายฟังก์ชันที่สามารถดำเนินการแปลงข้อมูลจากชนิดหนึ่งไปเป็นอีกชนิดซึ่งค่าที่ได้รับคืนมานั้นจะเป็นวัตถุใหม่ของค่าที่ได้รับการแปลงมา

ฟังก์ชัน	รายละเอียด
int(x [,base])	แปลงค่า x เป็น integer เมื่อ x เป็นสายอักขระ
long(x [,base])	แปลงค่า x เป็น long integer เมื่อ x เป็นสายอักขระ
float(x)	แปลงค่า x เป็นการเก็บข้อมูลแบบ float
complex(real [,imag])	สร้างจำนวนเชิงซ้อน
str(x)	แปลงวัตถุ x เป็น a string representation
repr(x)	แปลงวัตถุ x เป็น an expression string.
eval(str)	ตรวจสอบสายอักขระและคืนค่าเป็นวัตถุ
tuple(s)	แปลงค่า s เป็นทับเปิ้ล
list(s)	แปลงค่า s เป็นลิสต์
set(s)	แปลงค่า s เป็นเซต
dict(d)	สร้างดิกชันนารี. d ต้องเป็นลำดับของทับเปิ้ล (ค่าอ้างอิง,ข้อมูลที่ถูกอ้างอิง)
frozenset(s)	แปลงค่า s เป็น a frozen set.
chr(x)	แปลงค่า integer เป็น character.
unichr(x)	แปลงค่า integer เป็น Unicode character.
ord(x)	แปลงค่า single character เป็น its integer value.
hex(x)	แปลงค่า integer เป็นเลขฐานสิบหก
oct(x)	แปลงค่า integer เป็นเลขฐานแปด