คู่มือเริ่มต้นการเรียนรู้

ภาษา Python เบื้องต้น

โดย

สมาคมกีฬาเครื่องบินจำลองและวิทยุบังคับ

V2.5

โครงสร้างของภาษา Python

Comment

คอมเมนต์ในภาษา Python นั้นเริ่มต้นด้วยเครื่องหมาย # คอมเมนต์สามารถเริ่มต้นที่ตำแหน่งแรกของบรรทัดและหลังจากนั้นจะประกอบไปด้วย Whilespace หรือโค้ดของโปรแกรม หรือคำอธิบาย ซึ่งโดยทั่วไปแล้วคอมเมนต์มักจะใช้สำหรับอธิบายซอสโค้ดที่เราเขียนขึ้นและมันไม่มีผลต่อการทำงานของโปรแกรม นี่เป็นตัวอย่างการคอมเมนต์ในภาษา Python

```
# My first Python program

'''
This is a multiline comment
'''

print ('Hello Python.') # Inline comment
```

ในตัวอย่าง เราได้คอมมเมนต์สามแบบด้วยกัน แบบแรกเป็นการคอมเมนต์แบบ single line แบบที่สองเป็นการคอมเมนต์แบบ multiline line และแบบสุดท้ายเป็นการคอมมเมนต์แบบ inline หรือการคอมเมนต์ภายในบรรทัดเดียวกัน

Statement

Statement คือคำสั่งการทำงานของโปรแกรม แต่ละคำสั่งในภาษา Python นั้นจะแบ่งแยกด้วยการขึ้นบรรทัดใหม่ ซึ่งจะแตกต่างจากภาษา C และ Java ซึ่งใช้เครื่องหมายเซมิโคลอนสำหรับการจบคำสั่งการทำงาน แต่อย่างไรก็ตาม ในภาษา Python นั้นคุณสามารถมีหลายคำสั่งในบรรทัดเดียวกันได้โดยการใช้เครื่องหมายเซมิโคลอน ;

```
name = input('What is your name?\n')
print ('Hi, %s.' % name);
print ('Welcome to Python.'); print ('Do you love it?')
```

ในตัวอย่าง เรามี 4 คำสั่งในโปรแกรม สองบรรทัดแรกเป็นคำสั่งที่ใช้บรรทัดใหม่ในการจบคำสั่ง ซึ่งเป็นแบบปกติในภาษา Python และบรรทัดสุดท้ายเรามีสองคำสั่งในบรรทัดเดียวที่คั่นด้วยเครื่องหมาย ; สำหรับการจบคำสั่ง

Indentation and while space

ในภาษา Python นั้นใช้ Whilespace และ Tab สำหรับกำหนดบล็อคของโปรแกรม เช่น คำสั่ง If Else For หรือการประกาศฟังก์ชัน ซึ่งคำสั่งเหล่านี้นั้นเป็นคำสั่งแบบบล็อค โดยจำนวนช่องว่างที่ใช้นั้นต้องเท่ากัน มาดูตัวอย่างของบล็อคคำสั่งในภาษา Python

```
n = int(input ('Input an integer: '))

if (n > 0):
    print ('x is positive number')
    print ('Show number from 0 to %d' % (n - 1))

else:
    print ('x isn\'t positive number')

for i in range(n):
    print(i)
```

ในตัวอย่าง เป็นบล็อคของโปรแกรมจากท 3 คำสั่ง ในคำสั่งแรกคือ If ในบล็อคนี้มีสองคำสั่งย่อยอยู่ภายใน ที่หัวของบล็อคนั้นจะต้องมีเครื่องหมาย : กำหนดหลังคำสั่งในการเริ่มต้นบล็อคเสมอ อีกสองบล็อคสุดท้ายนั้นเป็นคำสั่ง Else และ For ซึ่งมีหนึ่งคำสั่งย่อยอยู่ภายใน ในภาษา Python นี้เข้มงวดกับช่องว่างภายในบล็อคมาก นั้นหมายความว่าทุกคำสั่งย่อยภายในบล็อคนั้นต้องมีจำนวนช่องว่างเท่ากันเสมอ

Literals

ในการเขียนโปรแกรม Literal คือเครื่องหมายที่ใช้แสดงค่าของค่าคงที่ในโปรแกรม ในภาษา Python นั้นมี Literal ของข้อมูลประเภทต่างๆ เช่น Integer Floating-point number และ String หรือแม้กระทั่งตัวอักษรและ boolean นี่เป็นตัวอย่างของการกำหนด Literal ให้กับตัวแปรในภาษา Python

```
a = 1
b = -1.64E3
c = True
d = "marcuscode.com"
e = 'A'
```

ในตัวอย่าง เป็นการกำหนด Literal ประเภทต่างๆ ให้กับตัวแปร ในค่าที่เป็นแบบตัวเลขนั้นสามารถกำหนดค่าลงไปโดยตรงได้ทันทีและสามารถกำหนดในรูปแบบสั้นได้อย่างในตัวแปร b และสำหรับ boolean นั้นจะเป็น True ส่วน String หรือ Character นั้นจะต้องอยู่ภายในเครื่องหมาย double quote หรือ single quote เสมอ

Expressions

Expression คือการทำงานร่วมกันระหว่างค่าตั้งแต่หนึ่งไปจนถึงหลายค่า โดยค่าเหล่านี้จะมีตัวดำเนินการสำหรับควบคุมการทำงาน ในภาษา Python นั้น Expression จะมีสองแบบคือ Boolean expression เป็นการกระทำกันของตัวแปรและตัวดำเนินการและจะได้ผลลัพธ์เป็นค่า Boolean โดยทั่วไปแล้วมักจะเป็นตัวดำเนินการ

เปรียบเทียบค่าและตัวดำเนินการตรรกศาสตร์ และ Expression ทางคณิตศาสตร์ คือการกระทำกันกับตัวดำเนินการและได้ค่าใหม่ที่ไม่ใช้ Boolean นี่เป็นตัวอย่างของ Expressions ในภาษา Python

```
a = 4
b = 5

# Boolean expressions
print(a == 4)
print(a == 5)
print(a == 4 and b == 5)
print(a == 4 and b == 8)

# Non-boolean expressions
print(a + b)
print(a + 2)
print(a * b)
print(((a * a) + (b * b)) / 2)
print("Python " + "Language")
```

ในตัวอย่าง เรามีตัวแปร a และ b และกำหนดค่าให้กับตัวแปรเหล่านี้และทำงานกับตัวดำเนินการประเภทต่างๆ ที่แสดง Expression ในรูปแบบของ Boolean expression ที่จะได้ผลลัพธ์สุดท้ายเป็นเพียงค่า True และ False เท่านั้น ส่วน Non-Boolean expression นั้นสามารถเป็นค่าใดๆ ที่ไม่ใช่ Boolean

```
True
False
True
False
9
6
20
20.5
Python Language
```

นี่เป็นผลลัพธ์การทำงานของโปรแกรมในการทำงานของ Expression ในภาษา Python

Keywords

Keyword เป็นคำที่ถูกสงวนไว้ในการเขียนโปรแกรมภาษา Python เราไม่สามารถใช้คำสั่งเหล่านี้ในการตั้งชื่อตัวแปร ชื่อฟังก์ชัน คลาส หรือ identifier ใดๆ ที่กำหนดขึ้นโดยโปรแกรมเมอร์ นี่เป็นรายการของ Keyword ในภาษา Python

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

ตัวแปรและประเภทข้อมูล

ในบทนี้ คุณจะได้เรียนรู้เกี่ยวกับตัวแปรและประข้อมูลในภาษา Python เราจะพูดถึงการประกาศตัวแปรและการนำตัวแปรไปใช้งานในโปรแกรม และเราจะอธิบายถึงข้อมูลประเภทต่างๆ ที่เป็น Primitive datatype ในภาษา Python และรวมทั้งฟังก์ชันสำหรับการใช้งานกับตัวแปร

ตัวแปร

ตัวแปร (variable) คือชื่อหรือเครื่องหมายที่กำหนดขึ้นสำหรับใช้เก็บค่าในหน่วยความจำ ตัวแปรจะมีชื่อ (identifier) สำหรับใช้ในการอ้างถึงข้อมูลของมัน ในการเขียนโปรแกรม ค่าของตัวแปรสามารถที่จะกำหนดได้ใน run-time หรือเปลี่ยนแปลงอยู่ตลอดเวลาในขณะที่โปรแกรมทำงาน (executing)

ในการเขียนโปรแกรมคอมพิวเตอร์นั้น ตัวแปรจะแตกต่างจากตัวแปรในทางคณิตศาสตร์ ค่าของตัวแปรนั้นไม่จำเป็นต้องประกอบไปด้วยสูตรหรือสมการที่สมบูรณ์เหมือนกับในคณิตศาสตร์ ในคอมพิวเตอร์ ตัวแปรนั้นอาจจะมีการทำงานซ้ำๆ เช่น การกำหนดค่าในที่หนึ่ง และนำไปใช้อีกที่หนึ่งในโปรแกรม และนอกจากนี้ยังสามารถกำหนดค่าใหม่ให้กับตัวแปรได้ตลอดเวลา ต่อไปเป็นตัวอย่างของการประกาศตัวแปรในภาษา Python

```
a = 3
b = 4.92
c = "marcuscode.com"
c = 10.5
```

ในตัวอย่าง เราได้ทำการประกาศ 3 ตัวแปร ในการประกาศตัวแปรในภาษา Python คุณไม่จำเป็นต้องระบุประเภทของตัวแปรในตอนที่ประกาศเหมือนในภาษา C ในตัวแปร a มีค่าเป็น 3 และเป็นประเภทเป็น Integer ตัว

แปร b มีค่าเป็น 4.92 และเป็นประเภทเป็น Float และตัวแปร c มีค่าเป็น "marcuscode.com" และเป็นประเภท String ภายหลังเราได้เปลี่ยนค่าของตัวแปร c เป็น 10.5 ตัวแปรกลายเป็นประเภท Float

```
a, b = 1, 2
x = y = z = 10
print("a = " , a)
print("b = " , b)
print("x = " , x)
print("y = " , y)
print("z = " , z)
```

ในภาษา Python นั้นสนับสนุนการกำหนดค่าให้กับตัวแปรหลายค่าในคำสั่งเดียว ในตัวอย่าง เป็นการกำหนดค่า 1 และ 2 กับตัวแปร a และ b ตามลำดับ และในคำสั่งต่อมาเป็นการกำหนดค่า 10 ให้กับตัวแปร x y และ z ซึ่งทำให้การเขียนโปรแกรมสะดวกและรวดเร็วมากขึ้น

```
a =  1
b =  2
x =  10
y =  10
z =  10
```

นี่เป็นผลลัพธ์การทำงานของโปรแกรม

ต่อไปจะเป็นการพูดถึงประเภทข้อมูลชนิดต่างๆ ที่ภาษา Python สนับสนุน ซึ่งจะมีอยู่สามประเภทใหญ่ๆ คือ ข้อมูลแบบตัวเลข นั้นจะแบ่งย่อยออกเป็น Integer และ Float ข้อมูลประเภท String และข้อมูลแบบลำดับ เช่น List และ Tuple ประเภทข้อมูลทั้งหมดนี้เป็น Built-in type ในภาษา Python

THE RADIO CONTROL AIRPLANE MODELER SPORT ASSOCIATION

Numbers

ในภาษา Python นั้นสนับสนุนข้อมูลแบบตัวเลข ซึ่งข้อมูลประเภทนี้จะแบ่งออกเป็น Integer Float Decimal และ Complex อย่างไรก็ตามเราจะเน้นย้ำใน Integer ซึ่งเป็นการเก็บข้อมูลแบบจำนวนเต็ม และ Float เป็นข้อมูลแบบจำนวนจริง สำหรับประเภทแบบ Decimal นั้นแตกต่างไปจาก Float คือสามารถเก็บความละเอียดของจุดทศนิยมได้มากกว่า นอกจากนี้ Python ยังสนุนตัวเลขในรูปแบบ Complex ที่แสดงในแบบ a +bj ต่อไปเป็นตวอย่างในการประกาศและใช้งานตัวแปรแบบตัวเลขในภาษา Python

```
# Integer
a = 7
b = 3
c = a + b
d = a / b

print ('a = %d' % a)
print ('b = %d' % b)
print ('c = %d' % c)
print ('d = ', d)
```

ในตัวอย่าง เป็นการประกาศและใช้งานตัวแปรประเภท Integer เราได้ทำการประกาศตัวแปรและกำหนดค่าให้กับ a และ b ในการแสดงผลในรูปแบบของ String format กับฟังก์ชัน print() นั้นจะใช้ specifier เป็น %d เราสามารถกำหนดค่าให้กับตัวแปรได้โดย Literal หรือ Expression และการหารตัวเลขในภาษา Python นั้นจะได้ค่าเป็น Float เสมอ ถึงแม้ตัวเลขทั้งสองจะเป็น Integer ก็ตาม เช่นในตัวแปร d

```
a = 7
b = 3
c = 10
d =  2.3333333333333335
```

นี่เป็นผลลัพธ์การทำงานของโปรแกรม

```
# Floating point number
speed = 34.12
pi = 22 / 7
height = 2.31E5
length = 1.3E-3

print ('speed = %f' % speed)
print ('pi = %f' % pi)
print ('height = %f' % height)
print ('length = %f' % length)
print (pi)
```

ต่อไปเป็นการประกาศและใช้งานตัวแปรประเภท Float หรือตัวเลขที่มีจุดทศนิยม ในการกำหนดค่าใก้กับตัวแปรนั้นเมื่อคุณกำหนดค่าที่มีจุดนั้นตัวเลขจะเป็นประเภท Float อัตโนมัติ เราสามารถกำหนดค่าโดยตรงหรือใน

รูปแบบของ Expression ได้ และนอกจากนี้ในภาษา Python ยังสามารถกำหนดในรูปแบบสัญกรณ์วิทยาศาสตร์ได้ เหมือนในตัวแปร height ซึ่งหมายถึง 2.31 x 10 ^ 5 และในตัวแปร length ซึ่งหมายถึง 1.3 x 10 ^ -3

```
speed = 34.120000
pi = 3.142857
height = 231000.000000
length = 0.001300
3.142857142857143
```

นี่เป็นผลลัพธ์การทำงานของโปรแกรม ซึ่งในการแสดงผลของข้อมูลประเภท Float กับการจัดรูปแบบของตัวเลขนั้นจะใช้ %f สำหรับการดูค่าเต็มของตัวเลขจริงๆ นั้นเราจะแสดงค่าของตัวเลขโดยเหมือนในคำสั่งแสดงผลค่าของ pi ในคำสั่งบรรทัดสุดท้าย

Strings

Strings นั้นเป็นประเภทข้อมูลที่สำคัญและใช้งานทั่วไปในการเขียนโปรแกรม ในภาษาเขียนโปรแกรมส่วนมากแล้วจะมีประเภทข้อมูลแบบ String และในภาษา Python เช่นกัน String เป็นลำดับของตัวอักษรหลายตัวเรียงต่อกัน ซึ่งในภาษา Python นั้น String จะอยู่ในเครื่องหมาย Double quote หรือ Single quote เท่านั้น นอกจากนี้ในภาษา Python ยังมีฟังก์ชันในการจัดการกับ String มากมายซึ่งเราจะพูดอีกครั้งในบทของ String ในบทนี้มาทำความรู้จักกับ String เบื้องต้นกันก่อน

```
name = "Mateo"
country = "Ukrain"
language = 'Python'
interest = 'Mountain Everest'
```

ในตัวอย่าง เป็นการประกาศตัวแปรประเภท String สองตัวแปรแรกเป็นการประโดยการใช้ Double quote และสองตัวแปรต่อม่เป็นการใช้ Single quote ซึ่งคุณสามารถใช้แบบไหนก็ได้ แต่มีสิ่งที่แตกต่างกันเล็กน้อยคือเกี่ยวกับการกำหนดตัวอักพิเศษหรือเรียกว่า Escape character

```
sentent1 = "What's your name?"
sentent2 = 'I\'m Mateo.'
sentent3 = "He said \"I would learn Python first\"."
sentent4 = 'His teach replied "Oh well!"'
print (sentent1)
print (sentent2)
print (sentent3)
print (sentent4)
```

ในตัวอย่าง เป็นสิ่งที่แตกต่างของการประกาศ String ทั้งสองแบบกับ Escape character ตัวอักษร ' และ " นั้นเป็น Escape character ดังนั้นในการใช้งานตัวอักษรเหล่านี้ เราจะต้องทำการใส่เครื่องหมาย \ ลงไปข้างหน้าเสมอ แต่ในภาษา Python เมื่อคุณใช้ Double quote ในการประกาศ String คุณไม่ต้องทำการ Escape character สำหรับ Single quote และในทางกลับกัน อย่างไรก็ตามเราจะพูดอีกครั้งในบทของ String

```
What's your name?
I'm Mateo.
He said "I would learn Python first".
His teach replied "Oh well!"
```

นี่เป็นผลลัพธ์การทำงานของโปรแกรมในการใช้งาน Escape character ในภาษา Python

```
site = 'marcuscode' + '.com'
tutorial = 'Python' ' Language'
print(site)
print(tutorial)
```

การทำงานอย่างหนึ่งที่สำคัญเกี่ยวกับ String ก็คือการเชื่อมต่อ String ซึ่งเป็นการนำ String ตั้งต่อสองอันขึ้นไปมาต่อกัน ในภาษา Python คุณสามารถต่อ String ได้โดยการใช้เครื่องหมาย + หรือคั่นด้วยช่องว่างหรือบรรทัดใหม่เหมือนในตัวอย่างข้างบน

```
marcuscode.com
Python Language
```

นี่เป็นผลลัพธ์การทำงานของโปรแกรม

อย่างไรก็ตาม นี่เป็นการแนะนำเกี่ยวกับ String ในเบื้องต้นเท่านั้น เพราะว่า String นั้นมีเนื้อหาเป็นจำนวนมาก คุณจะได้เรียนรู้เกี่ยวกับ String อย่างละเอียด อีกครั้งในบทของ String

Lists

Lists เป็นประเภทข้อมูลที่เก็บข้อมูลแบบเป็นชุดและลำดับ กล่าวคือมันสามารถเก็บข้อมูลได้หลายค่าในตัวแปรเดียวและมี Index สำหรับเข้าถึงข้อมูล ในภาษา Python นั้น List จะเป็นเหมือนอาเรย์ในภาษา C มันสามารถเก็บข้อมูลได้หลายตัวและยังสามารถเป็นประเภทข้อมูลที่แตกต่างกันได้อีกด้วย มาดูการประกาศและใช้งาน List ในเบื้องต้น

```
# Declare lists
numbers = [1, 2, 4, 6, 8, 19]
names = ["Mateo", "Danny", "James", "Thomas", "Luke"]
mixed = [-2, 5, 84.2, "Mountain", "Python"]

# Display lists
print(numbers)
print(names)
print(mixed)

# Display lists using the for loops
for n in numbers:
    print(n, end=" ")
print()

for n in names:
    print(n, end=" ")
print()

for n in mixed:
    print(n, end=" ")
print()
```

ในตัวอย่าง เราได้ทำการประกาศ 3 Lists โดยตัวแปรแรกนั้นเป็น List ของตัวเลข และตัวแปรที่สองเป็น List ของ String และตัวแปรสุดท้ายเป็น List แบบรวมกันของประเภทข้อมูล เราใช้ฟังก์ชัน print() ในการแสดงผลข้อมูลใน List และใช้คำสั่ง For loop ในการอ่านค่าในลิสต์และนำมาแสดงผลเช่นกัน

```
[1, 2, 4, 6, 8, 19]
['Mateo', 'Danny', 'James', 'Thomas', 'Luke']
[-2, 5, 84.2, 'Mountain', 'Python']
1 2 4 6 8 19
Mateo Danny James Thomas Luke
-2 5 84.2 Mountain Python
```

นี่เป็นผลการทำงานของโปรแกรม

```
languages = ["C", "C++", "Java", "Python", "PHP"]

print("Index at 0 = ", languages[0])
print("Index at 3 = ", languages[3])
languages[0] = "Scalar"
print("Index at 0 = ", languages[0])
```

Lists นั้นทำงานกับ Index ดังนั้นเราสามารถเข้าถึงข้อมูลของ List โดยการใช้ Index ของมันได้ ในตัวอย่างเป็นการเข้าถึงข้อมูบภายใน Index ซึ่ง Index ของ List นั้นจะเริ่มจาก 0 ไปจนถึงจำนวนทั้งหมดของมันลบด้วย 1 ในตัวอย่างเราได้แสดงผลข้อมูลของสอง List ในตำแหน่งแรกและในตำแหน่งที่ 4 ด้วย Index 0 และ 3 ตามลำดับ หลังจากนั้นเราเปลี่ยนค่าของ List ที่ตำแหน่งแรกเป็น "Scalar"

```
Index 0 =  C
Index 3 =  Python
Index 0 =  Scalar
```

นี่เป็นผลลัพธ์การทำงานของโปรแกรม ซึ่งคุณได้ทำความรู้จักกับ List ในเบื้องต้น คุณจะได้เรียนรู้เกี่ยวกับ List ในภาษา Python อย่างละเอียดอีกครั้งในบทของ List ซึ่งเราจะพูดเกี่ยวกับการดำเนินการและการใช้ฟังก์ชันของ List นอกจากนี้ Python ยังมีประเภทข้อมูลแบบ Tuple และ Dictionary ซึ่งมีรูปแบบการเก็บข้อมูลคล้ายกับ List จึงคุณจะได้เรียนในบทต่อไป

ฟังก์ชันที่ใช้กับตัวแปร

ในภาษา Python นั้นมีฟังก์ชันที่สร้างมาเพื่อให้ใช้งานกับตัวแปร เช่น ฟังก์ชันสำหรับหาขนาดของตัวแปร ฟังก์ชันในการหาประเภทของตัวแปร ฟังก์ชันลบตัวแปรออกไปในหน่วยความจำ และฟังก์ชันในการตรวจสอบว่าตัวแปรมีอยู่หรือไม่ ซึ่งในบางครั้งการเขียนโปรแกรมก็จำเป็นที่คุณอาจจะต้องมีการตรวจสอบสิ่งเหล่านี้ในขณะที่โปรแกรมทำงาน นี่เป็นตัวอย่างการใช้งาน

```
import sys

a = 8
b = 13.4
c = "Python"
d = [1, 2, 3, 4]

print('Size of a = ', sys.getsizeof(a))
print('Type of a = ', type(a))

print('Size of b = ', sys.getsizeof(b))
print('Type of b = ', type(b))

print('Size of c = ', sys.getsizeof(c))
print('Type of c = ', type(c))

print('Size of d = ', sys.getsizeof(d))
print('Type of d = ', type(d))

del a
del b, c, d

if 'a' in locals():
    print("a is exist")
else:
    print("a is not exist")
```

ในตัวอย่าง เราได้ประกาศตัวแปรกับประเภทต่างๆ เราได้ฟังก์ชัน getsizeof() สำหรับหาขนาดของตัวแปรที่มีหน่วยเป็น Byte และฟังก์ชัน type() สำหรับประเภทของตัวแปรว่าอยู่ในคลาสไหน ฟังก์ชัน del() สำหรับยกเลิกหรือลบการประกาศตัวแปรออกไปจากหน่วยความจำ และสุดท้ายเป็นการตรวจสอบว่าตัวแปรถูกประกาศและหรือยังในฟังก์ชัน locals() สำหรับตรวจสอบตัวแปรในโมดูลปัจจุบัน หรือ globals() สำหรับตรวจสอบตัวแปรในโปรแกรมทั้งหมด

```
Size of a =  14
Type of a =  <class 'int'>
Size of b =  16
Type of b =  <class 'float'>
Size of c =  31
Type of c =  <class 'str'>
Size of d =  52
Type of d =  <class 'list'>
a is not exist
```

นี่เป็นผลลัพธ์การทำงานของโปรแกรมในการใช้ฟังก์ชันที่จำเป็นกับตัวแปร

การแสดงผลด้วยฟังก์ชัน print()

ในการแสดงผลในภาษา Python นั้นจะใช้ฟังก์ชัน print() เพื่อแสดงผลข้อความ ตัวเลข หรือข้อมูลประเภทอื่นๆ ออกทางหน้าจอหรือสร้าง Http response นี่เป็นรูปแบบของการใช้งานฟังก์ชัน print() ในภาษา Python

```
print(value, ..., sep = ' ', end = '\n');
```

ในรูปแบบการใช้งาน ฟังก์ชัน print() เราสามารถส่งอาร์กิวเมนต์ได้ตั้งแต่หนึ่งถึงหลายตัวเข้าไปในฟังก์ชัน นอกจากนี้ฟังก์ชันยังมี keyword อาร์กิวเมนต์ sep ซึ่งเป็นตัวแบ่งหากอาร์กิวเมนต์ที่ส่งเข้าไปนั้นมากกว่า 1 ตัว ซึ่งมีค่า default เป็น whitespace และ keyword อาร์กิวเมนต์ end เป็นการแสดงผลในตอนท้ายของฟังก์ชัน ซึ่งมีค่า default เป็น \n หมายถึงการขึ้นบรรทัดใหม่ มาดูตัวอย่างการใช้งานฟังก์ชัน

```
print("Hello Python")
print("My name is Mateo")
print("Mercury", "Venus", "Earth")
name = "marcuscode.com"
year = 2017
print(name)
print(year)
```

ในตัวอย่าง เป็นการแสดงผลในภาษา Python โดยในคำสั่งแรกและคำสั่งที่สองนั้นเป็นการแสดงข้อความ และในคำสั่งที่สามเป็นการส่งค่าแบบหลายอาร์กิวเมนต์ และในสองคำสั่งสุดท้ายเป็นการแสดงผลข้อมูลจากตัวแปร name และตัวแปร year

```
Hello Python
My name is Mateo
Mercury Venus Earth
marcuscode.com
2017
```

นี่เป็นผลลัพธ์การทำงานของโปรแกรม

นอกจากนี้ เรายังสามารถใช้ keyword อาร์กิวเมนต์สำหรับกำหนดการแสดงผลเพื่อแบ่งแต่ละอาร์กิวเมนต์และการแสดงผลในตอนท้ายของฟังก์ชัน นี่เป็นตัวอย่างการใช้งาน

```
print("Mercury", "Venus", "Earth", sep=', ')
print("One", end=' ')
print("Two", end=' ')
print("Three", end=' ')
```

ในตัวอย่าง เป็นการใช้งาน keyword อาร์กิวเมนต์ในการจัดรูปแบบการแสดงผล โดยอาร์กิวเมนต์sep เป็นตัวแบ่งการแสดงในแต่ละอาร์กิวเมนต์ และ end เป็นตัวแบ่งการแสดงผลในแต่ละบรรทัด โดยปกติฟังก์ชัน print() จะขึ้นบรรทัดใหม่ทุกครั้ง เราสามารถใช้อาร์กิวเมนต์นี้เพื่อเปลี่ยนเป็นอย่างอื่นได้

```
Mercury, Venus, Earth
One Two Three
```

นี่เป็นผลลัพธ์การทำงานของโปรแกรม

นอกจากนี้ฟังก์ชัน print() นั้นยังสามารถส่งพารามิเตอร์ในรูปแบบของ String Formatting ได้โดยการใช้รูปแบบการแปลงข้อมูล ที่คล้ายกับการจัดรูปแบบการแสดงผลในภาษา C นี่เป็นตัวอย่างการใช้งาน

```
lang = "Python"
version = 3.6
print("%s language" % lang)
print("Version %f" % version)
print("%d" % 123)
print("%s %f %d" % (lang, version, 123))
```

ในตัวอย่าง เป็นการจัดรูปแบบการแสดงผลของ String โดยการแทรกรูปแบบของการแสดงผลใน String literal ได้ เช่น %s สำหรับการแสดงผล String %f สำหรับการแสดงผล Float และ %d สำหรับการแสดงผล Integer

```
Python language
Version 3.600000
123
Python 3.600000 123
```

นี่เป็นผลลัพธ์การทำงานของโปรแกรม สำหรับข้อมูลเพิ่มเติมเกี่ยวกับประเภทของการจัดรูปแบบ String คุณสามารถเรียนรู้เพิ่มเติมได้ที่เว็บไซต์ของ Python

ตัวดำเนินการ ในภาษา Python

ตัวดำเนินการ (Operators) คือกลุ่มของเครื่องหมายหรือสัญลักษณ์ที่ใช้ทำงานเหมือนกับฟังก์ชัน แต่แตกต่างกันตรงไวยากรณ์หรือความหมายในการใช้งาน ในภาษา Python นั้นสนับสนุนตัวดำเนินการประเภทต่างๆ สำหรับการเขียนโปรแกรม เช่น ตัวดำเนินการ + เป็นตัวดำเนินการทางคณิตศาสตร์ที่ใช้สำหรับการบวกตัวเลขเข้าด้วยกัน หรือตัวดำเนินการ > เป็นตัวดำเนินการเพื่อให้เปรียบเทียบค่าสองค่า

นี่เป็นรายการของตัวดำเนินการในภาษา Python ที่คุณจะได้เรียนในบทนี้

- Assignment operator
- Arithmetic operators
- Logical operators
- Comparison operators
- Sequence Operators
- Truth Value Testing

Assignment operator

ตัวดำเนินการที่เป็นพื้นฐานที่สุดสำหรับการเขียนโปรแกรมในทุกๆ ภาษาก็คือ ตัวดำเนินการกำหนดค่า (Assignment operator) ตัวดำเนินการนี้แสดงโดยใช้เครื่องหมายเท่ากับ (=) มันใช้สำหรับกำหนดค่าให้กับตัวแปร มาดูตัวอย่างการใช้งานในภาษา Python

```
a = 3
b = 5.29
c = b
name = 'Mateo'
my_list = [2, 5, 8, 10, 24]
x, y = 10, 20
```

ในตัวอย่าง เป็นการใช้งานตัวดำเนินการกำหนดค่าสำหรับกำหนดค่าให้กับตัวแปรประเภทต่างๆ โดยทั่วไปแล้ว ตัวดำเนินการกำหนดค่านั้นเกือบจะใช้ในทุกๆ ที่ในโปรแกรมและเป็นตัวดำเนินการที่ใช้บ่อยที่สุดของในบรรดาตัวดำเนินการทั้งหมด

Arithmetic operators

ตัวดำเนินการทางคณิตศาสตร์ (Arithmetic operators) คือตัวดำเนินการที่ใช้สำหรับการคำนวณทางคณิตศาสตร์ในพื้นฐาน เช่น การบวก การลบ การคูณ และการหาร มากไปกว่านั้น ในภาษา Python ยังมีตัวดำเนินการทางคณิตศาสตร์เพิ่มเติม เช่น การหารเอาเศษ (Modulo) การหารแบบเลขจำนวนเต็ม และการยกกำลัง เป็นต้น

นี่เป็นตารางของตัวดำเนินการทางคณิตศาสตร์ในภาษา Python

Operator	Name	Example
+	Addition	a + b
-	Subtraction	a - b
*	Multiplication	a * b
/	Division	a / b
//	Division and floor	a // b
%	Modulo	a % b
**	Power	a ** b

ในตารางข้างบน เรามีตัวดำเนินการทางคณิตศาสตร์ประเภทต่างๆ สำหรับการคำนวณเกี่ยวกับคณิตศาสตร์เบื้องต้น คุณอาจจะคุ้นเคยกับตัวดำเนินการบวก ลบ คูณ หาร ในการเรียนระดับมัธยมศึกษามาบ้างแล้ว ในภาษา Python นั้นสนับสนุนตัวดำเนินการสำหรับการหารเอาเศษเช่นเดียวกับภาษาอื่นๆ และนอกจากนี้ ยังมีตัวดำเนินการแบบการหารที่ได้ผลลัพธ์เป็นจำนวนเต็ม และการหาเลขยกกำลังเพิ่มเข้ามา มาดูตัวอย่างการใช้ตัวดำเนินการประเภทต่างๆ ในภาษา Python

```
a = 5
b = 3
print("a + b = ", a + b)
print("a - b = ", a - b)
print("a * b = ", a * b)
print("a / b = ", a / b)
print("a // b = ", a // b) # floor number to integer
print("a % b = ", a % b)   # get division remainder
print("a ** b = ", a ** b) # power
```

ในตัวอย่าง เราได้ประกาศตัวแปร a และ b และกำหนดค่าให้กับตัวแปรทั้งสองเป็น 5 และ 3 ตามลำดับ ในสี่ตัวดำเนินการแรกเป็นการดำเนินการทางคณิตศาสตร์พื้นฐาน สำหรับตัวดำเนินการ // เป็นการหารเช่นเดียวกัน แต่ผลลัพธ์ของการหารนั้นจะตัดส่วนที่เป็นทศนิยมทิ้งไป ส่วนตัวดำเนินการ % นั้นเป็นการหารโดยผลลัพธ์จะเป็นเศษของการหารแทน ส่วนสุดท้าย ** นั้นแทนการยกกำลัง

```
a + b =  8
a - b =  2
a * b =  15
a / b =  1.6666666666666667
a // b =  1
a % b =  2
a ** b =  125
```

นี่เป็นผลลัพธ์การทำงานของโปรแกรมในการใช้งานตัวดำเนินการทางคณิตศาสตร์

Comparison operators

ตัวดำเนินการเปรียบเทียบ (Comparison operators) คือตัวดำเนินการที่ใช้สำหรับเปรียบเทียบค่าหรือค่าในตัวแปร ซึ่งผลลัพธ์ของการเปรียบเทียบนั้นจะเป็น True หากเงื่อนไขเป็นจริง และเป็น False หากเงื่อนไขไม่เป็นจริง ตัวดำเนินการเปรียบเทียบมักจะใช้กับคำสั่งตรวจสอบเงื่อนไข if และคำสั่งวนซ้ำ for while เพื่อควบคุมการทำงานของโปรแกรม

นี่เป็นตารางของตัวดำเนินการเปรียบเทียบในภาษา Python

Operator	Name	Example
<	Less than	a < b
<=	Less than or equal	a <= b
>	Greater than	a > b
>=	Greater than or equal	a >= b
==	Equal	a == b
!=	Not equal	a != b
is	Object identity	a is b
is not	Negated object identity	a is not b

ในตาราง แสดงให้เห็นถึงตัวดำเนินการเปรียบเทียบประเภทต่างๆ เช่น การเปรียบเทียบความเท่ากัน โดยคุณสามารถใช้ตัวดำเนินการเปรียบเทียบเพื่อเปรียบเทียบว่าค่าในตัวแปรนั้นเท่ากันหรือไม่ หรือการเปรียบเทียบค่ามากกว่าหรือน้อยกว่า ต่อไปมาดูตัวอย่างการใช้งานตัวดำเนินการเปรียบเทียบในภาษา Python

```
# Constant comparison
print('4 == 4 :', 4 == 4)
print('1 < 2:', 1 < 2)
print('3 > 10:', 3 > 10)
print('2 <= 1.5', 2 <= 1.5)
print()

# Variable comparison
a = 10
b = 8
print('a != b:', a != b)
print('a - b == 2:', a - b == 2)
print()

# Type comparison
name = 'Mateo'
str = 'Python'
number = 10
print(name is number)
print(name is not number)
print(name is not str)
```

ในตัวอย่าง เป็นการเปรียบเทียบค่าประเภทต่างๆ ในคำสั่งกลุ่มแรกนั้นเป็นการใช้ตัวดำเนินการเปรียบเทียบกับค่าคงที่ ในกลุ่มที่สองเป็นการใช้งานกับตัวแปร และในกลุ่มที่สามนั้นเป็นการตรวจสอบความเท่ากันของออบเจ็ค ซึ่งถ้าหากเงื่อนไขเป็นจริงจะได้ผลลัพธ์เป็น True และถ้าหากไม่จริงจะได้ผลลัพธ์เป็น False

THE RADIO CONTROL AIRPLANE MODELER SPORT ASSOCIATION

```
4 == 4 : True
1 < 2: True
3 > 10: False
2 <= 1.5 False

a != b: True
a - b == 2: True

False
True
True
```

นี่เป็นผลลัพธ์การทำงานของโปรแกรมในการใช้ตัวดำเนินการเปรียบเทียบ

ในการใช้ตัวดำเนินการเปรียบเทียบนั้น เรามักจะใช้กับคำสั่งควบคุมการทำงานของโปรแกรม เช่น คำสั่ง if หรือ for เพื่อให้คุณเข้าใจมากขึ้น มาดูตัวอย่างการนำไปใช้ในการเขียนโปรแกรมกับคำสั่งเหล่านี้

```
n = int(input("Enter a number: "))

if n % 2 == 0:
    print('%d is even number' % n)
else:
    print('%d is odd number' % n)

if n > 0:
    print('%d is positive number' % n)
elif n == 0:
    print('%d is zero' % n)
else:
    print('%d is negative number' % n)
```

ในตัวอย่าง เป็นโปรแกรมในการรับค่าตัวเลขแบบ Integer แล้วเราใช้คำสั่ง if ในการตรวจสอบตัวเลขสองอย่างคือ ตรวจสอบว่าเป็นเลขคู่หรือเลขคี่ และตรวจสอบว่าเป็นจำนวนเต็มบวก เต็มลบ หรือศูนย์ ในเงื่อนไข n % 2 == 0 นั้นเป็นการตรวจโดยการหารเอาเศษ ซึ่งมีความหมายว่า หากตัวเลขนั้นหารแล้วมีเศษเท่ากับ 0 นั้นหมายความว่าตัวเลขจะเป็นเลขคู่และในบล็อคคำสั่ง if จะทำงาน และถ้าไม่เป็นจริงโปรแกรมจะทำงานในบล็อคคำสั่ง else แทนในบล็อคของคำสั่ง if ต่อมา เป็นการตรวจสอบว่าตัวเลขนั้นเป็นตัวเลขที่มากกว่า น้อยกว่า หรือเท่ากับศูนย์

```
Enter a number: 5
5 is odd number
5 is positive number
```

```
Enter a number: -1
-1 is odd number
-1 is negative number
```

นี่เป็นผลลัพธ์การทำงานของโปรแกรมสองครั้ง เมื่อเรากรอกตัวเลขเป็น 5 และ -1 ตามลำดับ และโปรแกรมจะบอกเราว่าตัวเลขนั้นเป็นแบบไหน

Logical operators

ตัวดำเนินการตรรกศาสตร์ (Logical operators) คือตัวดำเนินการที่ใช้สำหรับประเมินค่าทางตรรกศาสตร์ ซึ่งเป็นค่าที่มีเพียงจริง (True) และเท็จ (False) เท่านั้น โดยทั่วไปแล้วเรามักใช้ตัวดำเนินการตรรกศาสตร์ในการเชื่อม Boolean expression ตั้งแต่หนึ่ง expression ขึ้นไปและผลลัพธ์สุดท้ายที่ได้นั้นจะเป็น Boolean
นี่เป็นตารางของตัวดำเนินการตรรกศาสตร์ในภาษา Python

Operator	Example	Result
and	a and b	True if a and b are true, else False
or	a or b	True if a or b are true, else False
not	not a	True if a is False, else True

ในภาษา Python นั้นมีตัวดำเนินการทางตรรกศาสตร์ 3 ชนิด คือ ตัวดำเนินการ and เป็นตัวดำเนินการที่ใช้เชื่อมสอง Expression และได้ผลลัพธ์เป็น True หาก Expression ทั้งสองเป็น True ไม่เช่นนั้นจะได้ผลลัพธ์เป็น False ตัวดำเนินการ or เป็นตัวดำเนินการที่ใช้เชื่อมสอง Expression และได้ผลลัพธ์เป็น True หากมีอย่างน้อยหนึ่ง Expression ที่เป็น True ไม่เช่นนั้นได้ผลลัพธ์เป็น False และตัวดำเนินการ not ใช้ในการกลับค่าจาก True เป็น False และในทางกลับกัน มาดูตัวอย่างการใช้งาน

```
print('Log in page')
username = input('Username: ')
password = input('Password: ')

if (username == 'mateo' and password == '3456'):
    print('Welcome Mateo, you\'ve logged in.')
else:
    print('Invalid username or password.')
```

ในตัวอย่าง เราได้สร้างโปรแกรมจำลองในการเข้าสู่ระบบของหน้าเว็บไซต์ ในการที่จะเข้าสู่ระบบ ผู้ใช้ต้องกรอกชื่อผู้ใช้และรหัสผ่านให้ถูกต้อง ดังนั้นเราจึงใช้ตัวดำเนินการ and เพื่อตรวจสอบว่าทั้งชื่อผู้ใช้และรหัสผ่านนั้นถูกต้อง ทำให้เงื่อนไขเป็นจริงและในบล็อคคำสั่ง if จะทำงาน

```
Log in page
Username: mateo
Password: 3456
Welcome Mateo, you've logged in.
```

นี่เป็นผลลัพธ์การทำงานของโปรแกรม เมื่อเรากรอกชื่อผู้ใช้เป็น "mateo" และรหัสผ่านเป็น "3456" ซึ่งถูกต้องทั้งสองอย่างทำให้สามารถเข้าสู่ระบบได้สำเร็จ

Sequence Operators

ในภาษา Python มีตัวดำเนินการในการตรวจสอบการเป็นสมาชิกในออบเจ็คประเภท List Tuple และ Dictionary ตัวดำเนินการ in ใช้ในการตรวจสอบถ้าหากค่านั้นมีอยู่ในออบเจ็ค ถ้าหากพบจะได้ผลลัพธ์เป็น True และหากไม่พบจะได้ผลลัพธ์เป็น False และตัวดำเนินการ not in นั้นจะทำงานตรงกันข้าม หากไม่พบจะได้ผลลัพธ์เป็น True แทน

นี่เป็นตารางของตัวดำเนินการในการตรวจสอบการเป็นสมาชิกในออบเจ็ค ในภาษา Python

Operator	Name	Example
in	Object memberships	a in b
not in	Negated object memberships	a not in b

มาดูตัวอย่างการใช้งานของตัวดำเนินการเหล่านี้ เราจะใช้ในการตรวจสอบการมีอยู่ของข้อมูลใน List และ Dictionary

THE RADIO CONTROL AIRPLANE MODELER SPORT ASSOCIATION

```
names = ['Mateo', 'David', 'Andrill', 'Joshep']

if 'Mateo' in names:
    print('\'Mateo\' exist in the list')
else:
    print('\'Mateo\' not exist in the list')

if 'Jonathan' in names:
    print('\'Jonathan\' exist in the list')
else:
    print('\'Jonathan\' not exist in the list')


numbers = {'1': 'one', '3': 'three', '2': 'two', '5': 'five'}

if 'one' in numbers.values():
    print('\'one\' exist in the dictionary values')
else:
    print('\'one\' not exist in the dictionary values')

if '7' in numbers.keys():
    print('\'7\' exist in the dictionary keys')
else:
    print('\'7\' not exist in the dictionary keys')
```

ในตัวอย่าง เป็นการตรวจสอบข้อมูลใน List และ Dictionary ในโปรแกรมของเรามีตัวแปร List names ซึ่งมีรายชื่ออยู่ภายใน เราใช้คำสั่ง if เพื่อตรวจสอบว่า "Mateo" นั้นมีอยู่ใน List หรือไม่ ผลลัพธ์ที่ได้นั้นจะเป็นจริงเพราะชื่อมีอยู่

และต่อมาเราตรวจสอบ "Jonathan" นั้นไม่พบชื่อดังกล่าวใน List ต่อมาเป็นการตรวจสอบการมีอยู่ของข้อมูลใน Dictionary เนื่องจาก Dictionary นั้นเป็นข้อมูลที่เก็บในคู่ของ Key และ Values เพื่อตรวจสอบกับ Key เราต้องใช้เมธอด keys() และเมธอด values() สำหรับ Value

```
'Mateo' exist in the list
'Jonathan' not exist in the list
'one' exist in the dictionary values
'7' not exist in the dictionary keys
```

นี่เป็นผลลัพธ์การทำงานของโปรแกรม

Truth Value Testing

เนื่องจากตัวแปรในภาษา Python นั้นเป็นประเภทข้อมูลแบบไดนามิกส์ ดังนั้นออบเจ็คต่างๆ นั้นสามารถที่จะทำมาประเมินสำหรับค่าความจริง โดยการใช้คำสั่งตรวจสอบเงื่อนไขเช่น if หรือ while หรือการกระทำเพื่อตรวจหาค่า boolean โดยค่าข้างล่างนี้เป็นค่าที่ถูกประเมินเป็น False

- None
- False
- ค่าศูนย์ของข้อมูลประเภทตัวเลขใดๆ เช่น 0, 0L, 0.0, 0j
- ข้อมูลแบบลำดับที่ว่างปล่าว เช่น '', (), []
- ข้อมูลแบบ map ที่ว่างเปล่า {}
- ตัวแปรจากคลาสที่ผู้ใช้สร้างขึ้น และคลาสดังกล่างถูกกำหนดเมธอด __nonzero__() หรือ __len__() และเมธอดเหล่านี้ส่งค่ากลับเป็นศูนย์หรือค่า Boolean False

ส่วนค่าอื่นๆ ที่นอกเหนือจากที่ได้กล่าวไปนั้นจะถูกประเมินเป็น True ทั้งหมด และออบเจ็คของประเภทใดๆ ก็เป็น True เช่นกัน

ในบทนี้ คุณเรียนรู้เกี่ยวกับตัวดำเนินการในภาษา Python เราได้ครอบคลุมการใช้งานตัวดำเนินการประเภทต่างๆ และตัวอย่างในประยุกต์ใช้งานตัวดำเนินการเหล่านี้ในการเขียนโปรแกรม และหลักในการประเมินค่าความจริงของตัวแปรและออบเจ็ค

คำสั่งเลือกเงื่อนไข

ในบทนี้ คุณจะได้เรียนเกี่ยวกับคำสั่งเลือกเงื่อนไขในภาษา Python เราจะพูดถึงการควบคุมการทำงานโปรแกรมด้วยคำสั่ง if, if else และ elif เพื่อให้โปรแกรมสามารถทำงานซับซ้อนและมีประสิทธิภาพมากขึ้น ยกตัวอย่างเช่น เครื่องปรับอากาศจะทำงานอัตโนมัติถ้าหากอุณหภูมิในห้องสูงหรือต่ำเกินไป หรือรถยนต์จะแสดงสัญญาณเตือนหากน้ำมันกำใกล้จะหมด เป็นต้น ซึ่งทั้งหมดนี้เกิดการกำหนดเงื่อนไขการทำงานให้โปรแกรม มาเริ่มกับคำสั่ง if ในภาษา Python

คำสั่ง if

คำสั่ง if เป็นคำสั่งที่ใช้ควบคุมการทำงานของโปรแกรมที่เป็นพื้นฐานและง่ายที่สุด เราใช้คำสั่ง if เพื่อสร้างเงื่อนไขให้โปรแกรมทำงานตามที่เราต้องการเมื่อเงื่อนไขนั้นตรงกับที่เรากำหนด เช่น การตรวจสอบค่าในตัวแปรกับตัวดำเนินการประเภทต่างๆ นี่เป็นรูปแบบของการใช้งานคำสั่ง if ในภาษา Python

```
if expression:
    # statements
```

ในตัวอย่าง เป็นรูปแบบของการใช้งานคำสั่ง if และ expression เป็นเงื่อนไขที่สร้างจากตัวดำเนินการประเภทต่างๆ ที่เป็น boolean expression โดยโปรแกรมจะทำงานในบล็อคคำสั่ง if ถ้าหากเงื่อนไขเป็น True ไม่เช่นนั้นโปรแกรมจะข้ามการทำงานไป ในบล็อคของคำสั่ง if จะประกอบไปด้วยคำสั่งการทำงานของโปรแกรม คำสั่งทั้งหมดในบล็อคต้องมีระยะเว้นช่องว่างที่เท่ากัน ต่อไปมาดูตัวอย่างการใช้งานคำสั่ง if ในภาษา Python

```
n = 10
if n == 10:
    print('n equal to 10')

logged_in = False
if not logged_in:
    print('You must login to continue')

m = 4
if m % 2 == 0 and m > 0:
    print('m is even and positive numbers')

if 3 > 10:
    print('This block isn\'t executed')
```

ในตัวอย่าง เป็นการใช้งานคำสั่ง if เพื่อกำหนดให้โปรแกรมทำงานตามเงื่อนไขต่างๆ ในบล็อคแรกเป็นการตรวจสอบค่าในตัวแปร n ว่าเท่ากับ 10 หรือไม่ เนื่องจากค่าในตัวแปรนั้นเท่ากับ 10 ทำให้เงื่อนเป็นจริง และโปรแกรมทำงานในบล็อคของคำสั่ง if และต่อมาเรามีตัวแปร boolean logged_in เก็บค่าสถานะการเข้าสู่ระบบ เราได้ทำการตรวจสอบโดยการใช้ตัวดำเนินการ not สำหรับตรวจสอบว่าถ้าหากผู้ใช้ไม่เข้าสู่ระบบ จะแสดงข้อความบอกว่าต้องเข้าระบบก่อนที่จะใช้งาน

ต่อมาเป็นการตรวจสอบค่าในตัวแปร m ว่าเป็นทั้งจำนวนเต็มบวกและจำนวนคู่หรือไม่ เราได้ใช้ตัวดำเนินการ and เพื่อเชื่อม expression ย่อยทั้งสอง และเงื่อนไขเป็นจริงทำให้ในบล็อคคำสั่ง if ทำงาน สุดท้ายเป็นเปรียบเทียบค่าของตัวเลข เราได้เปรียบว่า 3 มากกว่า 10 หรือไม่ เนื่องจากเงื่อนไขเป็น False ทำให้โปรแกรมข้ามการทำงานบล็อคนี้ไป

```
n equal to 10
You must login to post
m is even and positive numbers
```

นี่เป็นผลลัพธ์การทำงานของโปรแกรม คุณจะเห็นว่าในสามบล็อคแรกของคำสั่ง if นั้นทำงานเพราะว่าเงื่อนไขเป็นจริงหรือ True และในบล็อคสุดท้ายไม่ทำงานเพราะเงื่อนไขไม่เป็นจริงหรือ False

คำสั่ง if else

หลังจากที่คุณได้รู้จักกับคำสั่ง if ไปแล้ว อีกคำสั่งหนึ่งที่ทำงานควบคู่กับคำสั่ง if คือคำสั่ง else clause โดยโปรแกรมจะทำงานในคำสั่ง else ถ้าหากเงื่อนไขในคำสั่ง if นั้นไม่เป็นจริง กล่าวอีกนัยหนึ่ง มันจะทำงานเมื่อเงื่อนไขก่อนหน้านั้นไม่เป็นจริงหรือเป็นเงื่อนไข Default มาดูตัวอย่างการใช้งาน if else ในภาษา Python

```
n = 5
if n == 10:
    print('n equal to 10')
else:
    print('n is something else except 10')

name = 'James'
if name == 'Mateo':
    print('Hi, Mateo.')
else:
    print('Who are you?')

money = 300
if money >= 350:
    print('You can buy an iPad')
else:
    print('You don\'t have enough money to buy an iPad')
```

ในตัวอย่าง เป็นโปรแกรมเพื่อทดสอบการทำงานของคำสั่ง else เราได้เพิ่มบล็อคของคำสั่ง else เข้ามาหลังจากคำสั่ง if ซึ่งโค้ดในบล็อคของคำสั่ง else จะทำงาน ถ้าหากเงื่อนไขใน if ไม่เป็นจริง นั่นหมายถึงโปรแกรมของเราสามารถทำงานได้เพียงหนึ่งทางเลือกเท่านั้น

ในการตรวจสอบครั้งแรก เป็นการตรวจสอบค่าในตัวแปร n ว่าเท่ากับ 10 หรือไม่ เพราะว่าค่าในตัวแปรนั้นเป็น 5 ทำให้เงื่อนไขไม่เป็นจริง และโปรแกรมทำงานในบล็อคคำสั่ง else แทน ต่อมาเป็นการตรวจสอบชื่อในตัวแปร name ว่าเป็น"Mateo"หรือไม่ เพราะว่าค่าในตัวแปรนั้นเป็น "James" ทำให้โปรแกรมทำงานในบล็อคคำสั่ง else และสุดท้ายนั้นเราตรวจสอบว่าหากมีเงินในตัวแปร money มากกว่าหรือเท่ากับ 350 จะได้ซื้อ iPad เพราะว่าเงินไม่พอ โปรแกรมจึงบอกว่าเงินไม่พอที่จะซื้อ

```
n is something else except 10
Who are you?
You don't have enough money to buy an iPad
```

นี่เป็นผลลัพธ์การทำงานของโปรแกรมซึ่งจะทำงานในบล็อคของคำสั่ง else ทั้งหมดเพราะเงื่อนไขไม่เป็นจริงทั้งหมด

คำสั่ง if elif

คำสั่ง elif นั้นเป็นคำสั่งที่ใช้สำหรับสร้างเงื่อนไขแบบหลายทางเลือกให้กับโปรแกรมที่มีการทำงานเช่นเดียวกับ switch case ในภาษาอื่นๆ คำสั่ง elif นั้นต้องใช้หลังจากคำสั่ง if เสมอและสามารถมี else ได้ในเงื่อนไขสุดท้าย มาดูตัวอย่างการใช้งานคำสั่ง elif ในภาษา Python

```
print('Welcome to marcuscode\'s game')
level = input('Enter level (1 - 4): ')

if level == '1':
    print('Easy')
elif level == '2':
    print('Medium')
elif level == '3':
    print('Hard')
elif level == '4':
    print('Expert')
else:
    print('Invalid level selected')
```

ในตัวอย่าง เป็นโปรแกรมจำลองในการเลือกโหมดของการเล่นเกม เราได้ให้ผู้ใช้กรอกค่าระหว่าง 1 -4 เพื่อใช้ในการเปรียบเทียบกับระดับความยากของเกม โดยที่ 1 เป็นระดับที่ง่ายที่สุด และ 4 นั้นเป็นระดับที่ยากที่สุด คุณจะเห็นว่าเราได้ให้คำสั่ง elif เพราะเรามีเงื่อนไข 4 แบบ และคำสั่ง else ในการณีที่ตัวเลขที่ผู้เล่นกรอกเข้ามานั้นไม่ตรงกับเงื่อนไขใดๆ ก่อนหน้าเลย

```
Welcome to marcuscode's game
Enter level (1 - 4): 4
Expert
```

```
Welcome to marcuscode's game
Enter level (1 - 4): 7
Invalid level selected
```

นี่เป็นผลลัพธ์การทำงานของโปรแกรมเมื่อเรากรอก 4 และ 7 ตามลำดับ เมื่อเรากรอก 4 นั้นโปรแกรมตรงกับเงื่อนไขของ elif ที่ level == 4 และเมื่อเรากรอก 7 โปรแกรมไม่ตรงกับเงื่อนไขใดๆ เลยทำให้ทำงานในบล็อคของคำสั่ง else

คำสั่งวนซ้ำ

ในบทนี้ คุณจะได้เรียนรู้คำสั่งวนซ้ำในภาษา Python เราจะพูดถึงการควบคุมการทำงานโดยการใช้คำสั่ง while loop และ for loop คำสังเหล่านี้สามารถควบคุมโปรแกรมให้ทำงานซ้ำๆ ในเงื่อนไขที่กำหนดและเพิ่มความสามารถของการเขียนโปรแกรม ตัวอย่างของการทำงานซ้ำๆ นั้นพบเห็นเห็นได้ทั่วไปในชีวิตประจำวัน เช่น โปรแกรมพยากรณ์สภาพอากาศที่เกิดขึ้นในทุกๆ วัน หรือการไปทำงานของคุณในทุกๆ เช้า เป็นต้น ดังนั้นแนวคิดเหล่านี้จึงถูกนำมาใช้กับการเขียนโปรแกรม

คำสั่ง while loop

while loop เป็นคำสั่งวนซ้ำที่ง่ายและพื้นฐานที่สุดในภาษา Python คำสั่ง while loop นั้นใช้ควบคุมโปรแกรมให้ทำงานบางอย่างซ้ำๆ ในขณะที่เงื่อนไขของลูปนั้นยังคงเป็นจริงอยู่ นี่เป็นรูปแบบของการใช้งานคำสั่ง while loop ในภาษา Python

```
while expression:
    # statements
```

ในรูปแบบการใช้งานคำสั่ง while loop นั้น เราสร้างลูปด้วยคำสั่ง while และตามด้วยการกำหนด expression ซึ่งเป็นเงื่อนไขที่จะให้โปรแกรมทำงาน ซึ่งโปรแกรมจะทำงานจนกว่าเงื่อนไขจะเป็น False และสิ้นสุดการทำงานของลูป ภายในบล็อคคำสั่ง while นั้นประกอบไปด้วยคำสั่งการทำงานของโปรแกรม ต่อไปมาดูตัวอย่างโปรแกรมนับเลขที่แสนคลาสสิคด้วยการใช้คำสั่ง while loop ในภาษา Python

```
i = 1

while i <= 10:
    print(i, end = ', ')
    i = i + 1
```

ในตัวอย่าง โปรแกรมในการแสดงตัวเลข 1 ถึง 10 โดยการใช้คำสั่ง while loop ในตอนแรก เราได้ประกาศตัวแปร i และกำหนดค่าให้กับตัวแปรเป็น 1 หลังจากนั้นเราสร้างเงื่อนไขสำหรับ while loop เป็น i <= 10 นั่นหมายความว่าโปรแกรมจะทำงานในขณะที่ค่าในตัวแปร i ยังคงน้อยกว่าหรือเท่ากับ 10 และเราแสดงผลค่าของ i ในบล็อคของคำสั่ง while และเราเพิ่มค่าของตัวแปรขึ้นทุกครั้งหลังจากที่แสดงผลเสร็จ ถ้าหากคุณไม่เพิ่มค่าของ i ลูปจะทำงานไม่มีวันหยุดหรือเรียกว่า Infinite loop

```
1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
```

นี่เป็นผลลัพธ์การทำงานของโปรแกรมในการแสดงตัวเลข 1 ถึง 10 โดยการใช้คำสั่งวนซ้ำ คุณจะเห็นว่าเราสามารถเขียนโปรแกรมได้ง่ายขึ้นโดยที่คุณไม่จำเป็นต้องใช้ฟังก์ชัน print()เพื่อแสดงผล 10 ครั้ง มาดูตัวอย่างการใช้งาน loop เพิ่มเติม

```
# decrease numbers by 5, start from 50
i = 50

while i >= 0:
    print(i, end = ', ')
    i = i - 5

# an empty line
print()

# print number series for 2n + 1
# where 1 <= n <= 10
n = 1

while n <= 10:
    print(2 * n + 1, end = ', ')
    n = n + 1
```

ขึ้นกับว่าคุณต้องการใช้ loop ทำอะไร ต่อมาเป็นตัวอย่างของโปรแกรมในการแสดงตัวเลขตั้งแต่ 50 จนถึง 0 โดยลดค่าลงมาทีละ 5 และในลูปที่สองเป็นการแสดงลำดับของอนุกรมเลขคณิตจากสูตร *2n + 1* โดย n มีค่าตั้งแต่ 1 ถึง 10

THE RADIO CONTROL AIRPLANE MODELER SPORT

```
50, 45, 40, 35, 30, 25, 20, 15, 10, 5, 0,
3, 5, 7, 9, 11, 13, 15, 17, 19, 21,
```

นี่เป็นผลลัพธ์การทำงานของโปรแกรมในการใช้ while loop แสดงค่าที่ลดลงและแสดงลำดับของอนุกรมเลขคณิต

นอกจากการใช้งานคำสั่งวนซ้ำกับการแสดงผลแล้ว เรายังสามารถใช้งานเพื่อควบคุมการรับค่าจากผู้ใช้ได้ ต่อไปเป็นตัวอย่างของโปรแกรมในการใช้คำสั่ง while loop เพื่อวนรับค่าจากผู้ใช้ทางคีย์บอร์ดและนำค่าเหล่านั้นมาแสดงผล

```
number = []
MAX_INPUT = 10

# getting input into list
print('Enter %d numbers to the list' % MAX_INPUT)

i = 1
while i <= MAX_INPUT:
    print('Number %d: ' % i, end = '')
    n = int(input())
    number.append(n)
    i += 1

# displaying numbers from list
print('Your numbers in the list')

sum = 0

i = 1
while i <= MAX_INPUT:
    print(number[i - 1], end = ', ')
    sum += number[i - 1]
    i += 1

print('\nSum = %d' % sum)
print('Average = %f' % (sum / MAX_INPUT))
```

ในตัวอย่าง เป็นโปรแกรมวนอ่านค่าจากผู้ใช้โดยการใช้คำสั่ง while loop เรามีตัวแปร MAX_INPUT สำหรับกำหนดตัวเลขที่ต้องการอ่านค่า ซึ่งโปรแกรมแบ่งออกเป็นสอง loop คือการรับค่าและการแสดงผล

```
i = 1
while i <= MAX_INPUT:
    print('Number %d: ' % i, end = '')
    n = int(input())
    number.append(n)
    i += 1
```

นี่เป็นส่วนของการรับค่า เราได้กำหนดเงื่อนไขให้กับ loop เป็น i <= MAX_INPUT เพื่อให้โปรแกรมวนรับค่าจากผู้ใช้เป็นตัวเลขจำนวน 10 ตัว หลังจากนั้นเราเก็บตัวเลขที่อ่านมาได้ลงไปใน List number โดยการใช้เมธอด append()

```
sum = 0


i = 1
while i <= MAX_INPUT:
    print(number[i - 1], end = ', ')
    sum += number[i - 1]
    i += 1
```

นี่เป็นส่วนของการแสดงผลและการหาผลรวมของตัวเลขใน List ไว้ในตัวแปร sum โดยเราใช้ตัวแปร i เป็น Index ในการเข้าถึงข้อมูลภายในลิสต์ และหลังจากนั้นเราแสดงผลรวมของตัวเลขที่รับเข้ามา และหาค่าเฉลี่ยของตัวเลขทั้งหมด

THE RADIO CONTROL AIRPLANE MODELER SPORT
ASSOCIATION

```
Enter 10 numbers to the list
Number 1: 5
Number 2: 8
Number 3: 12
Number 4: 34
Number 5: 23
Number 6: 17
Number 7: 9
Number 8: 10
Number 9: 43
Number 10: 20
Your numbers in the list
5, 8, 12, 34, 23, 17, 9, 10, 43, 20,
Sum = 181
Average = 18.100000
```

นี่เป็นผลลัพธ์การทำงานของโปรแกรม ตอนนี้คุณอาจจะเริ่มมีไอเดียในการใช้งานลูปกับการเขียนโปรแกรมแล้ว จากตัวอย่างของเรา แน่นอนว่าคุณสามารถรับค่าตัวเลขเป็น 20 หรือ 100 ตัวเลขก็ได้เพียงแค่เปลี่ยนค่าในตัวแปร MAX_INPUT เป็นจำนวนที่ต้องการ

คำสั่ง for loop

คำสั่ง for loop เป็นคำสั่งวนซ้ำที่ใช้ควบคุมการทำงานซ้ำๆ ในจำนวนรอบที่แน่นอน ในภาษา Python นั้นคำสั่ง for loop จะแตกต่างจากภาษาอื่นๆ อย่างภาษา C มันมักจะใช้สำหรับการวนอ่านค่าภายในออบเจ็ค เช่น ลิสต์หรือออบเจ็คจากฟังก์ชัน range() เป็นต้น มาดูตัวอย่างการใช้งานคำสั่ง for ในภาษา Python

THE RADIO CONTROL AIRPLANE MODELER SPORT ASSOCIATION

```
# loop through string
site = 'marcuscode'
for n in site:
    print(n)

# loop through list
names = ['Mateo', 'John', 'Eric', 'Mark', 'Robert']
for n in names:
    print(n)

numbers = [10, 20, 30, 40, 50, 60, 70, 80]
for n in numbers:
    print(n)
```

ในตัวอย่าง เป็นการใช้คำสั่ง for loop ในการวนอ่านค่าในตัวแปร String และอ่านข้อมูลภายในลิสต์ ในลูปแรกเป็นการวนอ่านค่าตัวอักษรในตัวแปร String site โดยโปรแกรมจะวนอ่านค่าทีละตัวมาเก็บไว้ในตัวแปร n ซึ่งเป็นพารามิเตอร์ของคำสั่ง for loop และวนอ่านค่าจนครบทุกตัวอักษรและจบการทำงานของ loop และอีกในสอง loop ต่อมาเป็นการใช้คำสั่ง for loop ในการวนอ่านข้อมูลภายในลิสต์ของ String และตัวเลข

THE RADIO CONTROL AIRPLANE MODELER SPORT
ASSOCIATION

```
m
a
r
c
u
s
c
o
d
e
Mateo
John
Eric
Mark
Robert
10
20
30
40
50
60
70
80
```

นี่เป็นผลลัพธ์การทำงานของโปรแกรมในการใช้คำสั่ง for loop วนอ่านค่าภายในออบเจ็ค String และ List ในภาษา Python

คำสั่ง for loop กับฟังก์ชัน range()

ในภาษา Python เรามักจะใช้คำสั่ง for loop กับฟังก์ชัน range() ในการวนอ่านค่าออบเจ็คของตัวเลข ฟังก์ชัน range() นั้นเป็น built-in ฟังก์ชันใช้สำหรับสร้างออบเจ็คของตัวเลข โดยมีพารามิเตอร์ 3 ตัว คือตัวเลขเริ่มต้น ตัวเลขสุดท้าย และค่าที่เปลี่ยนแปลงในลำดับของตัวเลข มาดูตัวอย่างการใช้งานฟังก์ชัน range() ในภาษา Python

```
a = list(range(10))
b = list(range(1, 11))
c = list(range(0, 30, 5))
d = list(range(0, -10, -1))

print(a)
print(b)
print(c)
print(d)
```

ในตัวอย่าง เป็นการสร้างออบเจ็คตัวเลขจากฟังก์ชัน range() หลังจากนั้นเราแปลงจากออบเจ็คให้เป็นลิสต์ด้วยฟังก์ชัน list() ในตัวแปร a นั้นเราใช้พารามิเตอร์เดียวคือ 10 เข้าไปในฟังก์ชัน ซึ่งเป็นการสร้างออบเจ็คของตัวเลขจาก 0 ถึง 9 (ไม่รวม 10) ในตัวแปร b นั้นใช่พารามิเตอร์สองตัวในการสร้างจาก 1 ถึง 10 (ไม่รวม 11) ต่อมาในตัวแปร c และ d เป็นการใช้งานพารามิเตอร์ครบทุกตัว โดยพารามิเตอร์ตัวสุดท้ายเป็นค่าที่เพิ่มและลดในลำดับของตัวเลข

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[0, 5, 10, 15, 20, 25]
[0, -1, -2, -3, -4, -5, -6, -7, -8, -9]
```

นี่เป็นผลลัพธ์การทำงานของโปรแกรมจากการสร้างออบเจ็คของตัวเลขด้วยฟังก์ชัน range()หลังจากนั้นแปลงให้เป็นลิสต์ด้วยฟังก์ชัน list()

ในตัวอย่าง คุณได้เห็นวิธีการแสดงออบเจ็คของตัวเลขด้วยฟังก์ชัน range() แล้ว นั่นเป็นเหตุผลที่เราจะนำมาใช้งานกับคำสั่ง for loop เพื่อวนอ่านค่าของตัวเลขในออบเจ็ค ต่อไปเป็นการแสดงผลเพื่อแสดงตัวเลข 1 ถึง 10

THE RADIO CONTROL AIRPLANE MODELER SPORT ASSOCIATION

```
for i in range(1, 11):
    print(i, end = ', ')

print()
for i in range(10, 0, -1):
    print(i, end = ', ')

print()
names = ['Mateo', 'John', 'Eric', 'Mark', 'Robert']
for i in range(len(names)):
    print(names[i], end = ', ')
```

ในตัวอย่าง เป็นการแสดงผลตัวเลข 1 ถึง 10 ด้วยการใช้คำสั่ง for loop ในการทำงานกับฟังก์ชัน range() นั้น จะเป็นการวนอ่านค่าภายในออบเจ็ค ซึ่งการอ่านค่าในออบเจ็คนั้นแตกต่างจากการอ่านค่าในลิสต์ของตัวเลข ซึ่งจะต้องใช้หน่วยความจำสำหรับตัวเลขทั้งหมด แต่ในกรณีนี้เราใช้หน่วยความจำเพียงพื้นที่ของออบเจ็คเท่านั้น ซึ่งแน่นอนว่าประหยัดหน่วยความจำ และใน loop สุดท้ายเป็นการเข้าถึงข้อมูลภายในลิสต์ด้วย Index ของมัน

```
1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
10, 9, 8, 7, 6, 5, 4, 3, 2, 1,
Mateo, John, Eric, Mark, Robert,
```

นี่เป็นผลลัพธ์การทำงานของโปรแกรม ในการแสดงตัวเลข 1 ถึง 10 และในทางกลับกันแสดงจาก 10 ถึง 1 ด้วย และแสดงผลชื่อจากลิสต์

คำสั่ง break

คำสั่ง break ใช้สำหรับหยุดการทำงานของ loop ในทันทีโดยไม่จำเป็นต้องให้เงื่อนไขเป็น False ก่อน มันมักจะใช้ในกรณีที่เราต้องการสร้างเงื่อนไขให้โปรแกรมออกจาก loop ที่นอกเหนือจากเงื่อนไขของ loop มาดูตัวอย่างการใช้งานคำสั่ง break ในภาษา Python

```
for i in range(1, 11):
    if i == 5:
        break
    print(i, end = ', ')
```

ในตัวอย่าง เป็นโปรแกรมในการแสดงตัวเลขจาก 1 ถึง 10 เราได้สร้างเงื่อนไขด้วยคำสั่ง if ว่าถ้าหากค่าของตัวแปร i นั้นเท่ากับ 5 เราจะเรียกใช้คำสั่ง break นั้นหมายความว่าโปรแกรมจะออกจาก loop ในทันที ถึงแม้ว่าการวนค่าในคำสั่ง for นั้นจะยังไม่เสร็จสิ้น

```
1, 2, 3, 4,
```

นี่ เป็นผลลัพธ์การทำงานของโปรแกรมที่แสดงเพียง 1 ถึง 4 เพราะว่าโปรแกรมเจอคำสั่ง break เมื่อค่าของ i นั้นเท่ากับ 5

คำสั่ง continue

ไม่เหมือนคำสั่ง break คำสั่ง continue ใช้สำหรับข้ามการทำงานของ loop ไปทำงานในรอบใหม่ทันทีโดยไม่สนใจคำสั่งที่เหลือหลังจากนั้น มาดูตัวอย่างการใช้งานคำสั่ง continue เพื่อแสดงผลตัวเลขคู่ในภาษา Python

```
for i in range(1, 11):
    if i % 2 == 1:
        continue
    print(i, end = ', ')
```

ในตัวอย่าง เราใช้คำสั่ง for loop ในการวนจาก 1 ถึง 10 เช่นเดียวกันกับในตัวอย่างที่แล้ว แต่สิ่งที่แตกต่างกันเราสร้างเงื่อนไขให้โปรแกรมข้ามการแสดงผลของเลขคี่ไปจาก i % 2 == 1 คำให้คำสั่งการแสดงผลถูกข้ามการทำงานไปรอบใหม่ในทันที

```
2, 4, 6, 8, 10,
```

นี่เป็นผลลัพธ์การทำงานของโปรแกรมสำหรับการแสดงผลเลขคู่ 1 ถึง 10

จากการใช้คำสั่ง break และ continue กับ loop นั้นถึงแม้ว่าเราจะแสดงตัวอย่างเพียงการใช้กับคำสั่ง for แต่คุณสามารถใช้ได้กับ loop ทุกประเภทแม้กระทั่งกับคำสั่ง while ก็เช่นกัน

คำสั่ง else กับ loop

ในการทำงานกับคำสั่งวนซ้ำนั้น เราอาจจะใช้คำสั่ง else clause เพื่อกำหนดบล็อคการทำงานให้กับ loop ได้ โดยในบล็อคของคำสั่ง else นั้นจะทำงานเมื่อโปรแกรมจบการทำงานโดยสิ้นสุดการอ่านค่าในลิสต์สำหรับ for loop และเมื่อเงื่อนไขเป็นเท็จสำหรับ while loop แต่ต้องไม่จบการทำงานของลูปด้วยคำสั่ง break มาดูตัวอย่างการใช้งาน else clause กับ loop ในภาษา Python

THE RADIO CONTROL AIRPLANE MODELER SPORT ASSOCIATION

```
names = ['Mateo', 'John', 'Eric', 'Mark', 'Robert']

search = 'Mark'
for n in names:
    if search == n:
        print(search + ' is found in list')
        break
else:
    print('Not found!')

search = 'Danny'
for n in names:
    if search == n:
        print(search + ' is found in list')
        break
else:
    print('Not found!')
```

ในตัวอย่าง เป็นโปรแกรมในการค้นหาชื่อภายในลิสต์ ใน loop แรกเป็นการค้นหาชื่อ "Mark"ภายในลิสต์ ถ้าหากพบชื่อดังกล่าวเราแสดงผลว่าพบและเรียกใช้คำสั่ง break เพื่อหยุดการทำงานของ loop นั่นทำให้โปรแกรมจบการทำงานของ loop และไม่ทำงานในบล็อคของคำสั่ง else เพราะว่าโปรแกรมจบการทำงานของ loop ด้วยคำสั่ง break

หลังจากนั้นเป็นการค้นหาชื่อ "Danny" ภายในลิสต์ เพราะว่าโปรแกรมนั้นได้วนอ่านค่าภายในลิสต์จนครบและจบการทำงานของลูป ทำให้โปรแกรมทำงานในบล็อคของคำสั่ง else และแสดงผลว่าไม่พบ

```
Mark is found in list
Not found!
```

นี่เป็นผลลัพธ์การทำงานของโปรแกรมในการใช้คำสั่ง else clause กับ loop ในการค้นหาชื่อภายในลิสต์ ในบทนี้ คุณได้เรียนรู้และการใช้งานคำสั่งวนซ้ำในภาษา Python ด้วยการใช้คำสั่ง while loop และ for loop นอกจากนี้เรายังพูดถึงการใช้งานฟังก์ชัน range() เพื่อสร้างออบเจ็คของตัวเลขเพื่อใช้งานกับคำสั่ง for loop และคุณได้รู้จักกับการใช้งานคำสั่ง break และ continue สำหรับหยุดการทำงานและข้ามการทำงานของลูปไป และการใช้งาน else clause กับลูปเมื่อโปรแกรมไม่ออกจากลูปด้วยคำสั่ง break

ฟังก์ชัน

ในบทนี้ คุณจะได้เรียนรู้เกี่ยวกับฟังก์ชันในภาษา Python เราจะพูดถึงการสร้างและการใช้งานฟังก์ชันในเบื้องต้น และการเรียกใช้งานฟังก์ชันในรูปแบบต่างๆ ที่สำคัญ เช่น Default Argument และ Keyword Augment และมากไปกว่านั้น เราจะแนะนำให้คุณรู้จักกับ built-in ฟังก์ชันในภาษา Python

การสร้างฟังก์ชันในภาษา Python

ฟังก์ชัน (Function) คือส่วนของโค้ดหรือโปรแกรมที่ทำงานเพื่อวัตถุประสงค์บางอย่าง ในภาษา Python คุณสามารถสร้างฟังก์ชันของคุณเองเพื่อให้ทำงานที่ต้องการ ในการเขียนโปรแกรมเรามักจะแยกโค้ดที่มีการทำงานเหมือนๆ กันเป็นฟังก์ชันเอาไว้ และเรียกใช้ฟังก์ชันนั้นซ้ำๆ 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 และเช่นเดียวกับภาษาอื่นๆ ฟังก์ชันอาจจะมีหรือไม่มีการส่งค่ากลับ สำหรับฟังก์ชันที่ไม่มีการ return ค่ากลับนั้น เรามักจะเรียกว่า โพรซีเยอร์ (Procedure) ต่อไปมาดูตัวอย่างการประกาศและใช้งานฟังก์ชันในภาษา Python

THE RADIO CONTROL AIRPLANE MODELER SPORT ASSOCIATION

```
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
```

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

```
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
```

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
```

และฟังก์ชันสุดท้ายคือฟังก์ชัน area() เป็นฟังก์ชันสำหรับหาพื้นที่ของรูปสี่เหลี่ยมด้านขนาน และฟังก์ชันมีพารามิเตอร์สองตัวสำหรับความกว้างและความยาวของสี่เหลี่ยม และฟังก์ชันทำการ return ผลลัพธ์ที่เป็นพื้นที่กลับไปด้วยคำสั่ง return

การเรียกใช้งานฟังก์ชันในภาษา Python

หลังจากเราได้สร้างฟังก์ชันในตัวอย่างก่อนหน้าแล้ว ต่อไปเราจะมาเรียกใช้งานฟังก์ชันเหล่านั้น ในการเรียกใช้ฟังก์ชันนั้นเราจะใช้ชื่อของฟังก์ชันและส่งอาร์กิวเมนต์ให้สอดคล้องกับพารามิเตอร์ที่กำหนดไว้ในฟังก์ชัน ดังนั้นอาร์กิวเมนต์คือค่าที่ส่งเข้าไปในฟังก์ชันตอนใช้งาน ส่วนพารามิเตอร์นั้นคือตัวแปรทีกำหนดไว้ในฟังก์ชันเพื่อรับค่าจากอาร์กิวเมนต์ 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
```

นี่เป็นผลลัพธ์การทำงานของโปรแกรม จากการเรียกใช้งานฟังก์ชันในภาษา Python

String ในภาษา Python

ในบทนี้ คุณจะได้เรียนรู้เกี่ยวกับ String ในภาษา Python เราจะพูดเกี่ยวกับการประกาศและใช้งาน String ในรูปแบบต่างๆ เพราะว่าเนื้อหาเกี่ยวกับ String นั้นมีค่อนข้างมาก ดังนั้นเราจึงได้รวบรวมเนื้อหาทั้งไว้ในบทนี้ และเราจะแนะนำการใช้งาน built-in function ในภาษา Python ที่สำคัญในการจัดการกับข้อมูลประเภท String

การประกาศตัวแปร String

String เป็นลำดับของตัวอักษรหลายตัวเรียงต่อกัน ซึ่งในภาษา Python นั้นการที่จะประกาศ String ค่าของมันจะอยู่ในเครื่องหมาย Double quote หรือ Single quote เท่านั้น มาดูตัวอย่างการประกาศตัวแปรของ String

```
name = "Mateo"
site = 'marcuscode.com'
str1 = "This is my string"
str2 = 'This is my string'
```

ในตัวอย่าง เราได้ประกาศ 4 ตัวแปรของ String คุณจะสังเกตว่าในการกำหนดค่าให้กับตัวแปรนั้น String literal จะถูกภายในเครื่องหมาย Double quote ("") หรือ Single quote ('') เท่านั้น ซึ่งได้ผลการทำงานที่เหมือนกัน ขนาดของ String นั้นจะขึ้นกับจำนวนตัวอักษรภายใน String

อย่างไรก็ตาม ถึงแม้เราจะสามารถใช้ Double quote หรือ Single quote กับ String ได้ แต่มีสิ่งที่แตกต่างกันเล็กน้อยสำหรับการใช้งานทั้งสองแบบคือการใช้ตัวอักษรพิเศษใน String หรือเรียกว่า Escape character ลองมาดูตัวอย่างต่อไปนี้

```
sentent1 = "What's your name?"
sentent2 = 'I\'m Mateo.'
sentent3 = "He said \"I would learn Python first\"."
sentent4 = 'His teach replied "Oh well!"'
print(sentent1)
print(sentent2)
print(sentent3)
print(sentent4)
```

ในตัวอย่าง เป็นสิ่งที่แตกต่างของการประกาศ String ทั้งสองแบบกับ Escape character ซึ่ง ตัวอักษร ' และ " นั้นเป็น Escape character ดังนั้นในการใช้งานตัวอักษรเหล่านี้ เราต้องใส่เครื่องหมาย \ ลงไปข้างหน้าเสมอ แต่ในภาษา Python เมื่อคุณใช้ Double quote ในการประกาศ String คุณไม่ต้องทำการ Escape character สำหรับ Single quote และในทางกลับกัน

```
What's your name?
I'm Mateo.
He said "I would learn Python first".
His teach replied "Oh well!"
```

นี่เป็นผลลัพธ์การทำงานของโปรแกรมที่แสดงให้เห็นถึงความแตกต่างของการใช้ Double quote และ Single quote

```
raw_str = r"Python\tJava\tPHP\n"
print(raw_str)
```

ในภาษา Python มีอีกวิธีหนึ่งในการประกาศตัวแปร String คือการกำหนดค่าแบบ raw string ในตัวอย่างข้างบน เราใช้ตัวอักษร r ก่อนหน้าเครื่องหมาย ' หรือ " ซึ่ง raw string เป็นการแปลง String ให้เหมือนกับที่กำหนดใน String literal เสมอ

นอกจากนี้ เรายังสามารถประกาศ String แบบหลายบรรทัดได้ โดยการครอบด้วยเครื่องหมาย """ หรือ ''' String literal ที่ปรากฏภายในเครื่องหมายนี้จะเป็น raw string ที่จะทำการ Escape ให้อัตโนมัติ มาดูตัวอย่าง

```
str = """\
HTTP response code
    200         Success
    404         Not found
    503         Service unavailable
"""

print(str)
```

ในตัวอย่าง เป็นการประกาศตัวแปร String แบบการใช้หลายบรรทัด String literal ทั้งหมดนั้นจะต้องอยู่ภายใน """...""" เนื่องจากเราต้องการจัดรูปแบบของ String ให้สวยงาม ดังนั้นเราจึงต้อง Newline ลงมาเขียนในบรรทัดถัดมา ดังนั้นเราสามารถใช้เครื่องหมาย \ เพื่อลบ Newline ออกไปได้ นี่เป็นผลลัพธ์ของโปรแกรม

```
HTTP response code
    200         Success
    404         Not found
    503         Service unavailable
```

String concatenation

เช่นเดียวกับในภาษาอื่นๆ ในภาษา Python คุณสามารถเชื่อมต่อตั้งแต่สอง String เข้าด้วยกันได้ โดยการใช้เครื่องหมายบวก (+) ถึงแม้ว่านี่เป็นตัวดำเนินการทางคณิตศาสตร์ที่ใช้สำหรับบวกค่าของตัวเลข แต่เมื่อนำมาใช้กับ String จะเป็นการเชื่อมต่อ String เข้าด้วยกันแทนหรือเรียกว่า *Operator overloading* มาดูตัวอย่างการเชื่อมต่อ String

THE RADIO CONTROL AIRPLANE MODELER SPORT

```
first_name = 'Matt'
last_name = 'Williams'
full_name = first_name + ' ' + last_name
bless = 'Merry' + 'Christmas!'

print(full_name)
print(bless)
```

ในตัวอย่าง เป็นการเชื่อมต่อ String ในภาษา Python เรามีตัวแปร first_name สำหรับเก็บชื่อ และตัวแปร last_name สำหรับเก็บนามสกุล และเราสร้างตัวแปร full_name เพื่อเก็บชื่อแบบเต็มโดยการนำ String จาก

ทั้งสองตัวแปรก่อนหน้ามาเชื่อมต่อกัน ซึ่งเป็นการเชื่อมต่อ String จากตัวแปร และในตัวแปร bless เป็นการเชื่อมต่อจาก String literal โดยตรงเพื่อสร้างคำอวยพรในวันคริสมาสต์ นี่เป็นผลลัพธ์การทำงานของโปรแกรม

```
Matt Williams
MerryChristmas!
```

ในภาษา Python คุณยังสามารถเชื่อมต่อ String ได้ด้วยวิธีอื่นอีก เช่น White-space Tab หรือ Newline โดย String literal ที่คั่นด้วยช่องว่างในบรรทัดเดียวกันจะถูกนำมาต่อกันอัตโนมัติโดยตัวแปรของภาษา และคุณสามารถเชื่อมต่อโดยที่ String อยู่คนละบรรทัดได้ด้วยการใช้ \ต่อท้ายสำหรับบอกว่าคำสั่งมีอยู่ในบรรทัดต่อไป มาดูตัวอย่าง

```
name = 'Matt' 'Williams'
my_number = 'One '\
'Two '\
'Three '

print(name)
print(my_number)
```

ในตัวอย่าง เป็นการเชื่อมต่อ String โดยการใช้ช่องว่างและการขึ้นบรรทัดใหม่ และในการเขียนโปรแกรม เราแนะนำให้คุณใช้วิธีการใช้เครื่องหมาย + เพื่อให้การเขียนโปรแกรมเป็นไปในรูปแบบเดียวกัน อย่างไรก็ตาม คุณสามารถเลือกใช้แบบต่างๆ ได้ตามความเหมาะสมของโปรแกรม

```
MattWilliams
One Two Three
```

นี่เป็นผลลัพธ์การทำงานของโปรแกรมในการเชื่อมต่อ String เข้าด้วยกัน

Charterers of string

เนื่องจาก String นั้นเกิดจากตัวอักษรหลายๆ ตัวต่อกันจนเกิดเป็น String หรือกล่าวอีกนัยหนึ่งก็คือ String คืออาเรย์ของตัวอักษร ดังนั้นเราจึงสามารถเข้าถึงตัวอักษรตำแหน่งต่างๆ ของ String ได้ผ่านทาง Index ของมัน เช่นเดียวกับการเข้าถึงข้อมูลใน List มาดูตัวอย่าง

```
s = 'Mountain'
print(s[0]) # M
print(s[4]) # t
print(s[7]) # n
```

ในตัวอย่าง เป็นการเข้าถึงตำแหน่งของตัวอักษรในตัวแปร String s เราสามารถเข้าถึงด้วยการใช้ Index ในเครื่องหมาย [] โดยที่ Index ของ String นั้นจะเริ่มจาก 0 สำหรับตำแหน่งแรก และเพิ่มขึ้นทีละ 1 ไปจนถึงตำแหน่งสุดท้าย

```
s = 'Mountain'
s[0] = 'a' # invalid
```

อย่างไรก็ตาม การทำงานกับ String ผ่านทาง Index นั้นสามารถอ่านค่าได้เพียงอย่างเดียว และไม่สามารถแก้ไขค่าได้ จากตัวอย่างข้างบนจึงทำให้เกิดข้อผิดพลาดขึ้น เพราะเราพยายามที่จะเปลี่ยนแปลงค่าของ String ผ่านทาง Index ของมัน

```
s = 'Python'
for c in s:
    print(c)

print()

s = 'marcuscode.com'
for i in range(len(s)):
    print('s[%d] = %c' % (i, s[i]))

print()
```

เนื่องจาก String นั้นเป็นอาเรย์ของตัวอักษร ดังนั้นมันจึงสะดวกที่จะใช้คำสั่ง For loop วนอ่านค่าจากใน String ทีละตัวอักษร ในตัวอย่างข้างบน เป็นการวนอ่านค่าโดยการใช้คำสั่ง For วนอ่านค่าใน String สำหรับลูปแรก และในลูปที่สอง เราใช้คำสั่ง For สร้าง Index ตั้งแต่ 0 ถึงตำแหน่งสุดท้ายของ String และวนอ่านค่า

THE RADIO CONTROL AIRPLANE MODELER SPORT ASSOCIATION

```
P
y
t
h
o
n

s[0] = m
s[1] = a
s[2] = r
s[3] = c
s[4] = u
s[5] = s
s[6] = c
s[7] = o
s[8] = d
s[9] = e
s[10] = .
s[11] = c
s[12] = o
s[13] = m
```

นี่เป็นผลลัพธ์การทำงานของโปรแกรม

นอกจากนี้ ในภาษา Python นั้นยังสนับสนุนการตัดคำใน String ด้วย Index โดยการตัดเอาส่วนย่อยๆ ภายใน String มาดูตัวอย่างการใช้งาน การตัดคำใน String

```
s1 = 'Mountain'
print(s1[0:4]) # Moun
print(s1[4:6]) # ta
print(s1[0:1]) # M

s2 = 'marcuscode'
print(s2[:6]) #marcus
print(s2[6:]) #code
```

ในตัวอย่าง เป็นการตัดคำด้วย Index ในการใช้งานนั้นจะมีรูปแบบเป็น [start:end] โดยที่ start นั้นเป็นตำแหน่งของ Index เริ่มต้นที่ต้องการ และ end นั้นเป็นตำแหน่งก่อนหน้าตำแหน่งสุดท้ายของตัวอักษรที่ต้องการ เหมือนในตัวอย่างของการตัดคำในตัวแปร s1

และถ้าหากคุณปล่อยให้ตำแหน่ง start ให้ว่าง นั้นหมายความว่าเป็นการเริ่มมาจากตำแหน่งแรกของ String หรือหากคุณปล่อยให้ end ให้ว่าง เป็นการตัดจากตำแหน่งที่กำหนดไปจนสิ้นสุด String เหมือนในตัวแปร s2

```
Moun
ta
M
marcus
code
```

นี่เป็นผลลัพธ์การทำงานของของโปรแกรมในการตัด String ด้วย Index

และนอกจากนี้ เรายังสามารถหาความยาวของ String โดยการใช้ฟังก์ชันที่มากับภาษา Python คือ ฟังก์ชัน len() สำหรับหาขนาดของตัวอักษรของ String ที่ระบุ ซึ่งจะได้ผลลัพธ์กลับมาเป็น Integer นี่เป็นตัวอย่างการหาความยาวของ String

```
s1 = 'Mountain'
s2 = 'marcuscode'
s3 = 'Python'

print('length of s1 =', len(s1))
print('length of s2 =', len(s2))
print('length of s3 =', len(s3))
```

ในตัวอย่าง เป็นการใช้ฟังก์ชัน len() สำหรับหาความยาวของตัวแปร String สามตัวแปร และแสดงผลความยาวนั้นออกทาง console ซึ่งจะได้ผลลัพธ์ดังข้างล่าง

```
length of s1 = 8
length of s2 = 10
length of s3 = 6
```

ในบทนี้ คุณได้เรียนรู้เกี่ยวกับข้อมูลประเภท String ในภาษา Python การประกาศและใช้งานตัวแปรในรูปแบบต่างๆ รวมถึงการเชื่อมต่อ String เข้าด้วยกัน นอกจากนี้ คุณยังทราบว่าเราสามารถเข้าถึงตัวอักษรใน String ผ่านทาง Index ของมันและเราพูดถึงฟังก์ชันในการหาความยาวของ String ในบทต่อไป จะเป็นการใช้งานฟังก์ชันมาตรฐานในภาษา Python เพื่อจัดการกับ String

Python: จัดการกับข้อผิดพลาด

ในการเขียนโปรแกรมPythonอาจพบการแจ้งเตือนข้อผิดพลาดของการทำงานเช่น

```
IndentationError: unexpected indent
SyntaxError: invalid syntax
```

พวกนี้ เป็นความผิดพลาดจากการพิมพ์ของเราเอง แก้ไขได้โดยการไปฝึก code เยอะๆ ซะ จะได้พิมพ์ผิดน้อยลง

แต่ก็ยังมี error อีกประเภทที่อาจเกิดขึ้นได้แม้ว่าเราจะตรวจ syntax ว่าไม่มีที่ผิดแล้ว เช่น

```
a = int(input('dividend: '))
b = int(input('divisor: '))
print(a/b)
```

ถ้าผู้ใช้ป้อนค่า 0 เข้ามาในครั้งที่สอง (ตัวแปร b) จะทำให้เกิด

```
ZeroDivisionError: int division or modulo by zero
```

เพื่อแก้ไขให้ code นี้ใช้งานได้ เราอาจดักเช็คค่าของ b ง่ายๆ เช่น

```
while True:
    a = int(input('dividend: '))
    b = int(input('divisor: '))
    if b != 0:
        break

print(a/b)
```

แต่ทั้งนี้ผู้ใช้ยังสามารถป้อนตัวอักษรเข้ามาได้อีก (ซึ่งทำให้โปรแกรมไม่สามารถคำนวณได้) แม้ว่าเราอาจดักไม่ให้ผู้ใช้สามารถกดแป้นตัวอักษรได้ แต่ก็จะทำให้ code หนักขึ้น (มี maintainability ลดลง) เราจึงควรเลือกใช้ exception แทนดังตัวอย่างด้านล่าง

THE RADIO CONTROL AIRPLANE MODELER SPORT ASSOCIATION

```
while True:
    try:
        a = int(input('dividend: '))
        b = int(input('divisor: '))
        print('division is: ', a/b)
        print('division success, now leave program.')
        break
    except ValueError:
        print("please input a number, try again.")
    except ZeroDivisionError:
        print("divisor can't be zero, try again.")
```

หรือ เพื่อให้ code สวยงามยิ่งขึ้น

```
while True:
    try:
        # put code that can lead exception inside try.
        a = int(input('dividend: '))
        b = int(input('divisor: '))
        print('division is: ', a/b)
    except ValueError:
        print("please input a number, try again.")
    except ZeroDivisionError:
        print("divisor can't be zero, try again.")
    else:
        # put code that must be done when no exception.
        print('division success, now leave program.')
        break
    finally:
        # always done whether there are exception or not!
        print('--------------------')
```

สำหรับ exception ที่ควรรู้จัก คือ

```
Exception     # prototype

IOError       # no file to open
EOFError      # no char in file left for read

NameError     # variable does not exist
ValueError    # operation on inappropriate value
TypeError     # operation on inappropriate type

RuntimeError # not in other categories
```

R.C.S.A.
THE RADIO CONTROL AIRPLANE MODELER SPORT
ASSOCIATION