



Week 3 : Python

Conditional Statements

Natakorn Pramayan, Ph.D.

If – Else Statement

If – Else เป็นคำสั่ง if , else , elif เป็นคำสั่งที่ใช้ในการตั้งเงื่อนไขให้กับโปรแกรม เมื่อเงื่อนไขเป็นจริง (True) โปรแกรมจะทำงานตามคำสั่งที่กำหนดไว้ในขอบเขตของ if else statements (Block of Code) ในภาษาโปรแกรมส่วนใหญ่ ใช้เครื่องหมายวงเล็บปีกกา { } ในการกำหนดขอบเขตของ if else statements ตัวอย่าง if statement ภาษา PHP

```
if (isset($_POST['page'])) {  
    $page = $_POST['page'];  
} else {  
    $page = 1;  
}
```

```
if (isset($_POST['page'])) {  
    $page = $_POST['page'];  
} else {  
    $page = 1;  
}
```

การใช้คำสั่ง if else statement จะต้องกำหนดเงื่อนไขให้กับคำสั่ง if เสมอ
ในกรณีตัวอย่างข้างต้นเงื่อนไข คือ `isset($_POST['page'])`

ถ้าเงื่อนไขเป็นจริง จะทำงานตามคำสั่งที่อยู่ภายใต้ { } ของ if
นั่นคือ `$page = $_POST['page'];`

ถ้าเงื่อนไขเป็นเท็จ จะทำงานใน { } ของ else แทน

สำหรับภาษา Python ใช้การย่อหน้าแทนการใช้วงเล็บปีกกาในการบอกขอบเขต หรือ Block of code ตัวอย่างการใช้งาน if else statement ใน Python

ตัวอย่าง

ตัวแปร เงื่อนไข

```
3  Me = 24
4  myFriend = 30
5  if Me > myFriend: # if True print I am older than you / if False print No. It's me
6      print("I am older than you !!")
7  else :
8      print("No. It's Me")
```

No. It's Me

แสดงผลออกทางหน้าจอ

ภายใน block ของ if else statement สามารถมีคำสั่งได้ไม่จำกัด

ตัวอย่าง

```
3 Me = 24
4 myFriend = 30
5 if Me < myFriend :
6     print("I am older than you !!")
7     print("Call me brother !!")
8 else :
9     print("No. It's Me")
```

If -> Statement 1

If -> Statement 2

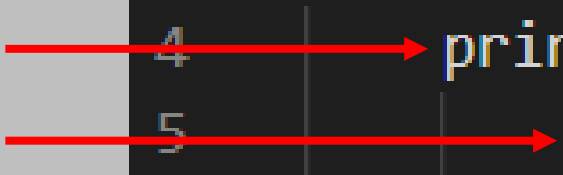
```
I am older than you !!
Call me brother !!
```

แสดงผลออกทางหน้าจอ

ข้อควรระวัง การย่อหน้าควรมีระยะที่เท่ากัน


ตัวอย่างที่ผิด

```
2  
3  if Me > myFriend :  
4      print("I am older than you !!")  
5      print("Call me brother !!")  
6
```



ตัวอย่างที่ถูกต้อง

```
2  
3  if Me > myFriend :  
4      print("I am older than you !!")  
5      print("Call me brother !!")  
6
```



ข้อสังเกต Statement ของ if และ else สามารถมีย่อหน้าที่ไม่ตรงกันได้ ไม่เกิด Error เนื่องจากไม่ใช่ Block เดียวกัน

ตัวอย่างที่ถูก

```
2
3  if Me > myFriend :
4      |   print("I am older than you !!")
5      |   print("Call me brother !!")
6  else :
7      |   |   print("No. It's Me")
```


การสร้างเงื่อนไขมากกว่า 2 เงื่อนไข

การใช้คำสั่ง if และ else เป็นการสร้างเงื่อนไขเพียง 2 เงื่อนไขเท่านั้น เมื่อเงื่อนไขใน if ไม่เป็นจริง จะทำเงื่อนไขใน else แทนที่ เมื่อนำไปใช้งานจริง อาจไม่เพียงพอต่อความต้องการ ภาษาคอมพิวเตอร์จึงมีคำสั่งที่ใช้สร้างเงื่อนไขเพิ่มเติม สำหรับ Python ใช้คำสั่ง elif ในการเพิ่มเงื่อนไข

ตัวอย่าง

```
3  Me = 24
4  myFriend = 30
5  if Me > myFriend : # Condition 1
6      print("I am older than you !!") # If -> Statement 1
7      print("Call me brother !!") # If -> Statement 2
8  elif Me == myFriend : # Condition 2
9      print("We are friend") # Elif -> Statement 1
10 else : # Condition 3
11     print("No. It's Me") # Else -> Statement 1
```

คำสั่ง elif และ else ใช้งานได้ก็ต่อเมื่อมีการใช้คำสั่ง if เท่านั้น

ตัวอย่าง

```
3  Me = 24
4  elif Me > 5 : # Invalid !!
5  | print("I am older than you !!")
6
```

การทำงานไม่ถูกต้องเนื่องจากยังไม่มีการใช้คำสั่ง if

ข้อสังเกต เมื่อเงื่อนไขใด ๆ เป็นจริงแล้ว จะทำงานในเงื่อนไขนั้นและออกจาก if else statementทันที ไม่มีการตรวจสอบเงื่อนไขเพิ่มเติม

ตัวอย่าง

```
3  x = 15
4  if x % 3 == 0 : # ถ้า x หาร 3 ลงตัว
5  |   print("Fizz")
6  elif x % 5 == 0 : # ถ้า x หาร 5 ลงตัว
7  |   print("Buzz")
8  elif x % 15 == 0 : # ถ้า x หาร 15 ลงตัว
9  |   print("Fizz - Buzz")
```

เมื่อพิจารณาเงื่อนไขทั้ง 3 พบว่า
เป็นจริงทั้งหมด แต่เมื่อลอง Run
โปรแกรมจะได้ผลลัพธ์เป็น Fizz
เพียงเท่านั้น ไม่มีการตรวจสอบ
เงื่อนไขเพิ่มเติม

Fizz


แสดงผลออกทางหน้าจอ

การใช้งาน if else แบบย่อ

การเขียน if else ในรูปแบบย่อ สามารถทำได้ก็ต่อเมื่อ statement ของ if และ else มีเพียง 1 statement เท่านั้น

ตัวอย่าง

```
3  Me = 24
4  myFriend = 30
5  print("I am older than you") if Me > myFriend else print ("No !!")
6
```



If -> Statement 1

Else -> Statement 1

การสร้างเงื่อนไขมากกว่า 2 เงื่อนไข ใน 1 บรรทัด (ไม่สามารถเขียน elif ในรูปแบบย่อได้)

ตัวอย่าง

```
3 x,y = 24,30
4 print("Older") if x > y else print("Equal") if x == y else print ("No !!")
5
```

เงื่อนไขที่ 1

เงื่อนไขที่ 2

Nested If

เป็นการใช้งาน if ภายใต้อีก if

ตัวอย่าง

```
2
3  x = 24
4  if x >= 7 :
5      if x <= 12 :
6          print("Elementary School")
7      elif x <= 18 :
8          print("High School")
9      else :
10         print("Older")
11 else :
12     print("Pre-School")
13
```

→ เงื่อนไข if ที่ทำการตรวจสอบเป็นขั้นแรก

→ เงื่อนไข if ภายใต้อีก if

แบบฝึกหัดที่ 3.1

คำชี้แจง ให้นักศึกษาเขียนโปรแกรมรับค่าระยะทาง (กิโลเมตร-km) เพื่อคิดค่าบริการระยะทางโดยมีเงื่อนไขดังนี้

แบบจ่ายเพิ่ม	
ค่าแรกเข้า	25 บาท
ระยะทาง 25 กิโลเมตรขึ้นไป (จ่ายเพิ่ม)	55 บาท
แบบเหมาจ่าย	
ถ้าระยะทางไม่เกิน 25 กิโลเมตร	25 บาท
ระยะทาง 25 กิโลเมตรขึ้นไป	55 บาท

เลือกเมนูเพื่อทำรายการ

#####

กด 1 เลือกเหมาจ่าย

กด 2 เลือกจ่ายเพิ่ม

1

กรุณกรอกระยะทาง

20

ค่าใช้จ่าย รวมทั้งหมด 25 บาท

Python Loops

ในการเขียนโปรแกรมคอมพิวเตอร์ Loops เป็นลำดับของคำสั่ง
ที่ทำงานวนซ้ำตามเงื่อนไขที่กำหนด ในภาษา Python Loops แบ่งออกเป็น
2 ชนิด คือ While loop และ For loop

While loop

Loop ชนิดนี้ จะทำงานคำสั่งที่อยู่ภายใต้ Block ไปเรื่อย ๆ จนกว่าเงื่อนไขที่กำหนดจะเป็นเท็จ

ตัวอย่าง

การใช้งานคำสั่ง loops ต้องกำหนด Block
เช่นเดียวกับการใช้งานคำสั่ง if else statement
จากตัวอย่าง True ที่อยู่ในวงเล็บหลังคำสั่ง
while คือเงื่อนไขที่กำหนด ผลลัพธ์ของตัวอย่าง
คือการแสดงข้อความ I love you วนซ้ำเรื่อย ๆ
ตราบใดที่เงื่อนไขไม่เปลี่ยนแปลงเป็นเท็จ

[illegible]

สามารถกำหนดจำนวนรอบในการวนซ้ำได้ โดยใช้เงื่อนไขภายในวงเล็บของคำสั่ง while

ตัวอย่าง

```
3 i = 1
4 z = 2
5 while(i < 13) :
6     print(str(z) + ' x ' + str(i) + ' = ' + str(z*i))
7     i += 1
```

กำหนดให้ while ทำงานทั้งหมด 12 รอบ
จากเงื่อนไข $i < 13$ เมื่อครบทั้งหมด 12 รอบ
ค่า $i = 13$ ทำให้เงื่อนไขของ while เป็นเท็จ
จบการทำงาน แสดงผลออกทางหน้าจอ

```
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20
2 x 11 = 22
2 x 12 = 24
```

แบบฝึกหัดที่ 3.2

คำชี้แจง ให้นักศึกษาเขียนโปรแกรมวนรับค่าตัวเลขมา n จำนวนโดยใช้คำสั่ง While Loop แล้วหาผลรวมของค่าที่รับมา

```
กรุณกรอกจำนวนครั้งการรับค่า
3
กรอกตัวเลข      5
กรอกตัวเลข      6
กรอกตัวเลข      5
ผลรวมค่าที่รับมาทั้งหมด = 16
```

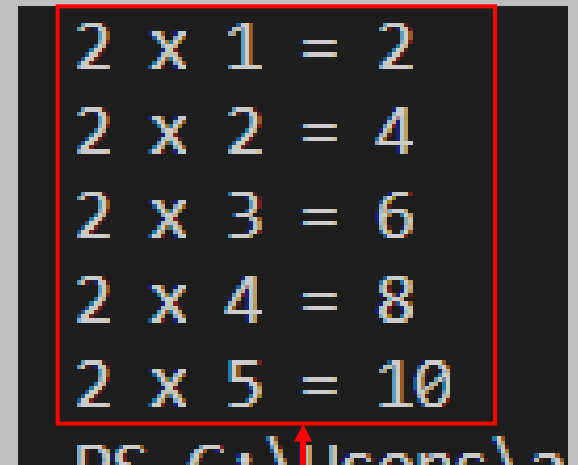
Python Control Statements

คำสั่ง **break** และ **continue** เป็นคำสั่งที่ใช้ในการหยุดการทำงานของ Loops
ซึ่ง break และ continue มีลักษณะการทำงานที่แตกต่างกัน

คำสั่ง break

ตัวอย่าง

```
3 i = 1
4 z = 2
5 while(i < 13) :
6     print(str(z) + ' x ' + str(i) + ' = ' + str(z*i))
7     if i == 5 :
8         break
9     i += 1
10
```




```
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
PS C:\Users\...
```

เมื่อ i มีค่าเป็น 5 คำสั่ง break จะทำงาน ทำให้ while loop
หยุดการทำงานทันที แม้ว่าเงื่อนไขยังเป็นจริงอยู่

ผลลัพธ์ที่แสดงออกทางหน้าจอ

ตัวอย่าง

```
2  
3 i = 1  
4 while(i < 13) :  
5     print(i)  
6     break
```



1

จากตัวอย่าง ผลลัพธ์ที่ได้ คือ 1 เนื่องจากคำสั่ง break ถูกสั่งให้ทำงาน
หลังจากคำสั่ง print(i) แตกต่างจากตัวอย่างแรก ที่มีการตั้งเงื่อนไขให้ใช้คำสั่ง
break เมื่อ $i == 5$

คำสั่ง Continue

ตัวอย่าง

```
3 i = 0
4 z = 2
5 while(i < 12) :
6     i += 1
7     if i == 5 :
8         continue
9     print(str(z) + ' x ' + str(i) + ' = ' + str(z*i))
10
```

คำสั่ง continue ที่ i == 5

```
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20
2 x 11 = 22
2 x 12 = 24
```

จากผลลัพธ์ พบว่าเมื่อค่า $i == 5$ (เมื่อ i มีค่าเป็น 5) คำสั่ง continue จะทำงาน
ส่งผลให้ while loop หยุดการทำงานในรอบนั้น ทำให้ $2 \times 5 = 10$ ไม่ถูกแสดงบนหน้าจอ

แบบฝึกหัดที่ 3.3

คำชี้แจง ให้นักศึกษาเขียนโปรแกรมวนรับชื่ออาหารไปเรื่อยๆ จนกว่าจะพิมพ์คำว่า exit
หลังจากพิมพ์คำว่า exit แล้วให้แสดงผล ดังนี้

```
ป้อนชื่ออาหารสุดโปรดของคุณ หรือ exit เพื่อออกจากโปรแกรม
อาหารโปรดอันดับที่ 1 คือ          ข้าวผัด
อาหารโปรดอันดับที่ 2 คือ          ข้าวต้ม
อาหารโปรดอันดับที่ 3 คือ          มาม่า
อาหารโปรดอันดับที่ 4 คือ          exit
อาหารสุดโปรดของคุณมีดังนี้ 1. ข้าวผัด 2. ข้าวต้ม 3. มาม่า
```

For loop

ในภาษา Python คำสั่ง For loop จะมีความแตกต่างจาก For loop ของภาษาโปรแกรมอื่น ๆ ส่วนใหญ่ใช้สำหรับการวนซ้ำเพื่อกระทำกับ Collection หรือ Object เช่น List Tuple Set หรือ Dictionary หรือการทำงานวนซ้ำในจำนวนรอบที่แน่นอน

ในการใช้คำสั่ง For loop ไม่จำเป็นต้องกำหนดเงื่อนไขให้ For เหมือนกับ While แต่การใช้งาน For loop จะทำงานร่วมกับ Object

ตัวอย่าง

```
3 names = ['A', 'B', 'C', 'D'] # create object (list)
4 for x in names : # use for with object. Assign each value to x
5     print(x)
```

ใช้คำสั่ง for วนซ้ำ print ค่าของ list ออกทางหน้าจอ

A
B
C
D

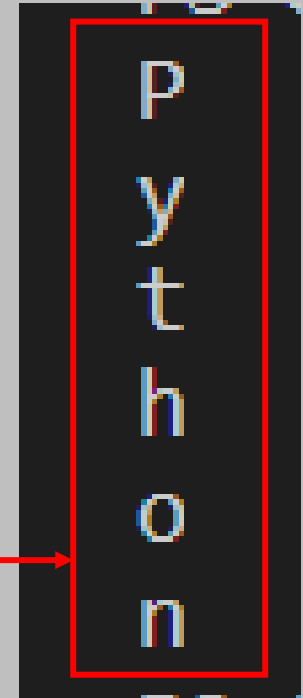
ผลลัพธ์ที่แสดงออกทางหน้าจอ

สามารถใช้ For loop วนซ้ำ String (ในการเขียนโปรแกรมแบบ OOP มองทุกอย่างเป็น Object)

ตัวอย่าง

```
2  
3 names = 'Python' # create object (string)  
4 for x in names : # use for with object.  
5     | print(x)
```

ผลลัพธ์ที่แสดงออกทางหน้าจอ



P
y
t
h
o
n

การกำหนดจำนวนรอบให้ For loop ทำได้โดยการใช้ Function range() เพื่อสร้าง object ของตัวเลข

ตัวอย่าง

```
3  for x in range(5): # use for with object (range).  
4      print(x)  
5
```

จากตัวอย่าง ผลลัพธ์ที่ได้ คือ 0 ถึง 4 นั้นหมายความว่า range(5) เป็นการสร้าง object ตัวเลขตั้งแต่ 0 ถึง 5 (กรณีที่ไม่ส่ง argument ให้กับ Function range)

0
1
2
3
4

Function `range()` มี parameter ที่สามารถควบคุมเพิ่มเติมได้ คือ

1. ค่าเริ่มต้น (ทางเลือก default 0)
2. ค่าสิ้นสุด (บังคับ) และ
3. จำนวนที่เปลี่ยนแปลงต่อรอบ (ทางเลือก default 1)

จากตัวอย่างที่ผ่านมา เป็นการใช้ `range(5)` กำหนดตัวเลขเพียง 1 ตัวให้กับ `range` จะเป็นการสร้าง object ตัวเลข 0 ถึง 4 แต่หากต้องการกำหนดค่าเริ่มต้น และค่าสิ้นสุด ด้วยตัวเอง สามารถทำได้ ดังนี้ **`range (start , stop , step)`**

ตัวอย่าง


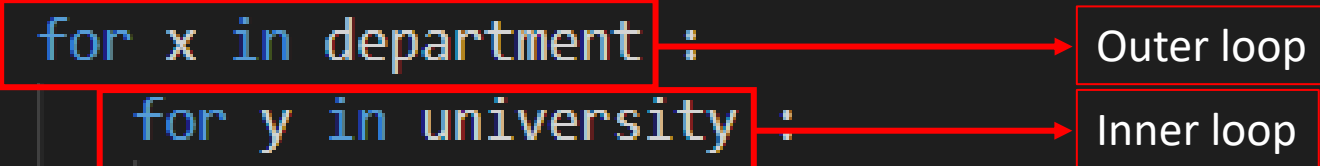
```
2
3  a = list(range(10)) # [0,1,2,3,4,5,6,7,8,9]
4  b = list(range(5,11)) # [5,6,7,8,9,10]
5  c = list(range(0,10,2)) # [0,2,4,6,8]
6  d = list(range(0,-10,-2)) # [0,-2,-4,-6,-8]
7
```

จากตัวอย่าง เป็นการสร้าง list โดยใช้ Function range() ในรูปแบบต่าง ๆ เป็นค่าเริ่มต้นของ list a b c และ d ในส่วนของ a เป็นการใช้ range(10) สร้าง object ตัวเลข 0 ถึง 9 b สร้าง object 5 ถึง 10 c สร้าง object 0 ถึง 10 แต่เพิ่มค่าทีละ 2 และ d สร้าง object 0 ถึง -10 ลดค่าทีละ -2

อีกหนึ่งวิธีในการใช้ For loop ในภาษา Python คือการใช้แบบ Nested loops เป็นการเขียน loop ซ้อนเข้าไปใน loop

ตัวอย่าง

```
3 department = ['ComED', 'SciED', 'MathED']
4 university = ['KKU', 'CMU', 'CU']
5 for x in department:
6     for y in university:
7         print(x + " " + y)
8
```



```
ComED KKU
ComED CMU
ComED CU
SciED KKU
SciED CMU
SciED CU
MathED KKU
MathED CMU
MathED CU
```

ผลลัพธ์ที่แสดงออกทางหน้าจอ Outer loop ทำงาน 1 ครั้ง
แล้ว Inner loop ทำงานจนเสร็จสิ้น จากนั้น Outer loop
จะทำงานครั้งต่อไปเรื่อย ๆ จนเสร็จสิ้น

แบบฝึกหัดที่ 3.4

คำชี้แจง ให้นักศึกษาเขียนโปรแกรมร้านค้าโดยมีระบบการทำงานอยู่ 3 ส่วนให้เลือกใช้งาน ดังนี้

เลือก → a ระบบเพิ่มข้อมูลลูกค้า

เลือก → b ระบบแสดงข้อมูลลูกค้า

เลือก → c ปิดโปรแกรม

โดยโปรแกรมจะทำงานไปเรื่อย ๆ จนกว่าจะเลือกคำสั่ง c ปิดโปรแกรม

ระบบเพิ่มข้อมูลลูกค้า

ร้านคุณหลินบิวตี้

เพิ่ม [a]

แสดง [s]

ออกจากระบบ [x]

a

ป้อนรายการลูกค้า (รหัส: ชื่อ: จังหวัด) 123: สมชาย: ขอนแก่น

*****ข้อมูลได้เข้าสู่ระบบแล้ว*****

ระบบแสดงข้อมูลลูกค้า

ร้านคุณหลินบีวดี

เพิ่ม [a]

แสดง [s]

ออกจากระบบ [x]

s

รหัส--ชื่อ-----จังหวัด

123 สมชาย (ขอนแก่น)

ปิดโปรแกรม

ร้านคุณหลินบีวดี

เพิ่ม [a]

แสดง [s]

ออกจากระบบ [x]

x

ต้องการปิดโปรแกรมใช่หรือไม่ y/n: y

จบการทำงาน



Q & A