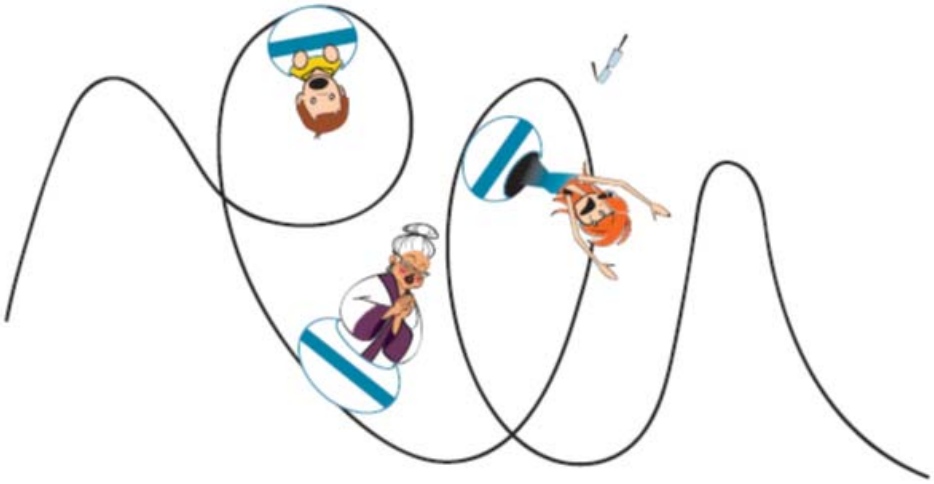
Lecture 8 Loop the Loop
การวนซ้ำ

13 กันยายน 2557 10:18

คนส่วนใหญ่จะทำกิจวัตรหรือการงานที่มีลักษณะซ้ำแล้วซ้ำอีกซึ่งเป็นสิ่งที่น่าเบื่อ ดังนั้นทำไมเราไม่ให้คอมพิวเตอร์ทำงานอย่างนี้แทนเรา เนื่องจากคอมพิวเตอร์ไม่เคยรู้สึกเบื่อและทำงานที่ซ้ำแล้วซ้ำอีกได้ดีกว่า ในบทนี้เราจะศึกษาการทำที่เป็นแบบวนทำซ้ำ repeat

การเขียนโปรแกรมคอมพิวเตอร์มีวิธีการที่จะทำซักแล้วซ้ำอีก เราเรียกว่า การวน looping มีลักษณะการวนอยู่สองรูปแบบดังนี้

- การวนแบบจำนวนที่แน่นอน เรียกว่า การวนแบบนับ counting loops
- การวนแบบตามเงื่อนไขที่เป็นจริง เรียกว่า การรวนแบบเงื่อนไข conditional loops



Counting loop การวนแบบนับ

การวนแบบนับเราจะเรียกว่า **for** loop เพราะว่าภาษาคอมพิวเตอร์ส่วนใหญ่รวมทั้ง Python จะใช้คำว่า for เพื่อสร้างการวน ลองทดลองการใช้ต่อไปนี้ โดยการสร้างไฟล์มาเขียนโปรแกรม File>New แล้วเขียนโค้ดดังนี้

```
for i in [1, 2, 3, 4, 5]:
    print("Hello")
```

บันทึก Loop1.py และรันโปรแกรม Run>Run Module หรือกด F5 จะได้ผลลัพธ์ดังต่อไปนี้

```
Hello
Hello
Hello
Hello
Hello
```

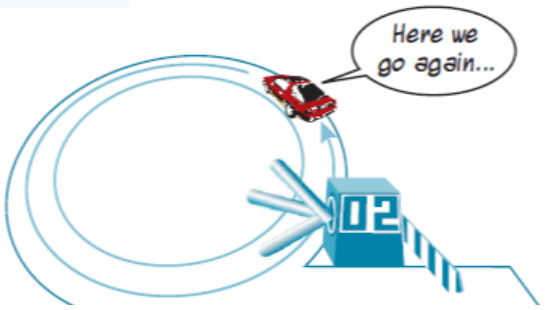
จากโค้ดตัวอย่างโปรแกรมจะวนพิมพ์ 5 ครั้ง ตามจำนวนสิ่งของที่อยู่ใน [ ] เราจะเรียกว่า ลิสต์หรือรายการ list ที่มีจำนวนของหลายสิ่ง เช่นถ้าเราไปซื้อของที่ห้างเราจะเขียนรายการหรือลิสต์ที่เราต้องการซื้อที่มีจำนวนหลายรายการ

```
[1, 2, 3, 4, 5]
```

จะเห็นได้ว่าเรามีเลขอยู่ 5 ตัวใน list ถ้าเราพิมพ์เพื่อดูชนิด

```
>>> type([1,2,3,4,5])
<class 'list'>
```

ซึ่งการวนเราสร้างตัวแปร i จากโค้ด for **i** in ... ตัวแปร i นี้จะไปรับค่าใน list มาทีละตัว ค่าของตัวแปร i จะเป็น 1, 2, 3, 4, 5 ตามลำดับแต่ละรอบ ซึ่งจะมีทั้งหมด 5 รอบถ้าเราพิมพ์ข้อความจะปรากฏข้อความ 5 ข้อความดังผลลัพธ์



1. เริ่มแรกตัวแปร i มีค่าเท่ากับ 1
2. Loop ทำงาน โดย ทำคำสั่งที่อยู่ใน block ใต้ for ทั้งหมด (ในตัวอย่างมีแค่คำสั่งเดียวคือ print("Hello"))
3. เมื่อจบคำสั่งใน block ตัวแปร i รับค่าใหม่
4. เมื่อครบทุกรายการใน list จบการทำงานของ loop

แต่ละรอบที่ทำงานเราเรียกว่า iteration

4. เมื่อครบทุกรายการใน list จบการทำงานของ loop

แต่ละรอบที่ทำงานเราเรียกว่า iteration

ลองทดสอบ print ค่าตัวแปร i

```
for i in [1, 2, 3, 4, 5]:
    print(i)
```

ผลลัพธ์

```
1
2
3
4
5
```

Using a counting loop การใช้งานการวนแบบนับ

ถ้าเราทำตารางสูตรคูณ มีโค้ดดังนี้

```
for i in [1, 2, 3, 4, 5]:
    print(i, "times  8 =", i*8)
```

ผลลัพธ์

```
1 times  8 = 8
2 times  8 = 16
3 times  8 = 24
4 times  8 = 32
5 times  8 = 40
```

ถ้าเราไม่ใช้ loop เราอาจจะต้องพิมพ์ข้อความอย่างนี้ จะได้ผลดังนี้

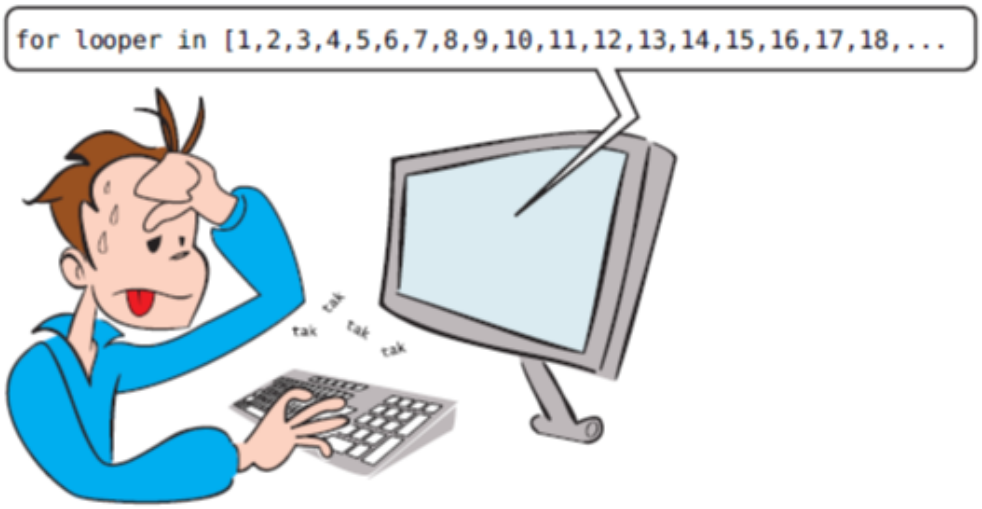
```
print("1 times  8 =", 1*8)
print("2 times  8 =", 2*8)
print("3 times  8 =", 3*8)
print("4 times  8 =", 4*8)
print("5 times  8 =", 5*8)
```

A shortcut - range() การทำให้สั้นโดยใช้คำสั่ง range()

ตอนนี้เราวนแค่ 5 ค่าเราใช้

```
for i in [1, 2, 3, 4, 5]:
    print(i, "times  8 =", i*8)
```

คำถามคือถ้าเราต้องการวนร้อยค่า เราจะต้องสร้าง list ร้อยตัวหรือไม่ เป็นงานที่เหนื่อย



โชคดีที่เรามีคำสั่ง range ที่ให้เราสร้าง list ขึ้นมาเองได้ ดังโค้ดด้านร่างนี้

```
for i in range(1,5):
    print(i, "times  8 =", i*8)
```

ผลลัพธ์ที่ได้คือ

```
1 times  8 = 8
2 times  8 = 16
3 times  8 = 24
4 times  8 = 32
```

แล้วค่าที่ 5 หายไปไหน เนื่องจากฟังก์ชัน range() จะไม่รวมตัวสุดท้ายไว้ด้วย ถ้าเราทดลองเปลี่ยนเป็น list เพื่อดูค่า ด้วยค่าสั้ง list() ดังนี้

```
print(list(range(1, 5)))
```

ได้ผลดังนี้

```
[1, 2, 3, 4]
```

ซึ่งมี 4 ค่า ทำอย่างไรจะได้ 5 ค่า หนึ่ง คือการเริ่มต้นที่ 0 หรือการเปลี่ยนขอบบนเป็น 6 เราได้ 5 จำนวน ดังตัวอย่างนี้ **(การเขียนโปรแกรมส่วนมากจะนิยมนับที่ 0)**

```
print(list(range(0, 5)))
```

ได้ผลลัพธ์ดังนี้

```
[0, 1, 2, 3, 4]
```

```
for i in range(0, 5):
    print(i+1, "times  8 =", (i+1)*8)
for i in range(1, 6):
    print(i, "times  8 =", i*8)
```

ผลลัพธ์ที่ได้จาก for สองอันนี้ ได้เหมือนกัน

คราวนี้เราสามารถนับถึง 20 ได้ง่ายดายขึ้น

```
for i in range(1,21):
    print(i, "times  8 =", i*8)
```

ผลลัพธ์

```
1 times  8 = 8
2 times  8 = 16
3 times  8 = 24
4 times  8 = 32
5 times  8 = 40
6 times  8 = 48
7 times  8 = 56
8 times  8 = 64
9 times  8 = 72
10 times  8 = 80
11 times  8 = 88
12 times  8 = 96
13 times  8 = 104
14 times  8 = 112
15 times  8 = 120
16 times  8 = 128
17 times  8 = 136
18 times  8 = 144
19 times  8 = 152
20 times  8 = 160
```

การตั้งชื่อตัวแปรการนับ นิยมใช้ i, j, k ขณะที่ตัวแปรในการคำนวนนิยมใช้ a, b, c , x, y,z

เราสามารถวนพิมพ์ตัวอักษรในข้อความได้

```
for letter in "Hello Bank":
    print(letter)
```

ผลลัพธ์ที่ได้คือ

```
H
e
l
l
o

B
a
n
k
```

Counting by Steps การนับเป็นก้าวกระโดดที่เท่ากัน

ถ้าเราต้องการนับที่ละ 2 หรือครั้งละ 5 หรือนับถอยหลัง กับฟังก์ชัน range เราสามารถทำได้ ดังนี้

นับทีละ 2

```
for i in range(1, 10, 2):
    print(i)
```

แสดงผลดังนี้

```
1
3
5
7
9
```

เราอาจจะนับทีละ 5

```
for i in range(5, 26, 5):
    print(i)
```

แสดงผลดังนี้

```
5
10
15
20
25
```

เราจะนับถอยหลัง

```
for i in range(10, 1, -1):
    print(i)
```

แสดงผลได้ดังนี้

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

ทดสอบเขียนโปรแกรมนับถอยหลังครั้งละหนึ่งวินาที ทำการปล่อยจรวด

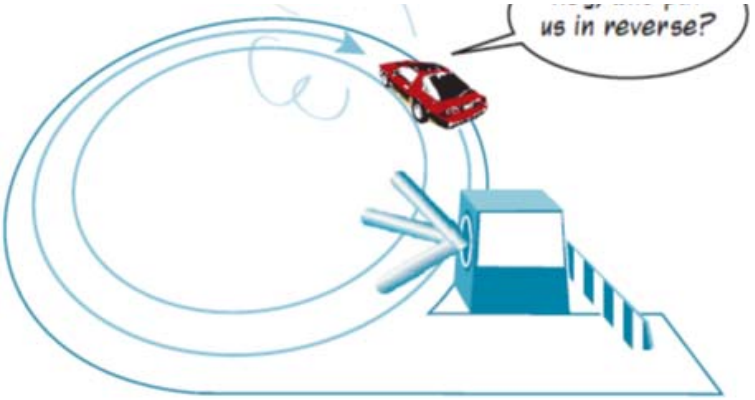
```
import time
for i in range(10, 0, -1):
    print(i)
    time.sleep(1)
print("Go!!")
```

แสดงผลลัพธ์ดังนี้

```
10
```

แสดงผลลัพธ์ดังนี้

```
10
9
8
7
6
5
4
3
2
1
Go!!
```



การใช้โมดูล (module) ช่วยในการหยุดเวลาทีละหนึ่งวินาที โมดูล time และเราเรียกใช้หยุดเวลา time.sleep(1) ค่า 1 ในวงเว็บคือหยุดหนึ่งวินาทีหยุดก่อนจะวนไปทำใหม่

Counting without numbers การนับแบบไม่เป็นตัวเลข
เรารู้ว่า for สามารถที่จะวนค่าใน list หรือข้อความได้ ทั้งนี้ list สามารถเก็บค่านอกจากตัวเลขได้ เราอาจจะเก็บข้อความได้เช่น

```
for color in ["Red", "Green", "Blue", "White", "Black"]:
    print(color)
```

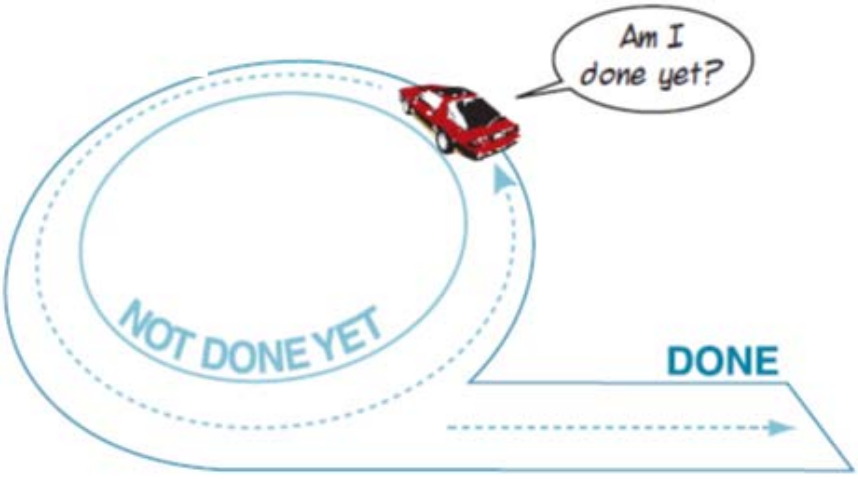
แสดงผลได้ดังนี้

```
Red
Green
Blue
White
Black
```

ตอนนี้เราจะวนทำซ้ำเราจำเป็นจะต้องมี list แต่ในกรณีที่เราไม่มี list อะไรให้วนซ้ำเราสามารถจะทำได้ โดยอีกอย่างหนึ่งคือ while

While การวนแบบเงื่อนไข
การวนแบบที่สองคือการวนแบบมีเงื่อนไข conditional loop โดยใช้คำสั่ง while for เป็นการวนที่เราจำเป็นจะต้องรู้จำนวนรอบที่ชัดเจนหรือมี list ให้วน แต่บางครั้งการวนอาจจะไม่รู้จำนวนรอบที่ชัดเจนแน่นอน การวนอาจจะวนจนกว่าเราจะถึงเงื่อนไขอะไรอย่างใดอย่างหนึ่ง
ในบทที่แล้ว เราได้รู้เรื่องเกี่ยวกับการสร้างเงื่อนไข conditions การทดสอบเงื่อนไข testing โดยใช้คำสั่ง **if** เราจะใช้ในลักษณะเดียวกันด้วยคำสั่ง while ถ้าเงื่อนไขเป็นจริงจะทำภายใต้ block ของ while และจะทำจนกว่าเงื่อนไขจะเป็นเท็จ

คำสั่ง while จะถูกค่อยถามว่า ทำเสร็จหรือไม่ ถ้าไม่เสร็จทำต่อ (เงื่อนไขยังเป็นจริงทำต่อ) ถ้าเสร็จหยุดทำ (ถ้าเงื่อนไขเป็นเท็จหยุดทำ)



ทดสอบการใช้โค้ดต่อไปนี้

```
print("Type 3 to continue, anything else to quit.")
someInput = input()
while someInput == '3':
    print("Thank you for 3")
    print("Type 3 to continue, anything else to quit.")
    someInput = input()
print("That's not 3. so I'm quitting now.")
```

เราเขียนโปรแกรมรับอินพุตเลข 3 ถ้าเรายังป้อนเลข 3 อยู่เราจะโดนวนถามต่อไปเรื่อยๆ จนกว่าเราจะป้อนตัวอื่นที่ไม่ใช่เลข 3
ถ้าเรายังป้อนเลข 3 ค่า someInput == '3' ก้อยังเป็นจริงเมื่อเราป้อนตัวอื่น เงื่อนไขนี้จะเป็นเท็จ การวนก็จะหยุด

```
Type 3 to continue, anything else to quit.
3
Thank you for 3
Type 3 to continue, anything else to quit.
3
Thank you for 3
Type 3 to continue, anything else to quit.
3
Thank you for 3
Type 3 to continue, anything else to quit.
1
That's not 3. so I'm quitting now.
```

การออกจาก loop ระหว่างทาง ด้วยคำสั่ง break และ continue
บางครั้งระหว่างที่วนทำงานอยู่เราอยากที่จะหยุดระหว่างทางหรือข้ามรอบนั้นได้

Jumping ahead - continue การโดดไปรอบถัดไป

```
for i in range(1, 6):
    print()
    print("i =",i)
    print("hello, how " , end= "")
    if i == 3:
        continue
    print("are you today?")
```

```
i = 1
hello, how are you today?

i = 2
hello, how are you today?

i = 3
hello, how
i = 4
hello, how are you today?

i = 5
hello, how are you today?
```

จะเห็นได้ว่าเมื่อ i=3 เงื่อนไข if เป็นจริง เราจะเข้าไปทำงาน continue ซึ่ง จะทำให้เรากระโดดไปเริ่มที่ 4 โดยไม่ทำคำสั่งที่เหลือ

Bailing out--break การหยุดระหว่างทาง

ทางตรงข้ามเมื่อเราแก้ continue เป็น break เรา จะได้ผลลัพธ์ดังนี้

```
i = 1
hello, how are you today?

i = 2
hello, how are you today?

i = 3
hello, how
```

โปรแกรมจะหยุดที่ 3 ทันที

บางครั้งเราอาจจะทำเงื่อนไข ที่วนตลอดเช่น

```
>>> while True:
        print("Stop me please!!")


Stop me please!!
Stop me please!!
Stop me please!!
Stop me please!!
Stop me please!!
Stop me please!!
Stop me please!!
Stop me please!!
Stop me please!!
Stop me please!!
Stop me please!!
Stop me please!!
Stop me please!!
Stop me please!!
Stop me please!!
Stop me please!!
Stop me please!!
Stop me please!!
Stop me please!!
Stop me please!!
Stop me please!!
Stop me please!!
Stop me please!!
Stop me please!!
Stop me please!!
```

เงื่อนไขที่ while เป็นจริงตลอด (True) ไม่มีโอกาสออกจากลูป ยกเว้นการใส่ break
เราสามารถหยุด โปรแกรมด้วยการกด **Ctrl+C** เพราะฉะนั้นควรระหว่างการเขียนโปรแกรมที่ไม่มีทางออกจาก loop เช่น

```
x = 0
while x >= 0:
    print("x =",x)
    x += 1
```

ชีวิตอาจจะติด loop ได้

What did you learn?
ทบทวน

- **for** การวนแบบนับ
- **range()** การสร้างข้อมูลสำหรับการวน
- การกระโดดที่ละจำนวน หรือลดค่า ด้วยคำสั่ง **range()**
- การใช้ **whlie** กับเงื่อนไขการวน
- การกระโดดข้ามรอบด้วย **continue**
- การจบจากการวนระหว่างทาง **break**

Test your knowledge
แบบทดสอบ

1. จำนวนครั้งในการวนคำสั่งนี้กี่ครั้ง?
2. จำนวนครั้งในการวนคำสั่งนี้กี่ครั้ง? แล้วค่าตัวแปร i เป็นอะไรบ้างแต่ละรอบ
3. จงบอกว่าลิสต์ของตัวเลขมีอะไรบ้าง list คืออะไรที่เกิดจากคำสั่ง range(1, 8)
4. จงบอกว่าลิสต์ของตัวเลขมีอะไรบ้าง list คืออะไรที่เกิดจากคำสั่ง range(8)
5. จงบอกว่าลิสต์ของตัวเลขมีอะไรบ้าง list คืออะไรที่เกิดจากคำสั่ง range(2, 9, 2)
6. จงบอกว่าลิสต์ของตัวเลขมีอะไรบ้าง list คืออะไรที่เกิดจากคำสั่ง range(10, 0, -2)

7. คำสั่งอะไรที่ใช้สำหรับข้ามรอบในการวน และกระไปรอบต่อไป
8. while จบการทำงานตอนไหน

จงเขียนโปรแกรมต่อไปนี้

1. จงเขียนโปรแกรมเพื่อสร้างตารางสูตรคูณ โดยรับค่ามาตัวเลขแล้วสร้างตารางดังรูป
   Which multiplication table would you like? (สูตรคูณของตัวเลขไหน) โดยใช้คำสั่ง **for**

```
Which multiplication table would you like?
5
Here's your table:
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
```

2. จากข้อ 1 ให้ใช้คำสั่ง **while** แทน **for**
3. จากข้อหนึ่งให้ทำให้โปรแกรมสามารถระบุค่าสุดท้ายได้
   How high do you want to go? (ค่าสุดท้ายที่ต้องการระบุ)

```
Which multiplication table would you like?
7
How high do you want to go?
12
Here's your table:
7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
7 x 4 = 28
7 x 5 = 35
7 x 6 = 42
7 x 7 = 49
7 x 8 = 56
7 x 9 = 63
7 x 10 = 70
7 x 11 = 77
7 x 12 = 84
```

   ใช้ทำสองรูปแบบทั้ง while และ for