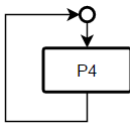
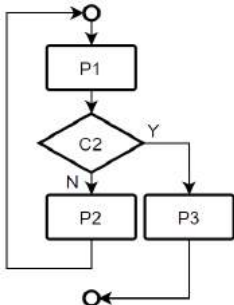
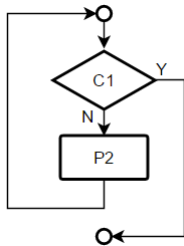
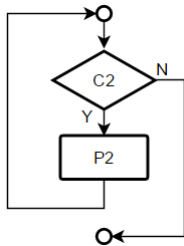


03 : Repetition (while, for)

สรุปเนื้อหา

Flowchart	Code	
	<pre>while True : P4</pre>	
	<pre>while True : P1 if C2 : P3 break P2</pre>	
	<pre>while True : if C1 : break P2</pre>	
	<pre>while True : if not C2 : break P2</pre>	<pre>while C2 : P2</pre>

Flowchart	Code
<pre> graph TD Start(()) --> C1{C1} C1 -- N --> End1(()) C1 -- Y --> P1[P1] P1 --> C2{C2} C2 -- N --> P3[P3] P3 --> End2(()) C2 -- Y --> P2[P2] P2 --> C2 </pre>	<pre> while C1 : P1 while C2 : P2 P3 </pre>

ให้สังเกตว่า ภายในวงวน while ควรมีคำสั่งที่เปลี่ยนแปลงเงื่อนไขของ while ไม่งั้นมันจะวนทำงานไม่สิ้นสุด เช่น

```
while i < j :
```

```
...
```

```
i += 2
```

เงื่อนไข $i < j$ ข้างบนนี้จะเป็เท็จได้ (เพื่อให้ออกจากวงวน) ก็ด้วยการที่ค่าของ i เพิ่มขึ้น หรือค่าของ j ลดลง

คำสั่ง $i += 2$ ในตัวอย่างข้างบนสร้างความมั่นใจว่า วงวนนี้ทำงานแล้วจะมีจุดสิ้นสุดและออกจากวงวน

Flowchart	Code	Flowchart	Code
<pre>graph TD; A[k = 0] --> B(()); B --> C{k < n}; C -- N --> D(()); C -- Y --> E[P1]; E --> F[k += 1]; F --> B;</pre>	<pre>k = 0 while k < n : P1 k += 1</pre>	<pre>graph TD; A(()) --> B{k = 0 ... n-1}; B -- Done --> C(()); B -- Not Done --> D[P1]; D --> B;</pre>	<pre>for k in range(n) : P1</pre>
		<pre>graph TD; A(()) --> B{k = 0 ... m-1}; B -- Done --> C(()); B -- Not Done --> D{k = 0 ... n-1}; D -- Done --> B; D -- Not Done --> E[P1]; E --> D;</pre>	<pre>for p in range(m) : for k in range(n) : P1</pre>

Flowchart	Code
<pre> graph TD Start(()) --> Cond1{k = 0 ... m-1} Cond1 -- Done --> End1(()) Cond1 -- Not Done --> P1[P1] P1 --> Cond2{C} Cond2 -- Y --> P3[P3] Cond2 -- N --> P2[P2] P3 --> P4[P4] P2 --> Cond1 P4 --> End1 </pre>	<pre> for k in range(m) : P1 if C : P3 break P2 else: P4 </pre> <p>จะมาทำหลัง else ของ for ก็เมื่อทำครบทุกรอบ หลังทำรอบที่ k = m-1 เสร็จ ก็มาทำที่ P4 ก่อนออกจากวงวน</p>
<pre> graph TD Start(()) --> Cond1{C1} Cond1 -- N --> P4[P4] Cond1 -- Y --> P1[P1] P1 --> Cond2{C2} Cond2 -- Y --> P3[P3] Cond2 -- N --> P2[P2] P3 --> P4 P2 --> Cond1 P4 --> End1(()) </pre>	<pre> while C1 : P1 if C2 : P3 break P2 else: P4 </pre> <p>จะมาทำหลัง else ของ while ก็เมื่อทำงานเงื่อนไข C1 ของ while เป็นเท็จ ก็มาทำที่ P4 ก่อนออกจากวงวน</p>

range(start, stop, step)

- start, stop และ step ต้องเป็นจำนวนเต็ม
- for k in range(10) : k = 0, 1, 2, ..., 9
- for k in range(2,10) : k = 2, 3, 4, ..., 9
- for k in range(2,10,2) : k = 2, 4, 6, 8
- for k in range(10,1,-2) : k = 10, 8, 6, 4, 2
- for k in range(11,11) : ไม่ทำซ้ำรอบ เพราะ step เป็นบวก และ start >= stop
- for k in range(9,10,-1) : ไม่ทำซ้ำรอบ เพราะ step ติดลบ และ start <= stop

*** break จะย้ายการทำงานออกจากวงวนที่ break นั้นอยู่เท่านั้น

```

for i in range(5):
    for j in range(6):
        if condition1 :
            break          # break นี้ออกจาก for j
        if condition2 :
            break          # break นี้ออกจาก for i
  
```

วงวนที่พบบ่อย

<p>เมื่อต้องการทำอะไรบางอย่างซ้ำกัน n ครั้ง</p> <p>ใช้ <code>for k in range(n)</code></p>	<p>เช่น ต้องการอ่านข้อมูลจำนวน 10 ตัว เพื่อหาค่าเฉลี่ย</p> <pre>s = 0 for k in range(10) : # เป็นแค่คำสั่งควบคุมจำนวนรอบการทำซ้ำ a = float(input()) s += a print('average =', (s/10))</pre>
<p>เมื่อต้องการนำค่าที่สร้างจาก <code>range(start, stop, step)</code> มาใช้ในการประมวลผล</p> <p>ใช้ <code>for k in range(...)</code></p>	<p>เช่น ต้องการหาว่า q เป็นจำนวนเฉพาะหรือไม่ ใช้วงวน <code>for</code> หาว่าจำนวนเต็มมากกว่าหนึ่งขนาดเล็กสุดอะไร ที่หาร q ลงตัว</p> <pre>for k in range(2, q+1) : # k = 2,3,...,q if q % k == 0 : break # มีการนำ k มาใช้ if k == q : print(q,'is prime') else: print(q, '=', k, 'x', q//k)</pre>
<p>เมื่อต้องการประมวลผลชุดคำสั่งซ้ำ ๆ จนกว่าเงื่อนไขหนึ่งจะเป็นจริง</p> <p>ใช้ <code>while</code> หรือใช้ <code>if break</code> ในวงวน</p>	<p>เช่น ต้องการหาค่าเฉลี่ยจากชุดข้อมูลที่ผู้ใช้ป้อนเข้ามาเรื่อย ๆ จนกว่าจะรับจำนวนติดลบ</p> <div> <pre>s = 0 n = 0 while True : t = float(input()) if t < 0 : break s += t n += 1 if n == 0 : print('No Data') else : print('avg =',(s/n))</pre> <pre>s = 0 n = 0 t = float(input()) while t >= 0 : s += t n += 1 t = float(input()) if n == 0 : print('No Data') else : print('avg =',(s/n))</pre> </div>
<p>เมื่อต้องการแจกแจงวิธีการเลือกหมายเลขจากหมายเลข 0 ถึง n-1 จำนวน 2 หมายเลข แบบเลือกแล้วไม่เลือกอีก (คือแจกแจงการเลือกหมายเลขออกมาทีละคู่)</p> <pre>for i in range(n) : for j in range(i+1, n) : ได้ i<j ทุก ๆ กรณี</pre> <p>หรือถ้าต้องการให้ i = j ด้วย ก็เป็น</p> <pre>for i in range(n) : for j in range(i, n) : ได้ i≤j ทุก ๆ กรณี</pre>	<p>เช่น จงหาว่ามีจำนวนเต็ม x y และ z อะไรบ้างที่ทำให้สมการ $z^3 = x^2 + y^2$ เป็นจริง (x กับ y มีค่า 0 ถึง 19) เช่น $5^3 = 5^2 + 10^2$ แต่เราไม่ต้องการคำตอบ $5^3 = 10^2 + 5^2$ เพราะซ้ำ จึงต้องกำหนดว่า $x < y$ แต่ถ้าเราต้องการคำตอบ $8^3 = 16^2 + 16^2$ ด้วย ก็ต้องให้ $x \leq y$ โดยให้ y มีค่าเริ่มที่ x เป็นต้นไป ด้วย <code>for y in range(x,n)</code> ข้างล่างนี้</p> <pre>n = 20 for x in range(1,n) : for y in range(x,n): t = x**2 + y**2 z = int(round(t**(1/3),0)) # ทารากที่สามแล้ว # ปิดเคส if z**3 == t : print(z,x,y)</pre>

เรื่องพิศบอย

<p>เข้าใจผิดเรื่องตัวสุดท้ายของ range</p>	<p>for k in range(1,5) k = 1,2,3,4 (ไม่รวม 5)</p>
<p>ใช้วงวน ทำอะไรบางอย่าง เพื่อสรุปว่า เป็น A หรือ เป็น B โดยจะเป็น A เมื่อเงื่อนไข C เป็นจริง อย่างน้อยหนึ่งครั้ง แต่จะเป็น B เมื่อ C ต้องไม่เป็นจริงทุกครั้งทุกรอบ ถ้าเขียน</p> <pre>for k in range(...): if C: print('A') else: print('B')</pre> <p>แบบนี้ผิด เพราะสรุปว่าเป็น B เร็วไป ยังตรวจไม่ครบทุกรอบทุกกรณี แก้ไขด้วยการใช้ตัวแปร เก็บสถานะการตรวจ</p> <pre>found = False for k in range(...): if C: print('A') found = True break if not found: print('B')</pre> <p>หรือใช้ for-else ก็ง่ายกว่า</p> <pre>for k in range(...): if C: print('A') break else: print('B')</pre>	<p>เช่น ต้องการหาว่า q เป็นจำนวนเฉพาะหรือไม่</p> <pre>for k in range(2, q): if q % k == 0: print(q, 'is composite') # สรุปถูก เพราะ # หาตัวหารพบ else: print(q, 'is prime') # ผิด สรุปเร็วไป # ต้องวนทดสอบต่อ</pre> <p>แก้ไขโดยใช้ตัวแปรเก็บสถานะการตรวจ เป็นดังนี้</p> <pre>iscomposite = False for k in range(2,q): if q % k == 0: print(q, 'is composite') iscomposite = True break if not iscomposite: print(q, 'is prime')</pre> <p>หรือใช้ for-else ก็ง่ายกว่า</p> <pre>for k in range(2,q): if q % k == 0: print(q, 'is composite') break else: print(q, 'is prime')</pre>
<p>วงวน ทำจำนวนรอบน้อยไปหรือมากไปกว่าที่ต้องการ (โดยทั่วไปมักขาดหรือเกินไปหนึ่งรอบ)</p>	<p>เช่น จากชุดคำสั่งตรวจสอบจำนวนเฉพาะข้างบนนี้ ถ้าเขียน</p> <pre>for k in range(2,q+1): # เขียน q+1 แทนที่จะเป็น q if q % k == 0: print(q, 'is composite') break else: print(q, 'is prime')</pre> <p>แบบนี้เกินไปรอบ ทำให้ผลออกมาเป็น composite เสมอ เพราะอะไร ?</p> <p>หรือ อยากหาค่าของ $\sum_{k=1}^n k^3$ ถ้าเขียน</p> <pre>s = 0 for k in range(1,n): s += k**3 print(s)</pre> <p>ก็จะพบว่าขาดไปรอบ</p>

<p>ลืมนับค่าของตัวแปรที่ใช้ในเงื่อนไขของ while หรือไม่ก็ปรับค่าผิด ความผิดพลาดแบบนี้อาจทำให้วนทำงานไม่สิ้นสุด</p>	<p>เช่น ต้องการรับจำนวนจากผู้ใช้ มาหาค่าเฉลี่ยจนกว่าจะพบจำนวนลบ</p> <pre>s = n = 0 t = float(input()) while t >= 0 : s += t n += 1 print('avg =', (s/n))</pre> <p>ใช้ t ในการตรวจสอบเงื่อนไขของ while แต่ค่า t ไม่ได้เปลี่ยนแปลงเลยในวงวน ถ้าหลุดเข้ามาในวงวนได้จะเกิดอะไรขึ้น ?</p> <p>ควรแก้เป็น</p> <pre>s = n = 0 t = float(input()) while t >= 0 : s += t n += 1 t = float(input()) # เพิ่มบรรทัดนี้ print('avg =', (s/n))</pre>
<p>ตั้งค่าให้กับตัวแปรที่ควรจะให้ค่าก่อนเข้าวงวน แต่กลับไปเขียนในวงวน</p>	<p>เช่น ต้องการรับจำนวนจากผู้ใช้ มาหาค่าเฉลี่ยจนกว่าจะพบจำนวนลบ</p> <pre>while True : s = n = 0 # บรรทัดนี้ไม่น่ามาอยู่ในวงวน t = float(input()) if t < 0 : break s += t n += 1 print('avg =', (s/n))</pre> <p>จะเกิดอะไรขึ้น ?</p>

เรื่องแปลกของ for ใน Python

<p>หลังจากวนทำงานใน for k จนเรียบร้อยแล้ว ค่า k หลังออกจากวงวน จะมีค่าเท่ากับค่าสุดท้ายที่ทำงานในวงวน</p>	<pre>for k in range(1,5): ... print(k) # ได้ 4 เพราะเป็นค่าสุดท้ายที่ทำในวงวน for m in range(4,10): if m % 3 == 0 : break ... print(m) # ได้ 6 เพราะค่าสุดท้ายในวงวนคือ 6 ก่อนจะ break ออก for w in range(10,10): ... print(w) # ผิด เพราะไม่ได้เข้าไปทำในวงวน, w จึงไม่มีค่า</pre>
---	---