Python Decision Making And Loop

PEP หรือ Python Enhancement Proposal คือข้อเขียน/ข้อเสนอโครงการสำหรับปรับปรุงเพิ่มความสามารถ Python ให้ดีขึ้น โดยที่ PEP 8 นั้นเป็นข้อเขียนสำหรับ coding standard ของ Python ที่เราควรยึดถือ

ตัวอย่างเช่น

- 4 white spaces or 1 tab for indentation
- อย่าให้แต่ละบรทัดเกิน 79 ตัวอักษร
- คั่นส่วนต่างๆของโปรแกรมด้วยการเว้นบรรทัด

Python Decision Making

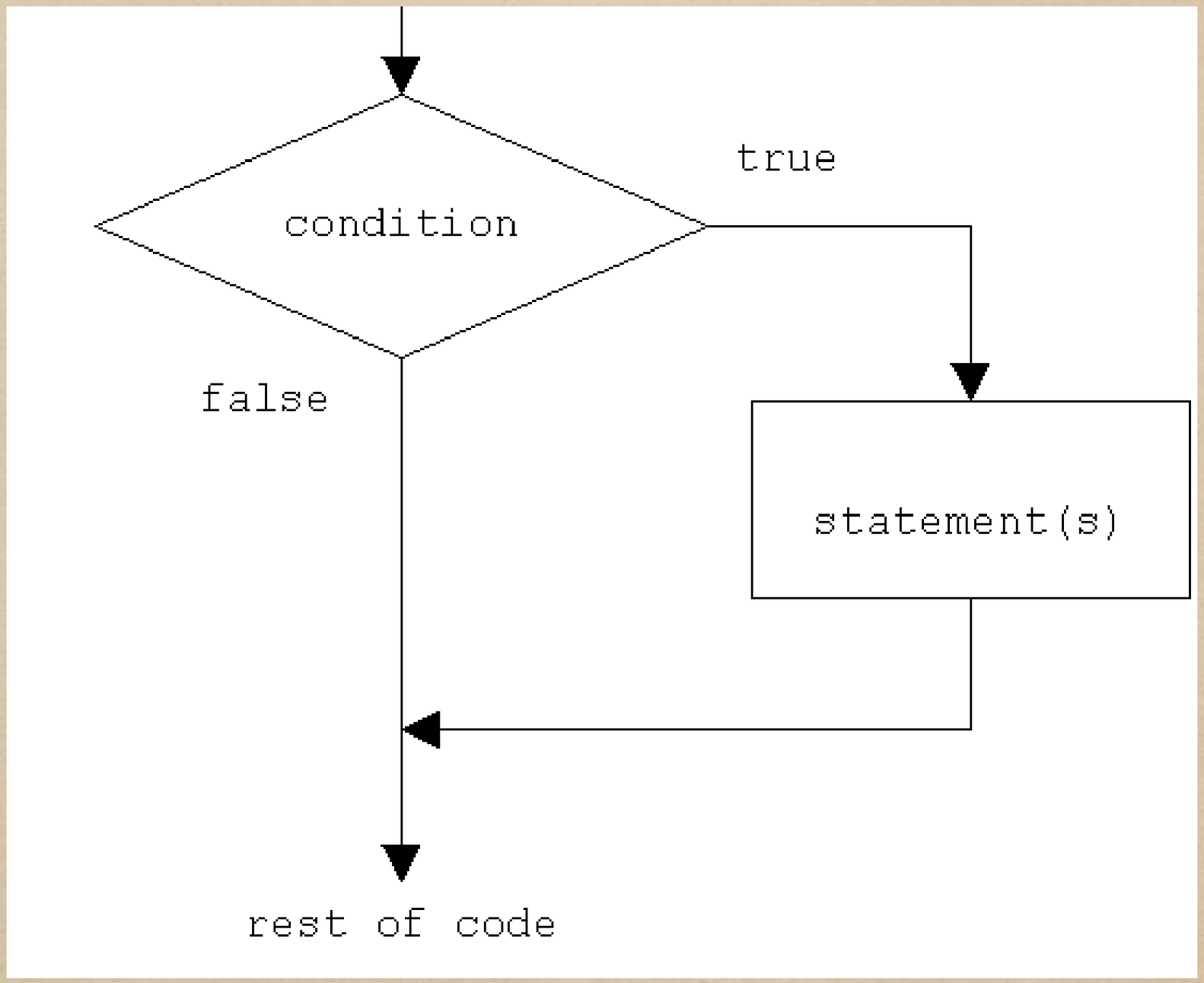
การตัดสินใจ(Decision)

1. if statements

คำสั่ง if ใช้ตัดสินใจว่าจะทำหรือไม่ทำคำสั่งชุดหนึ่งที่อยู่ภายในช่วงของการทำงานของ if ถ้าเงื่อนไขที่นำมาทดสอบทางตรรกศาสตร์เป็นจริง (True) ก็จะทำคำสั่งชุดนั้นถ้าเงื่อนไขที่นำมาทดสอบทางตรรกศาสตร์เป็นเท็จ (False) ก็จะไม่ทำคำสั่งชุดนั้น

Syntax :

```
if expression:
    statement(s)
```

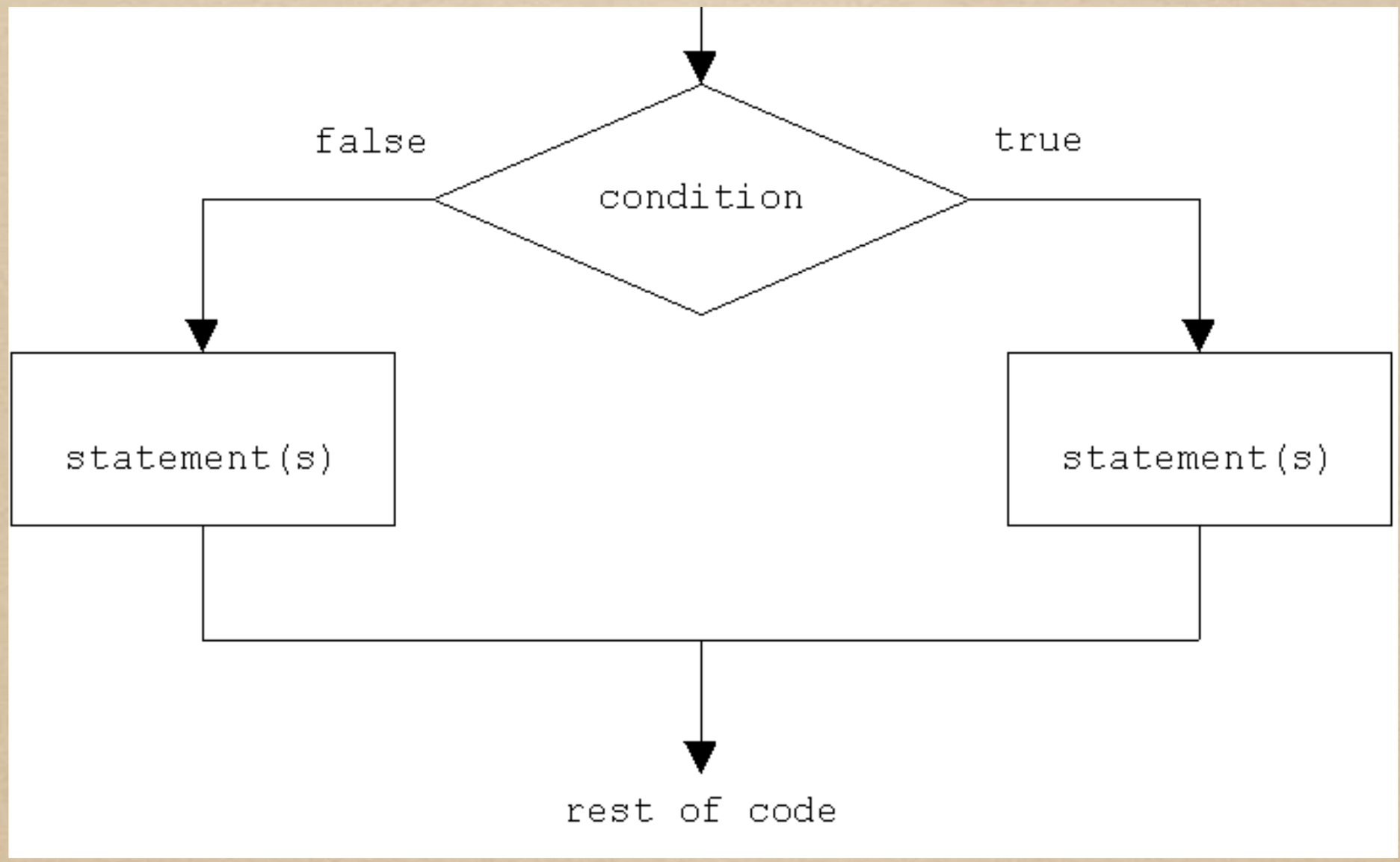

condition
true
false
statement(s)
rest of code

2. if…else statements

คำสั่ง else สามารถใช้ร่วมกับคำสั่ง if ได้ คำสั่ง else จะกระทำก็ต่อเมื่อการประมวลผลของคำสั่ง if ไม่ตรงตามเงื่อนไขที่ตรวจสอบ คำสั่ง else จะมีได้ก็ต่อเมื่อมีคำสั่ง if แต่คำสั่ง if ไม่จำเป็นต้องมีคำสั่ง else

Syntax :

```
if expression:
    statement(s)
else:
    statement(s)
```


false
condition
true
statement(s)
statement(s)
rest of code

Example :

```
x = 1
If x is 1:
    print(“Yes”)
else:
    print(“No”)
```

Result :
Yes

3. elif statements

ในบางครั้งการตัดสินใจในการทำชุดคำสั่งอาจมีมากกว่า 2 ทางข้างต้น เราสามารถในคำสั่งอีกหนึ่งคำสั่งที่ชื่อว่า elif ได้ (หรือมากจาก else if ในภาษาอื่น เช่น ภาษา java หรือ c เป็นต้น)

Syntax :

```
if expression1:
    statement(s)
elif expression2:
    statement(s)
elif expression3:
    statement(s)
else: statement(s)
```

Example :

```
x = 3
if x < 1:
    print(“x < 1”)
elif  x > 1:
    print(“x > 1”)
else:
    print(“x = 1”)
```

Result :

x > 1

4. nested if statements

ในบางเหตุการณ์เมื่อเราต้องการเช็คเงื่อนไขอื่นๆหลังจากเงื่อนไขก่อนหน้านั้นเป็นจริง เราสามารถใช้ nested if ได้

Syntax :

```
if expression1:
    statement(s)
    if expression2:
        statement(s)
    else:
        statement(s)
elif expression3:
    statement(s)
else:
    statement(s)
```

Example :

```
 var = 100
if var < 200:
      print"Expression value is less than 200"
      if var == 150:
            print"Whichis 150"
      elif "var == 100"
            print"Whichis 100"
      elif var==50:
            print"Which is 50"
      elif var<50
            print"Expression value is less than 50"
      else
            print"Could not find true expression"
            print"Good bye!"
```

Result:

```
Expression value is less than 200
Which is 100
```

- โจทย์

อาจารย์ต้องการโปรแกรมเพื่อตัดเกรดนักศึกษา โดย
มีเงื่อนไขคือ

จะได้ A ต้องมีคะแนนมากกว่าเท่ากับ80

B ต้องมีคะแนนมากกว่าเท่ากับ 70

C ต้องมีคะแนนมากว่าเท่ากับ 60

D ต้องมีคะแนนมากกว่าเท่ากับ 50

F คะแนนน้อยกว่า 50

โดยจะเก็บคะแนนเป็น midterm 35% final 50% และ การบ้าน 15%

โดยการเก็บคะแนนแต่ละครั้งมีคะแนนเต็ม 100 คะแนน

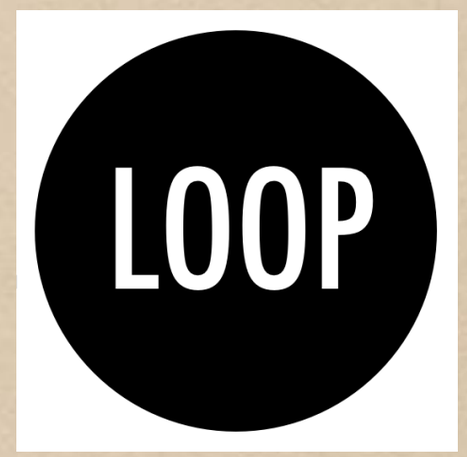
ตัวอย่างข้อมูลนำเข้า

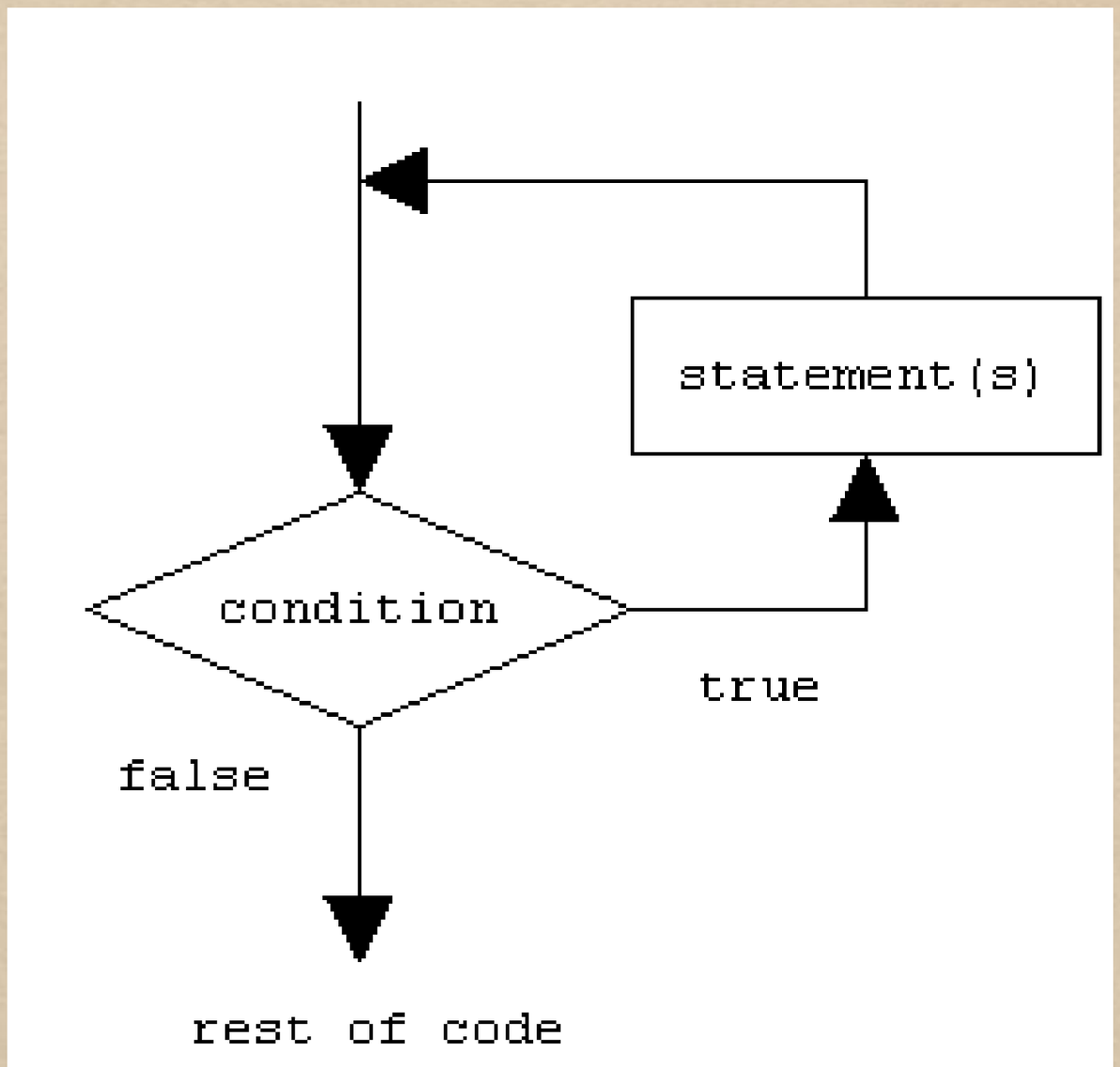
midterm : 60
final :65
homework :90

ข้อมูลส่งออก

67.0
grade : C

python




statement(s)
condition
true
false
rest of code

1. while loop

คำสั่ง while เป็นคำสั่งที่ใช้ในการวนทำซ้ำในช่วงของการทำงานของ while จนกว่าการทดสอบทางตรรกศาสตร์จะเป็นเท็จ มีรูปแบบดังนี้

Syntax :

```
while expression:

    statement(s)
```

Example :

```
count = 0
while count < 5:
    print (“The count is:”, count)
    count = count + 1
print ("Good bye!“)
```

Result :

```
The count is: 0
The count is: 1
The count is: 2
The count is: 3
The count is: 4
Goog bye
```

The Infinite Loop

loop จะเป็น infinite loop ได้ก็ต่อเมื่อ เงื่อนไขมีค่าเป็น true ตลอด เราต้องมีความระมัดระวังในการใช้ while loop เพราะมีโอกาสเป็นไปได้ที่เงื่อนไขจะไม่มีค่าเป็น false ซึ่งผลลัพธ์จะได้เป็น infinite loop

Example :

```
var = 1
while var == 1 : # This constructs an infinite loop
    num = input("Enter a number :")
    print( "You entered: ", num)
print("Good bye!“)
```

Result :

```
Enter number : 1
You entered: 1
.
.
.
.
```

The else Statement use with loop

เราสามารถประยุกต์ใช้ else statement กับ while loop ได้

Syntax :

```
while  expression:
    statements(s)
else:
    statements(s)
```

Example :

```
count = 0
while count < 5:
    print count, " is less than 5"
    count = count + 1
else:
    print count, " is not less than 5"
```

Result:

```
0 is less than 5
1 is less than 5
2 is less than 5
3 is less than 5
4 is less than 5
5 is not less  than 5
```

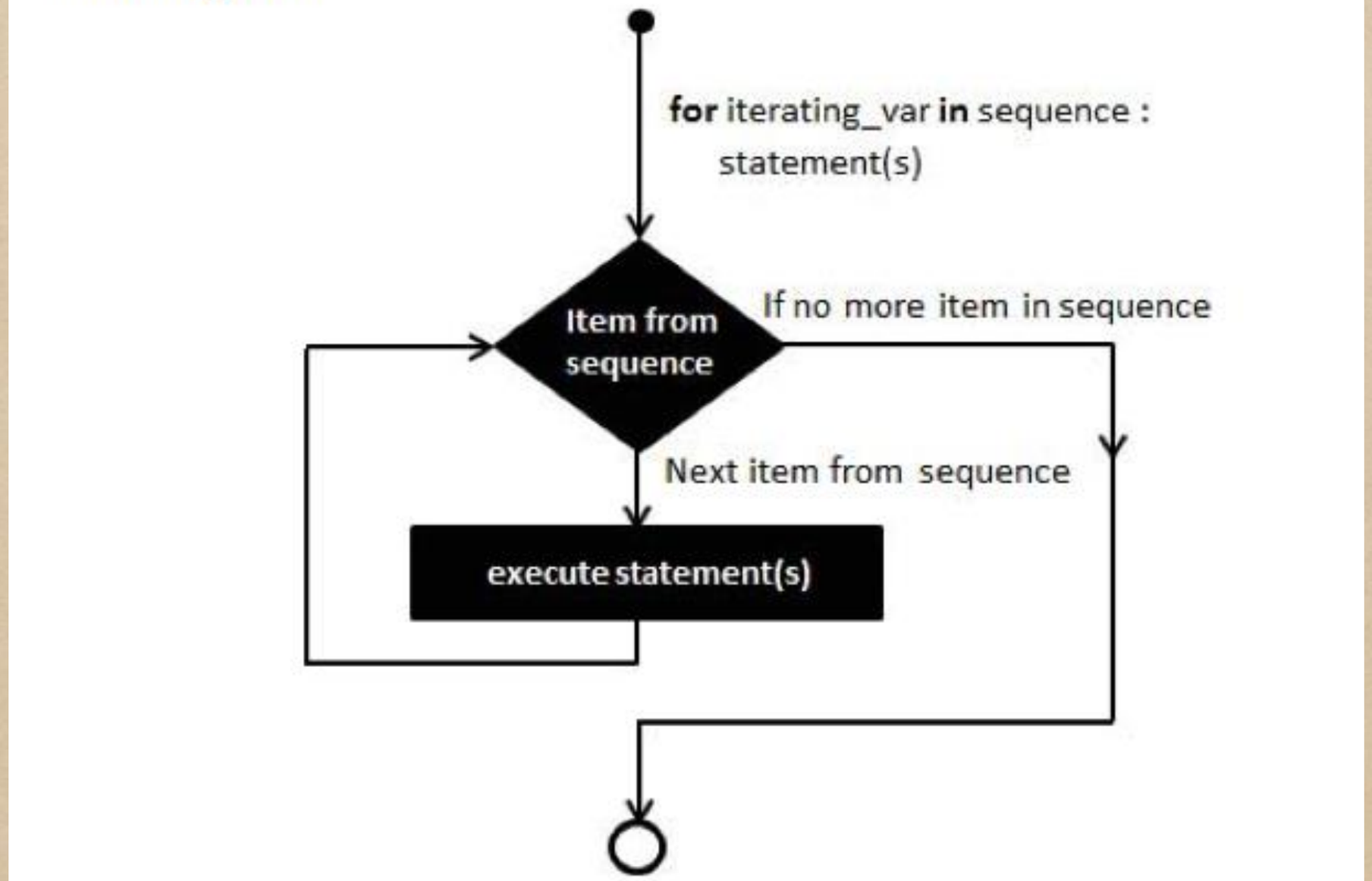
2. for loop

for loop ใน Python สามารถวนซ้ำสิ่งที่เป็นลำดับได้ เช่น list หรือ string

Syntax :

```
for iterating_var in sequence:
    statements(s)
```

Flow Diagram:



Example :

```
for letter in “Python”:
    print (“Current Letter :”, letter)
fruits = ['banana', 'apple', 'mango']
for fruit in fruits:
    print (“Current fruit :”, fruit)
print ("Good bye!“)
```

Result:

```
Current Letter: P
Current Letter: y
Current Letter: t
Current Letter: h
Current Letter: o
Current Letter: n
Current fruit: banana
Current fruit: apple
Current fruit: mango
Good bye!
```

Iterating by Sequence Index:

หนึ่งในวิธีการที่จะวนซ้ำ item ก็คือการใช้ index ในการเข้าถึงตามลำดับ

Example :

```
fruits = ['banana', 'apple','mango']
for index in range(len(fruits)):
    print (“Current fruit :”, fruits[index])
print ("Good bye!”)
```

Result:

Current fruit: banana
Current fruit: apple
Current fruit: mango
Good bye!

รูปแบบฟังก์ชั่น range(start, end, step)

The else Statement Used with Loops

เราสามารถประยุกต์ใช้ else statement กับ for loop ได้

Example :

```
for num in range(10,20):
    for i in range(2,num):
        if num%i == 0:
            j=num/i
            print '%d equals %d * %d' % (num,i,j)
            break
    else:
        print( num, “is a prime number”)
```

รูปแบบฟังก์ชั่น range(start, end, step)

Result :

10 equals 2 * 5
11 is a prime number
12 equals 2 * 6
13 is a prime number
14 equals 2 * 7
15 equals 3 * 5
16 equals 2 * 8
17 is a prime number
18 equals 2 * 9
19 is a prime number

nested loops

Python อนุญาติให้ใช้ loop ซ้อนกันได้

1.nested for loop

```
for iterating_var in sequence:
    for iterating_var in sequence:
        statements(s)
    statements(s)
```

2.nested while loop

```
while expression:
    while expression:
        statement(s)
    statement(s)
```

Example :

```
i=2
while(i < 10):
    j=2
    while(j <= (i/j)):
        if not(i%j): break
        j = j + 1
    if (j > i/j) : print (i, " is prime" )
    i = i + 1
print ("Good bye!“)
```

Result :

2 is prime
3 is prime
5 is prime
7 is prime
Good bye!

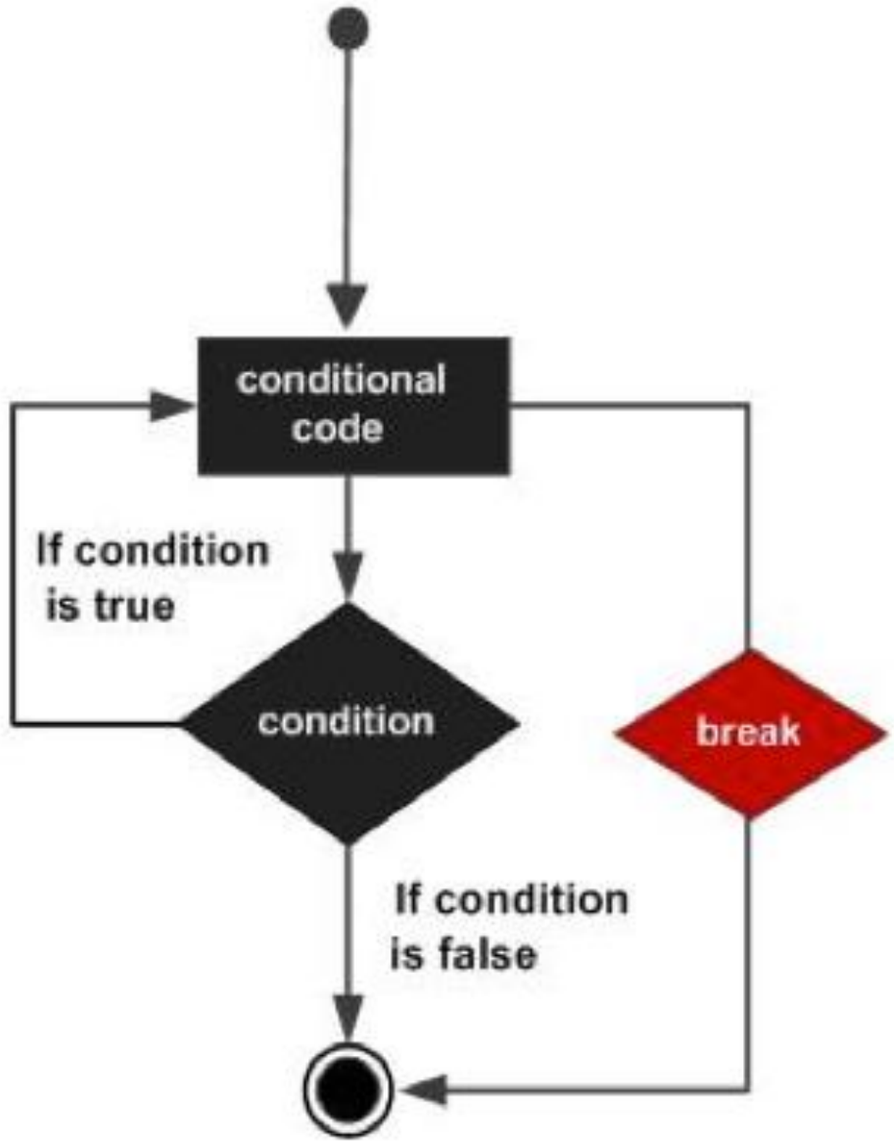
Loop Control Statements:

Loop Control Statements จะเปลี่ยนแปลงการทำงานจากลำดับตามปกติ เมื่อการทำงานอยู่นอกเหนือขอบเขต

1.คำสั่ง break

เป็นคำสั่งที่ให้โปรแกรมออกจาก loop ทันที โดยไม่ทำคำสั่งที่เหลือต่อ

Flow Diagram:



Example:

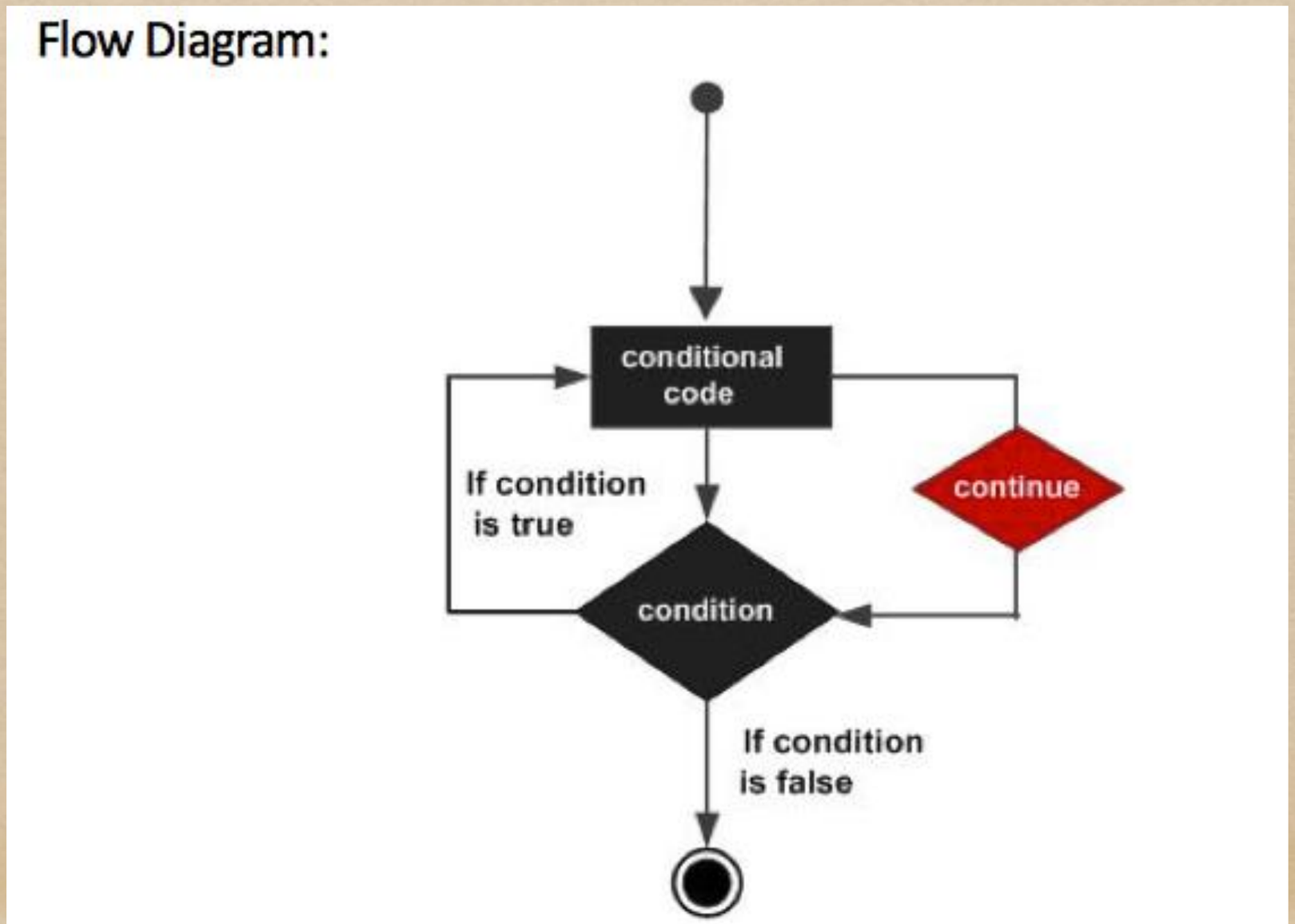
```
for letter in 'Python':
      if letter == 'h':
          break
     print (“Current Letter :” ,letter)
```

Result:

Current Letter: P
Current Letter: y
Current Letter: t

2.คำสั่ง continue

เป็นคำสั่งที่ให้โปรแกรม กลับไปทำงานที่ต้น loop โดยไม่ทำคำสั่งที่เหลือต่อ



Example :

```
for letter in 'Python':
    if letter == 'h':
        continue
    print 'Current Letter :', letter
```

Result:

```
Current Letter: P
Current Letter: y
Current Letter: t
Current Letter: o
Current Letter: n
```

3.คำสั่ง pass

เป็นคำสั่งที่ใช้เมื่อต้องการสร้าง statement แต่ไม่ได้ทำงานใด ๆ

Example :

```
for letter in ‘Python':
     if letter == 'h':
        pass
        print 'This is pass block'
    print 'Current Letter :', letter
print "Good bye!"
```

Result:

```
Current Letter: P
Current Letter: y
Current Letter: t
This is pass block
Current Letter: h
Current Letter: o
Current Letter: n
Good bye!
```

Ex.

จงเขียนโปรแกรมคำนวณค่า **factorial (โดยห้ามใช้library)**

ตัวอย่างข้อมูลเข้า 5! = 5x4x3x2x1x1 = 120

The End

ผู้จัดทำ

นางสาวปริยา วัฒนาราม รหัสนักศึกษา 07550454
นายพงศกร โชคสมงาม รหัสนักศึกษา 07550455
นายศุภกร ชัยศรีนุวัฒน์ รหัสนักศึกษา 07550475