

LECTURE 2-3_2

PYTHON PROGRAMMING

CODES AND STRUCTURES

DR. PRAPASSORN TANTIPHANWADI

INDUSTRIAL ENGINEERING, FACULTY OF ENGINEERING AT KHAMPAENGSAEN

DECEMBER 2565

REVIEW - SET

Create

```
In [160]: ┏━━━  
         1 | set1 = {1, 2, 3, 4, 5}  
         2 | set2 = {1, 2, 3, 'a', 'b', 'c'}
```

```
In [161]: ┏━━━  
         1 | type(set1)
```

Out[161]: set

```
In [162]: ┏━━━  
         1 | type(set2)
```

Out[162]: set

REVIEW - SET

Read

```
In [163]: ► 1 set3 = {'a', 'b', 'c', 'd', 'e'}
```

```
In [164]: ► 1 for x in set3:  
          2     print(x)
```

d
b
c
e
a

REVIEW - SET

add

```
In [165]: 1 set4 = {'a', 'b', 'c', 'd', 'e'}
```

```
In [166]: 1 set4.add(1)
```

```
In [167]: 1 set4
```

```
Out[167]: {1, 'a', 'b', 'c', 'd', 'e'}
```

REVIEW - SET

update

```
In [168]: 1 set5 = {'a', 'b', 'c', 'd', 'e'}
```

```
In [169]: 1 set5.update({1, 2})
```

```
In [170]: 1 set5
```

```
Out[170]: {1, 2, 'a', 'b', 'c', 'd', 'e'}
```

REVIEW - SET

remove

```
In [171]: 1 set6 = {'a', 'b', 'c', 'd', 'e'}
```

```
In [172]: 1 set6.remove('a')
```

```
In [173]: 1 set6
```

Out[173]: {'b', 'c', 'd', 'e'}

```
In [174]: 1 set6.remove('c')
```

```
In [175]: 1 set6
```

Out[175]: {'b', 'd', 'e'}

REVIEW - SET

clear

```
In [176]: 1 set7 = {'a', 'b', 'c', 'd', 'e'}
```

```
In [177]: 1 set7.clear()
```

```
In [178]: 1 set7
```

Out[178]: set()

REVIEW - SET

len

```
In [179]: 1 set8 = {1, 2, 3, 4, 5}
```

```
In [180]: 1 len(set8)
```

```
Out[180]: 5
```

REVIEW - SET

in

```
In [181]: 1 set9 = {1, 2, 3, 4, 5}
```

```
In [182]: 1 1 in set9
```

Out[182]: True

```
In [183]: 1 'a' in set9
```

Out[183]: False

REVIEW - SET

union

In [184]:

```
1 setA = {1, 2, 3, 4, 5}  
2 setB = {3, 4, 5, 6, 7}
```

In [185]:

```
1 setA | setB
```

Out[185]: {1, 2, 3, 4, 5, 6, 7}

REVIEW - SET

Intersection

```
In [186]: 1 setA = {1, 2, 3, 4, 5}  
           2 setB = {3, 4, 5, 6, 7}
```

```
In [187]: 1 setA & setB
```

Out[187]: {3, 4, 5}

REVIEW - SET

Difference

```
In [188]: ► 1 setA = {1, 2, 3, 4, 5}  
          2 setB = {3, 4, 5, 6, 7}
```

```
In [189]: ► 1 setA - setB
```

Out[189]: {1, 2}

```
In [190]: ► 1 setB - setA
```

Out[190]: {6, 7}

REVIEW - SET

Symmetric Difference

In [191]:

```
1 setA = {1, 2, 3, 4, 5}  
2 setB = {3, 4, 5, 6, 7}
```

In [192]:

```
1 setA ^ setB
```

Out[192]: {1, 2, 6, 7}

CONCLUSION

	List	Tuple	Dictionary	Set
Create	<code>listA=[]</code> <code>listB=[1, 2, 3]</code>	<code>tupleA=()</code> <code>tupleB=(1, 2, 3)</code>	<code>dictA={}</code> <code>dictB={'key':value}</code>	<code>setA=set()</code> <code>setB={1, 2, 3}</code>
Read	<code>listB[start]</code> <code>listB[start:end]</code>	<code>tupleB[start]</code> <code>tupleB[start:end]</code>	<code>dictB['key']</code>	<code>for b in setB:</code> <code> print(b)</code>

CONCLUSION

	List	Tuple	Dictionary	Set
Update	Replace append insert extend	✓ ✓ ✓ ✓	X	✓ X X X X X X
Access	Index Slice	✓ ✓	✓ X	✓ X
Iteration	for loop	for loop	for loop	for loop

CONCLUSION

		List	Tuple	Dictionary	Set
Delete	<code>del</code>	✓		✓	✗
	<code>remove</code>	✓	✗	✗	✓
	<code>clear</code>	✓		✓	✓
	<code>len</code>	<code>len(listB)</code>	<code>len(tupleB)</code>	<code>len(dictB)</code>	<code>len(setB)</code>
	<code>in</code>	<code>'a' in listB</code>	<code>'a' in tupleB</code>	<code>'keyA' in dictB</code>	<code>'a' in setB</code>

CONCLUSION

Set

Union (`|`)

Intersection (`&`)

Difference (`-`)

Symmetric Difference
(`^`)

OPERATOR

$$y = x^2 + 2x + 1$$

- $y, x, 1$ គឺ តាមត្រូវកាំបើនការ (Operand)
- $=, +$ គឺ តាមកាំបើនការ (Operator)
- $y = x^2 + 2x + 1$ កំណែអុំដ គឺ ឯករាជ្យ (expression)

OPERATOR

1. Arithmetic Operators
2. Comparison Operators
3. Logical Operators
4. Membership Operators
5. Operator Precedence

ARITHMETIC OPERATOR

1.1 ບົກ (+)

1.2 ລູ (-)

1.3 ຄູນ (*)

1.4 ອາດ (/)

1.5 ຍກກຳລັງ (**)

1.6 ເສເຫຈາກຮາດ (%)

1.7 ພລຮາກປັດເສບລົງ (//)

1.1 ບົກ (+)

In [1]:

1	a = 5
2	b = 10
3	c = 7

In [2]:

1	a + b
---	-------

Out[2]: 15

In [3]:

1	print(b + c)
---	--------------

17

ARITHMETIC OPERATOR

1.2 ຂົງ (-)

In [4]: 1 | a = 5
2 | b = 10
3 | c = 7

In [5]: 1 | a - b

Out[5]: -5

In [6]: 1 | b - c

Out[6]: 3

1.3 ອູບ (*)

In [7]: 1 | a = 5
2 | b = 10
3 | c = 7

In [8]: 1 | a * b

Out[8]: 50

In [9]: 1 | b * c

Out[9]: 70

ARITHMETIC OPERATOR

1.4 หาร (/)

In [10]:

1	a = 5
2	b = 10
3	c = 7

In [11]:

1	a/b
---	-----

Out[11]: 0.5

In [12]:

1	b/c
---	-----

Out[12]: 1.4285714285714286

1.5 ยกกำลัง (**)

In [13]:

1	a = 2
2	b = 3
3	c = 4

In [14]:

1	a**b
---	------

Out[14]: 8

In [15]:

1	b**c
---	------

Out[15]: 81

ARITHMETIC OPERATOR

1.6 ເສີ່ງຈາກຮາດ (%)

```
In [16]: 1 a = 5  
          2 b = 10  
          3 c = 7
```

```
In [17]: 1 a%b
```

Out[17]: 5

```
In [18]: 1 b%c
```

Out[18]: 3

1.7 ພາຫາກປັດເສີ່ງ (//)

```
In [19]: 1 a = 5  
          2 b = 10  
          3 c = 7
```

```
In [20]: 1 a//b
```

Out[20]: 0

```
In [21]: 1 b//c
```

Out[21]: 1

COMPARISION OPERATORS

2.1 ตรวจสอบความเท่ากัน (==)

2.2 ตรวจสอบความไม่เท่ากัน (!=)

2.3 ตรวจสอบความมากกว่า (>)

2.4 ตรวจสอบความน้อยกว่า (<)

2.5 ตรวจสอบความมากกว่าหรือเท่ากับ (>=)

2.6 ตรวจสอบความน้อยกว่าหรือเท่ากับ (<=)

COMPARISION OPERATORS

2.1 ตรวจสอบความเท่ากัน (==)

```
In [22]: 1 a = 5  
          2 b = 'name'
```

```
In [23]: 1 a == 5
```

Out[23]: True

```
In [24]: 1 a == 7
```

Out[24]: False

```
In [25]: 1 b == 'name'
```

Out[25]: True

```
In [26]: 1 b == 'fullname'
```

Out[26]: False

2.2 ตรวจสอบความไม่เท่ากัน (!=)

```
In [27]: 1 a = 5  
          2 b = 'name'
```

```
In [28]: 1 a != 5
```

Out[28]: False

```
In [29]: 1 a != 7
```

Out[29]: True

```
In [30]: 1 b != 'name'
```

Out[30]: False

```
In [31]: 1 b != 'fullname'
```

Out[31]: True

COMPARISON OPERATORS

2.3 ตรวจสอบความมากกว่า (>)

In [32]: ➜ 1 a = 5

In [33]: ► 1 a > 4

Out[33]: True

In [34]: ➜ 1 a > 5

Out[34]: False

In [35]: ➜ 1 a > 6

Out[35]: False

2.4 ទ្រង់សុបគាមន៉ែយកវា (<)

In [36]: ► 1 a = 5

In [37]: ► 1 a < 4

Out[37]: False

In [38]: ► 1 a < 5

Out[38]: False

In [39]: ► 1 a < 6

Out[39]: True

COMPARISION OPERATORS

2.5 ตรวจสอบความมากกว่าหรือเท่ากับ (\geq)

```
In [40]: ┏━ 1 | a = 5
```

```
In [41]: ┏━ 1 | a >= 4
```

Out[41]: True

```
In [42]: ┏━ 1 | a >= 5
```

Out[42]: True

```
In [43]: ┏━ 1 | a >= 6
```

Out[43]: False

2.6 ตรวจสอบความน้อยกว่าหรือเท่ากับ (\leq)

```
In [44]: ┏━ 1 | a = 5
```

```
In [45]: ┏━ 1 | a <= 4
```

Out[45]: False

```
In [46]: ┏━ 1 | a <= 5
```

Out[46]: True

```
In [47]: ┏━ 1 | a <= 6
```

Out[47]: True

LOGICAL OPERATORS

3.1 and

3.2 or

3.3 not

3.1 and

p	q	p and q
True	True	True
True	False	False
False	True	False
False	False	False

In [48]: ➔ 1 t = True
2 f = False

In [49]: ➔ 1 t and f

Out[49]: False

LOGICAL OPERATORS

3.2 or

p	q	p or q
True	True	True
True	False	True
False	True	True
False	False	False

```
In [50]: ➔ 1 t = True  
          2 f = False
```

```
In [51]: ➔ 1 t or f
```

Out[51]: True

LOGICAL OPERATORS

3.3 not

p	not p
True	False
False	True

In [52]: ➔ 1 t = True
2 f = False

In [53]: ➔ 1 not t

Out[53]: False

In [54]: ➔ 1 not f

Out[54]: True

COMPARISON OPERATORS

4.1 in

```
In [55]: 1 listA = [1, 2, 3, 4]
          2 tupleA = ('a', 'b', 'c', 'd')
```

```
In [56]: 1 1 in listA
```

Out[56]: True

```
In [57]: 1 1 in tupleA
```

Out[57]: False

```
In [58]: 1 dictA = {'name' : 'John', 'age' : 32}
          2 setA = {'Sushi', 'Salmon'}
```

```
In [59]: 1 'name' in dictA
```

Out[59]: True

```
In [60]: 1 'Egg' in setA
```

Out[60]: False

COMPARISON OPERATORS

4.2 not in

```
In [61]: ► 1 listA = [1, 2, 3, 4]
          2 tupleA = ('a', 'b', 'c', 'd')
```

```
In [62]: ► 1 1 not in listA
```

Out[62]: False

```
In [63]: ► 1 1 not in tupleA
```

Out[63]: True

```
In [64]: ► 1 dictA = {'name' : 'John', 'age' : 32}
          2 setA = {'Sushi', 'Salmon'}
```

```
In [65]: ► 1 'name' not in dictA
```

Out[65]: False

```
In [66]: ► 1 'Egg' not in setA
```

Out[66]: True

FLOWCHART



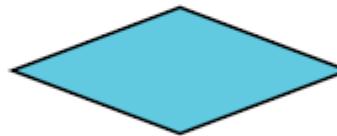
เริ่ม, จบ



พิมพ์



การประกาศตัวแปร
การคำนวณ

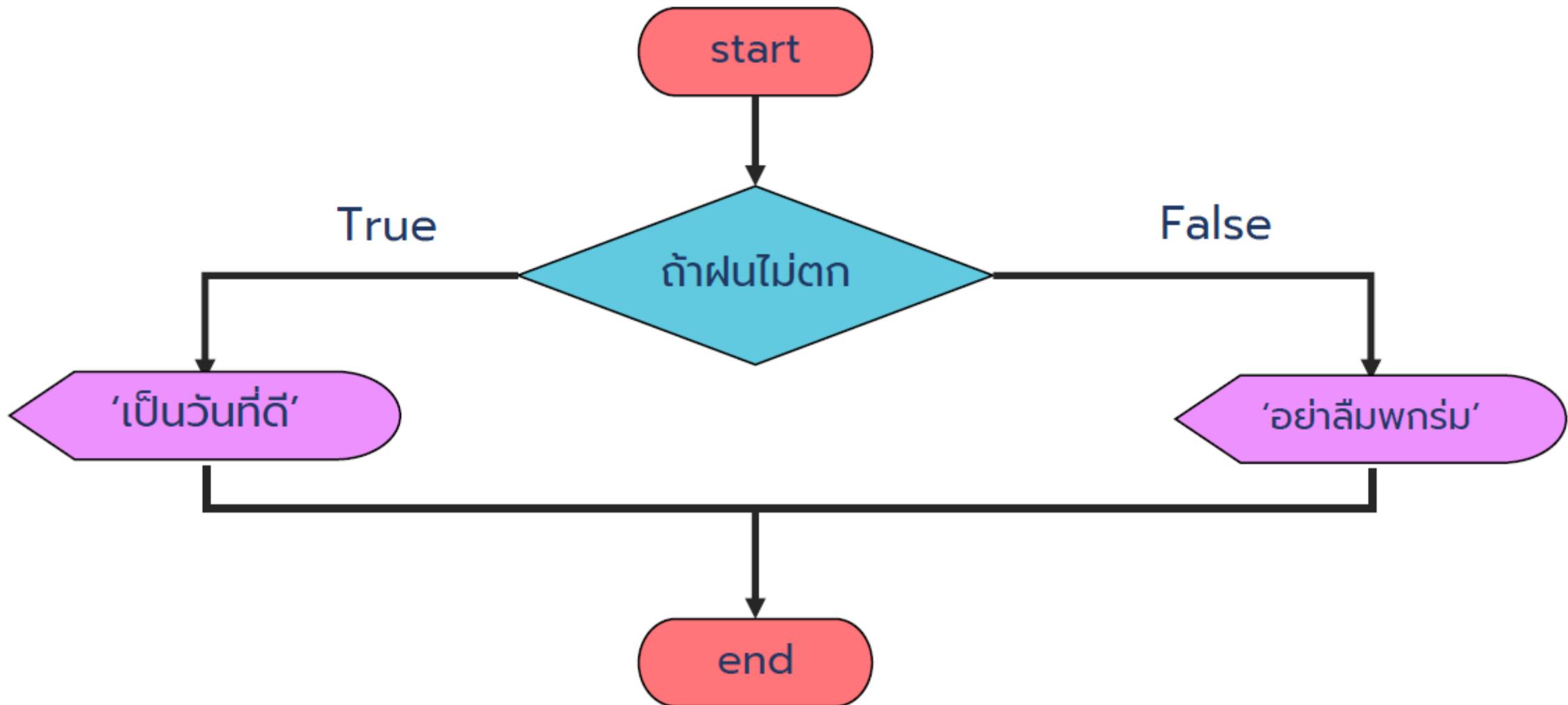


เงื่อนไข
(if-else)



รับอินพุต

FLOWCHART



คำสั่งเงื่อนไข

คำสั่งเงื่อนไข if

การทำงานของคำสั่งเงื่อนไข if มีรูปแบบดังต่อไปนี้

if เงื่อนไข :

 คำสั่งที่ 1

 คำสั่งที่ 2

```
age = input("กรอกอายุ: ")
if int(age) < 20:
    print("เด็กๆจ้า")
print("อายุ", age, "ปี")
```

กรอกอายุ:

กรอกอายุ: 18

เด็กๆจ้า

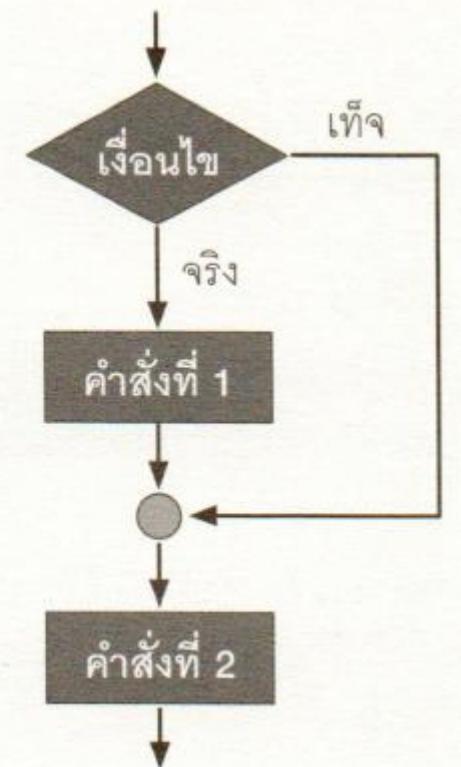
อายุ 18 ปี

กรอกอายุ: 21

อายุ 21 ปี

คำสั่ง if ใช้ตรวจสอบเงื่อนไขหนึ่ง ๆ หากเงื่อนไขเป็นจริงแล้วคำสั่งภายในบล็อกของเงื่อนไข if จะถูกประมวลผล ในที่นี่ คือ คำสั่งที่ 1

แต่ถ้าเงื่อนไขเป็นเท็จแล้วคำสั่งที่อยู่ภายใต้บล็อกของเงื่อนไข if จะไม่ได้รับการประมวลผล คือ จะข้ามไปประมวลผลคำสั่งที่อยู่นอกบล็อกของ if ทันที ในที่นี่ คือ คำสั่งที่ 2



คำสั่งเงื่อนไข

➤ คำสั่งเงื่อนไข if

```
In [1]: 1 name = input('Please insert your name: ')
2 print()
3 if name == 'John':
4     print('Hi John')
5 print('Nice to meet you')
```

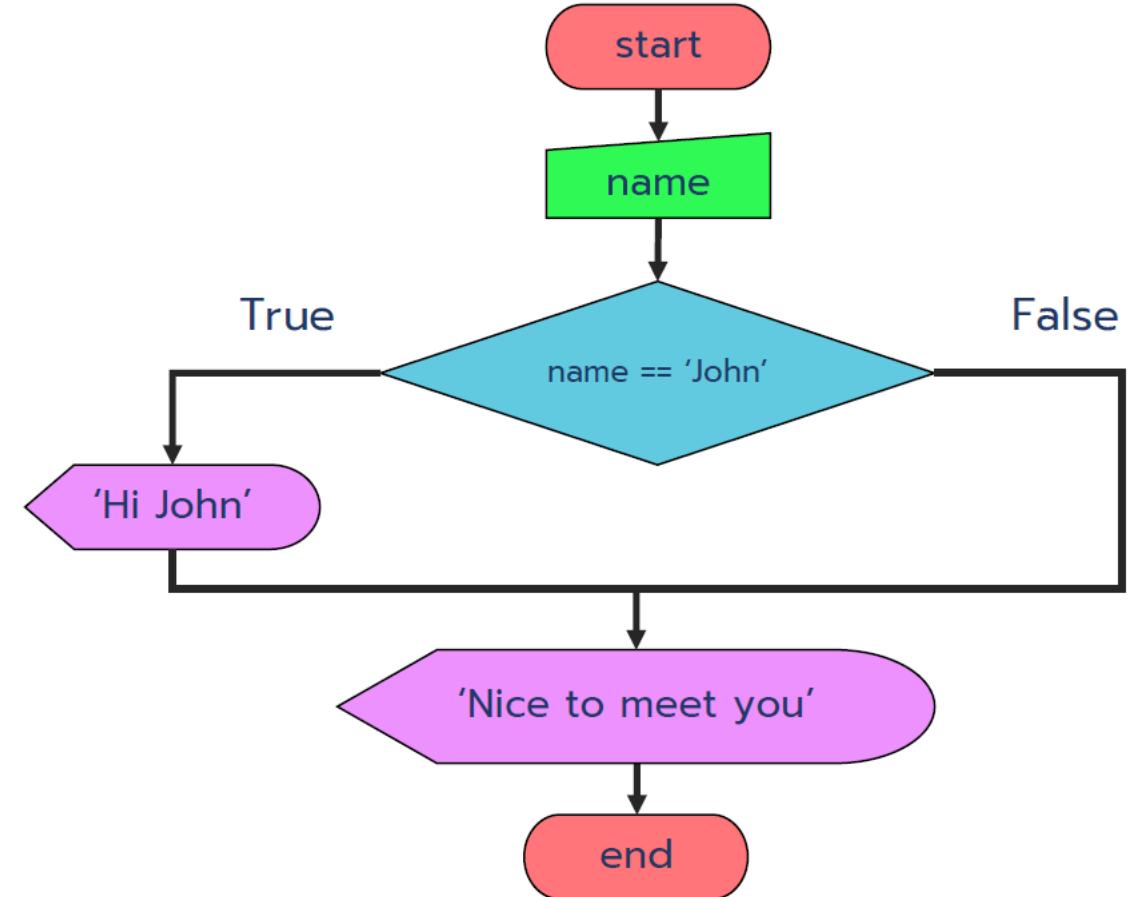
Please insert your name: John

Hi John
Nice to meet you

```
In [2]: 1 name = input('Please insert your name: ')
2 print()
3 if name == 'John':
4     print('Hi John')
5 print('Nice to meet you')
```

Please insert your name: Jane

Nice to meet you



คำสั่งเงื่อนไข

➤ คำสั่งเงื่อนไข if-else

การทำงานของคำสั่งเงื่อนไข if-else มีรูปแบบดังต่อไปนี้

if เงื่อนไข :

 คำสั่งที่ 1

else :

 คำสั่งที่ 2

 คำสั่งที่ 3

```
In [29]: guess = input("กรุณายกตัวเลข (1-10): ")
if int(guess) == 3:
    print("Right")
else:
    print("wrong")
print("bye bye")
```

กรุณายกตัวเลข (1-10): 4
wrong
bye bye

➤ if-elif-else

if เงื่อนไขที่ 1 :

 คำสั่งที่ 1

elif เงื่อนไขที่ 2 :

 คำสั่งที่ 2

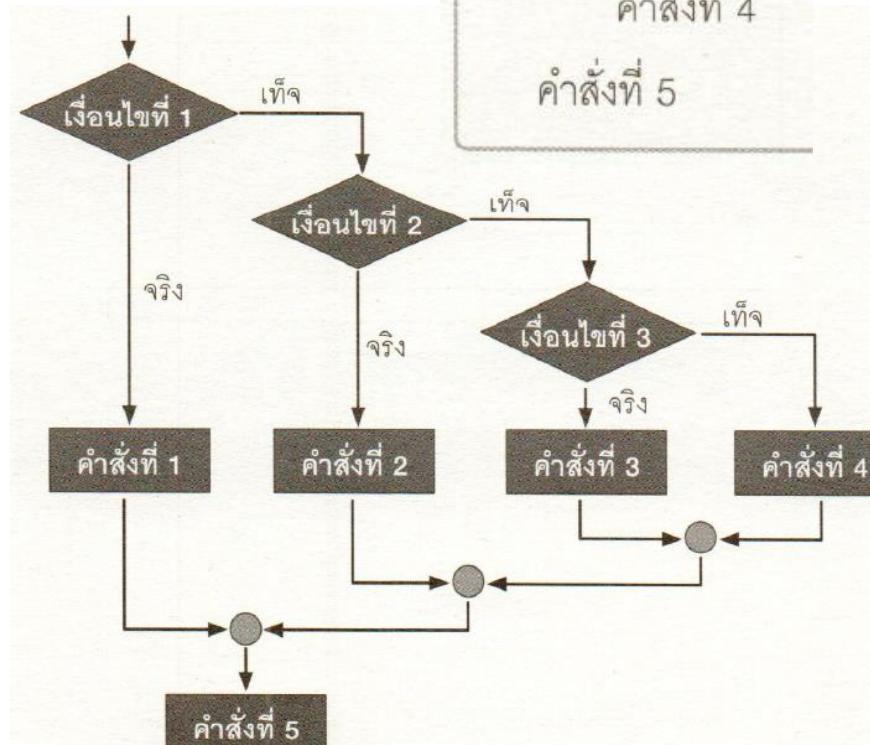
elif เงื่อนไขที่ 3 :

 คำสั่งที่ 3

else :

 คำสั่งที่ 4

 คำสั่งที่ 5



Workshop

ตัวอย่างที่ 3.24

```
score = input("กรุณากรอกคะแนน : ")
if int(score) >= 80 :
    print("คุณเก่งมากเลย")
    print("คุณได้เกรด A")
elif int(score) >= 70 :
    print("คุณได้เกรด B")
elif int(score) >= 60 :
    print("คุณได้เกรด C")
elif int(score) >= 50 :
    print("คุณได้เกรด D")
else :
    print("คุณได้เกรด F")
    print("พยายามใหม่นะ")
    print("แล้วพบกันใหม่เหมือนหน้า")
```

คำสั่งเงื่อนไข

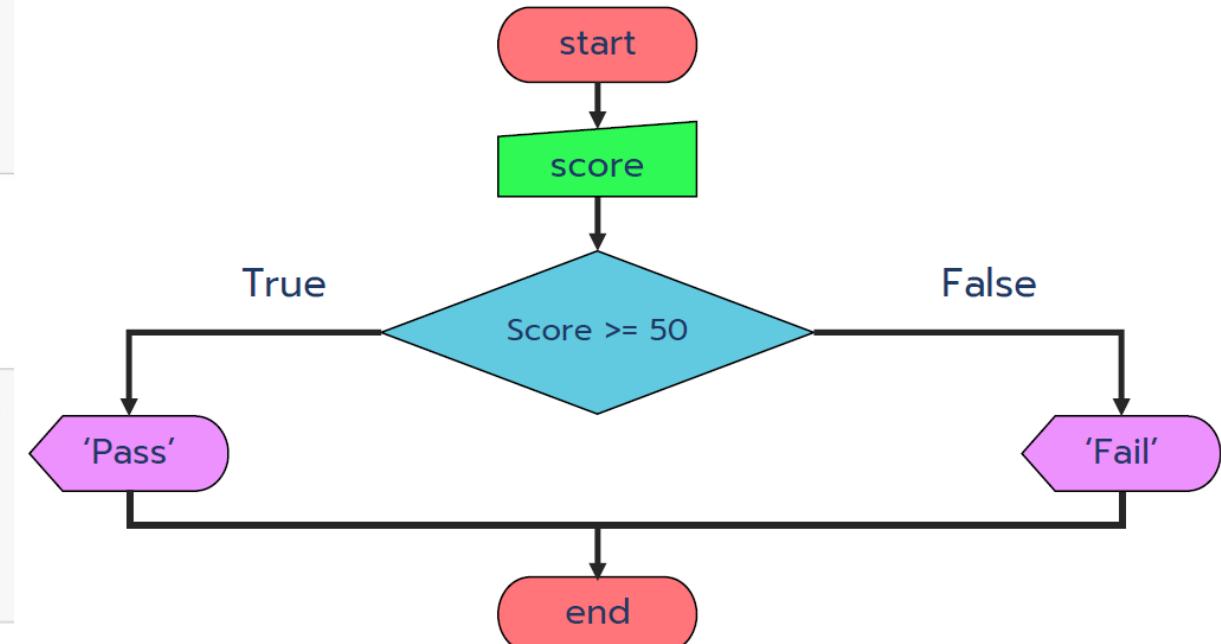
➤ คำสั่งเงื่อนไข if-else

```
In [3]: 1 score = int(input('Please insert score: '))
2 if score >= 50:
3     print('Pass')
4 else:
5     print('Fail')
```

```
Please insert score: 50
Pass
```

```
In [4]: 1 score = int(input('Please insert score: '))
2 if score >= 50:
3     print('Pass')
4 else:
5     print('Fail')
```

```
Please insert score: 49
Fail
```



คำสั่งเงื่อนไข

➤ คำสั่งเงื่อนไข if-elif

In [5]:

```
1 name = input('Please insert your name: ')
2 print()
3 if name == 'Alan Turing':
4     print('Oh God !')
5 elif name == 'Isac Newton':
6     print('OMG !')
7 print('Glad to see you')
```

Please insert your name: Alan Turing

Oh God !

Glad to see you

In [6]:

```
1 name = input('Please insert your name: ')
2 print()
3 if name == 'Alan Turing':
4     print('Oh God !')
5 elif name == 'Isac Newton':
6     print('OMG !')
7 print('Glad to see you')
```

Please insert your name: Isac Newton

OMG !

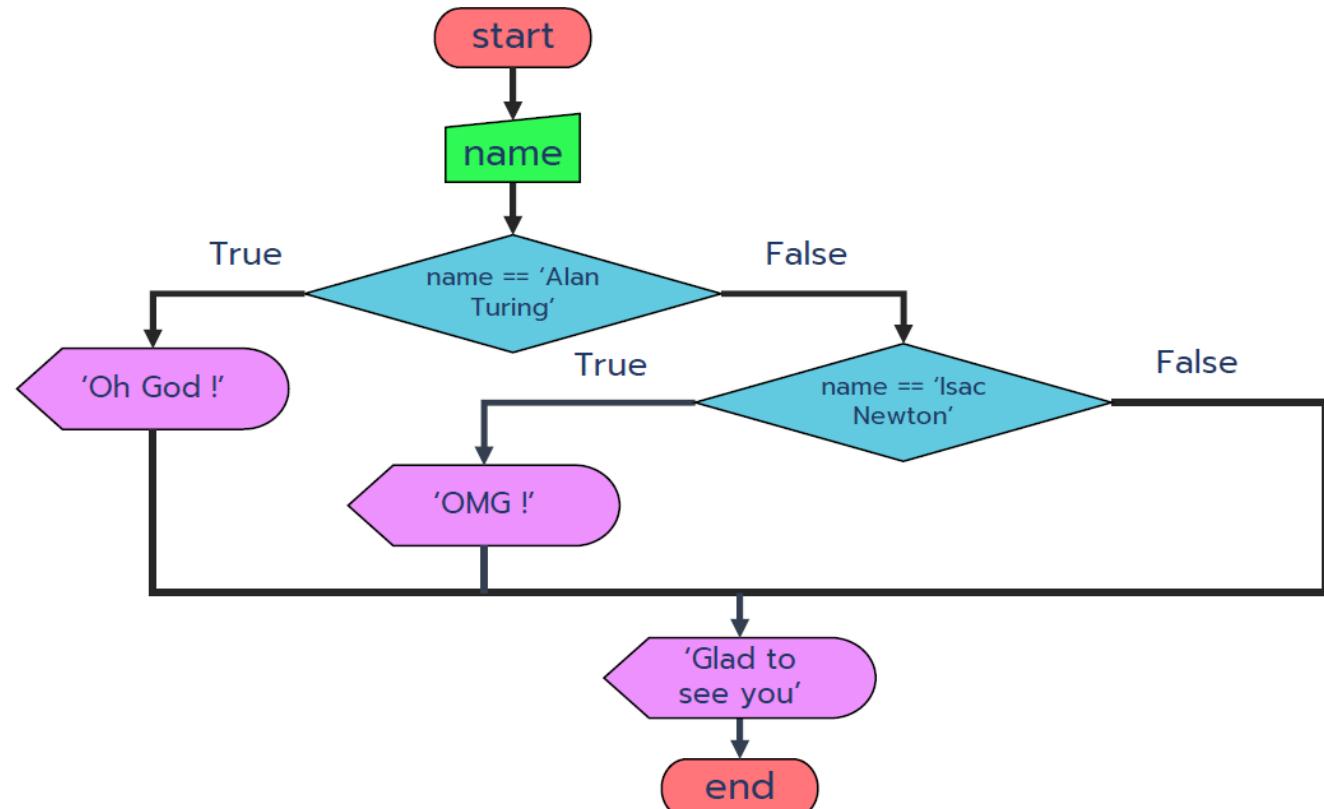
Glad to see you

In [7]:

```
1 name = input('Please insert your name: ')
2 print()
3 if name == 'Alan Turing':
4     print('Oh God !')
5 elif name == 'Isac Newton':
6     print('OMG !')
7 print('Glad to see you')
```

Please insert your name: Yoda

Glad to see you



คำสั่งเงื่อนไข

➤ คำสั่งเงื่อนไข if-elif-else

```
In [8]: 1 name = input('Please insert your name: ')
2 print()
3 if name == 'Gift':
4     print('I miss you')
5 elif name == 'Cherry':
6     print('I love you')
7 elif name == 'Wine':
8     print('I need you')
9 else:
10    print('Leave me alone')
```

Please insert your name: Gift

I miss you

```
In [9]: 1 name = input('Please insert your name: ')
2 print()
3 if name == 'Gift':
4     print('I miss you')
5 elif name == 'Cherry':
6     print('I love you')
7 elif name == 'Wine':
8     print('I need you')
9 else:
10    print('Leave me alone')
```

Please insert your name: Cherry

I love you

In [10]:

```
1 name = input('Please insert your name: ')
2 print()
3 if name == 'Gift':
4     print('I miss you')
5 elif name == 'Cherry':
6     print('I love you')
7 elif name == 'Wine':
8     print('I need you')
9 else:
10    print('Leave me alone')
```

Please insert your name: Wine

I need you

In [11]:

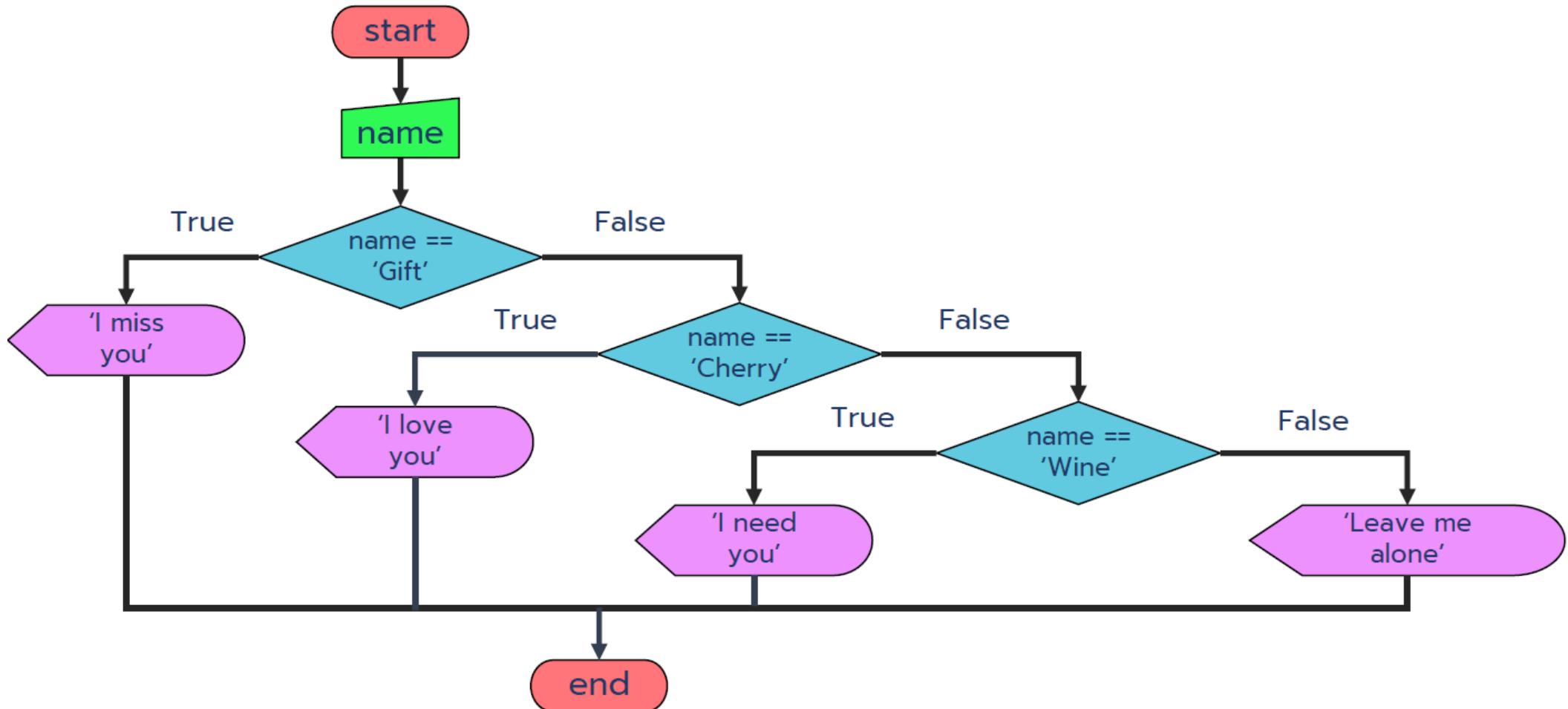
```
1 name = input('Please insert your name: ')
2 print()
3 if name == 'Gift':
4     print('I miss you')
5 elif name == 'Cherry':
6     print('I love you')
7 elif name == 'Wine':
8     print('I need you')
9 else:
10    print('Leave me alone')
```

Please insert your name: Longan

Leave me alone

คำสั่งเงื่อนไข

➤ คำสั่งเงื่อนไข if-elif-else



คำสั่งเงื่อนไข

> Nested-if

คำสั่ง

Nested-if

เป็นการนำ

คำสั่ง if

มาช้อนกัน

หลายชั้น

if เงื่อนไขที่ 1 :

 if เงื่อนไข 1.1 :

 คำสั่งที่ 1

 else

 คำสั่งที่ 2

 elif เงื่อนไขที่ 2 :

 if เงื่อนไข 2.1 :

 คำสั่งที่ 3

 elif เงื่อนไขที่ 2.2 :

 คำสั่งที่ 4

 else :

 คำสั่งที่ 5

else :

 คำสั่งที่ 6

 คำสั่งที่ 7

Workshop

ตัวอย่างต่อไปนี้จะพิจารณาเกรดตามคะแนนดังนี้

มากกว่าหรือเท่ากับ 80 คะแนน	เกรด A
75-79 คะแนน	เกรด B+
70-74 คะแนน	เกรด B
65-69 คะแนน	เกรด C+
60-64 คะแนน	เกรด C
55-59 คะแนน	เกรด D+
50-54 คะแนน	เกรด D
น้อยกว่า 50 คะแนน	เกรด F

ตัวอย่างที่ 3.25

```
score = input("กรุณารอกระดับคะแนน : ")
if int(score) >= 80 :
    print("คุณได้เกรด A")
elif int(score) >= 70 :
    if int(score) >= 75 :
        print("คุณได้เกรด B+")
    else :
        print("คุณได้เกรด B")
elif int(score) >= 60 :
    if int(score) >= 65 :
        print("คุณได้เกรด C+")
    else :
        print("คุณได้เกรด C")
elif int(score) < 60 :
    if int(score) >= 55 :
        print("คุณได้เกรด D+")
    elif int(score) < 55 and int(score) >= 50 :
        print("คุณได้เกรด D")
    else :
        print("คุณได้เกรด F")
else :
    print("พยายามใหม่นะ")
print("แล้วพบกันใหม่เหมือนหน้า")
```

คำสั่งทำซ้ำ > for

- มีจำนวนรอบการทำงานที่แน่นอน
- ใช้เรียกดึงค่าข้อมูลใน string, list, tuple, set, dictionary

```
for ตัวแปร in string/list/tuple/set/dictionary :  
    คำสั่งการทำงาน
```

ตัวอย่างที่ 3.29

```
mystr = "Hello"  
  
for x in mystr :  
    print(x)
```

การทำงานของคำสั่ง for ร่วมกับฟังก์ชัน range()

เราสามารถนำฟังก์ชัน range() มาใช้กำหนดรอบการทำงานของคำสั่งทำซ้ำ for ได้ เพื่อกำหนดว่าให้คำสั่ง for วนซ้ำการทำงานทั้งหมดกี่รอบ โดยรอบการทำงานจะเริ่มต้นที่ 0 และเพิ่มค่าขึ้นทีละ 1 ค่า ซึ่งจะวนซ้ำการทำงานทั้งหมดเท่ากับ (จำนวนรอบการทำงาน-1) รอบ ดังรูปแบบต่อไปนี้

```
for ตัวแปร in range(จำนวนรอบการทำงาน) :  
    คำสั่งการทำงาน
```

ทั้งนี้สามารถกำหนดได้ว่าจะให้การทำงานเริ่มต้นที่ค่าใด ให้ทำงานกี่รอบ และแต่ละรอบเพิ่มค่าเท่าไร ดังรูปแบบต่อไปนี้

```
for ตัวแปร in range(ค่าเริ่มต้น,จำนวนรอบการทำงาน) :  
    คำสั่งการทำงาน
```

ตัวอย่างที่ 3.30

```
i = 1  
  
for x in range(5) :  
    print(i)  
    i+=1  
  
print("")  
  
for x in range(1,10,2) :  
    print(x,end=" ")
```

คำสั่งทำซ้ำ > for

การทำงานของคำสั่ง for ร่วมกับคำสั่ง break และ continue

เมื่อพับคำสั่ง break จะทำให้หยุดการทำงานของคำสั่งทำซ้ำ for ทันที และไปทำงานคำสั่งอื่น ๆ นอกคำสั่งทำซ้ำ for ต่อไป

เมื่อพับคำสั่ง continue จะทำให้หยุดการทำงานของคำสั่งทำซ้ำ for ในรอบการทำงานนั้น และไปทำงานในรอบต่อไปทันที

ตัวอย่างที่ 3.31

```
for x in range(5) :  
    if x==2 :  
        continue  
    elif x==4 :  
        break  
    print(x)  
print("จบการทำงานของคำสั่ง for")
```

การทำงานของคำสั่งทำซ้ำ for ร่วมกับคำสั่ง else

คำสั่งต่าง ๆ ภายในตัวล็อกของคำสั่ง else จะถูกเรียกให้ทำงานเมื่อจบการทำงานของคำสั่งทำซ้ำ for

ตัวอย่างที่ 3.32

```
color = ["แดง", "เขียว", "น้ำเงิน"]  
  
for x in color :  
    print(x)  
else :  
    print("จบการแสดงผลสี")
```

คำสั่งทำซ้ำ ➤ For

1.1 Standard For

1.2 For + String

1.3 For + List

1.4 For + Tuple

1.5 For + Dictionary

1.6 For + Set

1.1 Standard For

1.1.1 for i in range (end)

1.1.2 for i in range (start, end)

1.1.3 for i in range (start, end, step)

คำสั่งทำซ้ำ ➤ For

1.1.1 for i in range (end)

```
In [1]: ┏━ 1 for i in range(5):  
      2     print(i)
```

```
0  
1  
2  
3  
4
```

1.1.2 for i in range (start, end)

```
In [2]: ┏━ 1 for i in range(5, 11):  
      2     print(i)
```

```
5  
6  
7  
8  
9  
10
```

1.1.3 for i in range (start, end, step)

```
In [3]: ┏━ 1 for i in range(10, 20, 2):  
      2     print(i)
```

```
10  
12  
14  
16  
18
```

คำสั่งทำซ้ำ ➤ For

1.2 For + String

```
In [4]: ┏━ 1 char = 'Python'  
      2 n = len(char)  
      3 for i in range(n):  
      4     print(char[i])
```

P
y
t
h
o
n

```
In [5]: ┏━ 1 char = 'Python'  
      2 for i in range(len(char)):  
      3     print(char[i])
```

P
y
t
h
o
n

```
In [6]: ┏━ 1 char = 'Python'  
      2 for c in char:  
      3     print(c)
```

P
y
t
h
o
n

```
In [7]: ┏━ 1 char = 'Python'  
      2 for i,c in enumerate(char):  
      3     print(i, c)
```

0 P
1 y
2 t
3 h
4 o
5 n

คำสั่งทำซ้ำ ➤ For

1.3 For + List

In [8]: ►

```
1 listx = ['a', 'b', 'c', 'd', 'e']
2 n = len(listx)
3 for i in range(n):
4     print(listx[i])
```

```
a
b
c
d
e
```

In [9]: ►

```
1 listx = ['a', 'b', 'c', 'd', 'e']
2 for i in range(len(listx)):
3     print(listx[i])
```

```
a
b
c
d
e
```

In [10]: ►

```
1 listx = ['a', 'b', 'c', 'd', 'e']
2 for x in listx:
3     print(x)
```

```
a
b
c
d
e
```

In [11]: ►

```
1 listx = ['a', 'b', 'c', 'd', 'e']
2 for i, x in enumerate(listx):
3     print(i, x)
```

```
0 a
1 b
2 c
3 d
4 e
```

คำสั่งทำซ้ำ > For

1.4 For + Tuple

```
In [12]: 1 tuplex = ('a', 'b', 'c', 'd', 'e')
          2 n = len(tuplex)
          3 for i in range(n):
          4     print(tuplex[i])
```

a
b
c
d
e

```
In [13]: 1 tuplex = ('a', 'b', 'c', 'd', 'e')
          2 for i in range(len(tuplex)):
          3     print(tuplex[i])
```

a
b
c
d
e

```
In [14]: 1 tuplex = ('a', 'b', 'c', 'd', 'e')
          2 for x in tuplex:
          3     print(x)
```

a
b
c
d
e

```
In [15]: 1 tuplex = ('a', 'b', 'c', 'd', 'e')
          2 for i, x in enumerate(tuplex):
          3     print(i, x)
```

0 a
1 b
2 c
3 d
4 e

คำสั่งทำชา > For

1.5 For + Dictionary

```
In [16]: 1 dictx = {'firstname' : 'John', 'lastname' : 'Doe', 'age' : 32}
          2 for key in dictx:
          3     print(key)
```

firstname
lastname
age

In [17]:

```
1 dictx = {'firstname' : 'John', 'lastname' : 'Doe', 'age' : 32}
          2 for key in dictx.keys():
          3     print(key)
```

firstname
lastname
age

```
In [18]: 1 dictx = {'firstname' : 'John', 'lastname' : 'Doe', 'age' : 32}
          2 for value in dictx.values():
          3     print(value)
```

John
Doe
32

คำสั่งทำซ้ำ ➤ For

1.6 For + Set

In [19]: ➤

```
1 setx = {1, 2, 3, 'a', 'b', 'c'}
2 for x in setx:
3     print(x)
```

b
1
2
3
a
c

คำสั่งทำซ้ำ

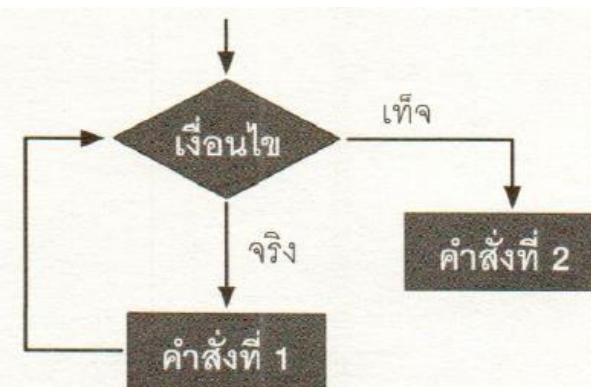
> while

คำสั่ง while จะตรวจสอบเงื่อนไขก่อนเสมอ หากเงื่อนไขเป็นจริงจึงทำคำสั่งที่ 1 จากนั้นจะวนซ้ำไปตรวจสอบเงื่อนไขอีกรอบ และทำงานไปเรื่อยๆ จนกว่าเงื่อนไขจะเป็นเท็จ จึงหยุดการทำงานของลูป while และไปทำงานคำสั่งนอกลูป while คือ คำสั่งที่ 2 ทันที

while เงื่อนไข :

คำสั่งที่ 1

คำสั่งที่ 2



ตัวอย่างที่ 3.26

```
i = 1
while i<=3 :
    print(i)
    i = i+1
```

ผลลัพธ์

1
2
3

การทำงานของคำสั่ง while ร่วมกับคำสั่ง break และ continue

เมื่อพบคำสั่ง break จะทำให้หยุดการทำงานของคำสั่งทำซ้ำ while ทันที และไปทำงานคำสั่งอื่นๆ นอกคำสั่งทำซ้ำ while ต่อไป แต่ถ้าพบคำสั่ง continue จะทำให้หยุดการทำงานของคำสั่งทำซ้ำ while ในรอบการทำงานนั้นและไปทำงานในรอบต่อไปทันที

i = 0

```
while i<=10 :
    i = i+1
    if i == 3 :
        continue
    elif i == 5 :
        break
    print(i)
print("จบการทำงาน")
```

การทำงานของคำสั่ง while ร่วมกับคำสั่ง else

คำสั่งต่างๆ ภายใต้บล็อกของคำสั่ง else จะถูกเรียกให้ทำงานเมื่อเงื่อนไขของคำสั่งทำซ้ำ while เป็นเท็จ

ตัวอย่างที่ 3.28

```
i = 0
while i<10 :
    i += 2
    print(i)
else :
    print("จบการแสดงผลเลขคู่")
```

คำสั่งทำซ้ำ ➤ while

In [20]: ►

```
1 i = 1
2 while i <= 5:
3     print(i)
4     i = i + 1
```

```
1
2
3
4
5
```

In [22]: ►

```
1 i = 0
2 while True:
3     print(i)
4     if i == 5:
5         break
6     i = i + 1
```

```
0
1
2
3
4
5
```

In [21]: ►

```
1 i = 5
2 while i <= 15:
3     print(i)
4     i = i + 2
```

```
5
7
9
11
13
15
```

คำสั่งทำซ้ำ

➤ Loop & Statement

3.1 break

3.2 continue

3.3 pass

3.1 break

```
In [23]: 1 for i in range(100000):
          2     print(i)
          3     if i == 5:
          4         break
```

0
1
2
3
4
5

3.2 continue

In [24]:

```
1 for i in range(10):
      2     if i%2 == 0:
      3         continue
      4         print('hello')
      5     else:
      6         print(i)
```

1
3
5
7
9

3.3 pass

3.3 pass

In [25]:

```
1 for i in range(10):
      2     if i%2 == 0:
      3         pass
      4         print('hello')
      5     else:
      6         print(i)
```

hello
1
hello
3
hello
5
hello
7
hello
9

In [26]:

```
1 v_list = [120, 70, 90]
2 for v in v_list:
3     if v >= 120:
4         pass
5     else:
6         print('500 baht finepage x')
```

500 baht finepage x
500 baht finepage x

ฟังก์ชัน (FUNCTION)

➤ การสร้างและเรียกใช้ function

การสร้างฟังก์ชัน ทำได้ดังนี้

```
def ชื่อฟังก์ชัน(พารามิเตอร์ตัวที่ 1, พารามิเตอร์ตัวที่ 2, ..., พารามิเตอร์ตัวสุดท้าย):
    คำสั่งการทำงานต่างๆ ภายในฟังก์ชัน
    Return ค่าข้อมูล
```

การเรียกใช้งานฟังก์ชัน ทำได้ดังนี้

ชื่อฟังก์ชัน()

ตัวอย่างที่ 3.33

```
def printvalue():
    print("การบวกเลข")

def plus(num1, num2):
    print(str(num1) + " + "
          + str(num2) + " =", num1 + num2)

def plus_return_value(num1, num2):
    return num1 + num2

# เรียกใช้งานฟังก์ชันที่สร้างขึ้นเอง
printvalue()
plus(1, 2)
print("3 + 4 = ", plus_return_value(3, 4))

# เรียกใช้งานฟังก์ชันสำหรับชื่อ sum() ที่ไฟรอนจัดเตรียมไว้ให้
x = (5, 6)
print("5 + 6 = ", sum(x))
```

ฟังก์ชัน (FUNCTION)

➤ ทำไมต้องใช้ Function

Ex. **list1 = [1, 3, 4, 7, 11]**
list2 = [4, 3, 2, 1, 0]

- ⇒ อายากหาผลรวมของ list1
- ⇒ อายากหาผลรวมของ list2

Ex.

```
sum1 = 0
for i in range(len(list1)):
    sum1 = sum1+list1[i]
sum2
for i in range(len(list2)):
    sum2 = sum2+list2[i]
```

Ex. **list1 = [1, 3, 4, 7, 11]**
:
listN = [7, 5, 12, 4, 0]

- ⇒ อายากหาผลรวมของ list1
- ⇒ อายากหาผลรวมของ listN

Ex.

```
def Sum(list):
    sumx = 0
    for i in range(len(list)):
        sumx = sumx + list[i]
    return sumx
```

Ex.

Sum1 = Sum(list1)
sum2 = Sum(list2)
:
sumN = Sum(listN)

ช้าช้อน !
จะเขียนทำไม่หลายรอบ !

ฟังก์ชัน (FUNCTION)

1. With Parameter

```
In [1]: 1 def sum_list(listA):
2     sumx = 0
3     for i in range(len(listA)):
4         sumx = sumx + listA[i]
5     return sumx
```

```
In [2]: 1 list1 = [1, 2, 3, 4, 5]
2 list2 = [6, 7, 8, 9, 10]
```

```
In [3]: 1 sum_list(list1)
```

Out[3]: 15

```
In [4]: 1 sum_list(list2)
```

Out[4]: 40

```
In [9]: 1 def plus_minus(a, b, c):
2     c = a + b + c
3     d = a - b - c
4     return c, d
```

```
In [10]: 1 result1, result2 = plus_minus(1, 3, 5)
```

```
In [11]: 1 result1
```

Out[11]: 9

```
In [12]: 1 result2
```

Out[12]: -11

```
In [5]: 1 def plus(a, b):
2     c = a + b
3     return c
```

```
In [6]: 1 plus(1, 3)
```

Out[6]: 4

```
In [7]: 1 def minus(a, b):
2     d = a - b
3     return d
```

```
In [8]: 1 minus(1, 3)
```

Out[8]: -2

ฟังก์ชัน (FUNCTION)

2. Without Parameter

In [13]:

```
1 def tree():
2     print('''
3     ---1---
4     --121--
5     -12321-
6     1234321'''')
```

In [14]:

```
1 tree()
```

```
---1---
--121--
-12321-
1234321
```

In [15]:

```
1 def PYTHON():
2     print('''
3     --XXXXXX--XX----XX--XXXXXXXXXX--XX----XX----XXXXXX--XX----XX--
4     --XX----XX--XX----XX----XX----XX----XX----XX----XXX----XX--
5     --XX----XX--XX----XX----XX----XX----XX----XX----XX----XXXXX--XX--
6     --XXXXXX----XXXX----XX----XXXXXXXXXX--XX----XX--XX--XX--XX--XX--XX--
7     --XX----XX----XX----XX----XX----XX----XX----XX----XX----XX----XXXXX--XX--
8     --XX----XX----XX----XX----XX----XX----XX----XX----XX----XX----XX----XXX--XX--
9     --XX----XX----XX----XX----XX----XX----XX----XX----XX----XX----XXXXXX--XX----XX--'''')
```

In [16]:

```
1 PYTHON()
```

```
--XXXXXX--XX----XX--XXXXXXXXXX--XX----XX----XXXXXX--XX----XX--
--XX----XX--XX----XX----XX----XX----XX----XX----XX----XXX----XX--
--XX----XX--XX----XX----XX----XX----XX----XX----XX----XX----XXXXX--XX--
--XXXXXX----XXXX----XX----XXXXXXXXXX--XX----XX--XX--XX--XX--XX--XX--XX--
--XX----XX----XX----XX----XX----XX----XX----XX----XX----XX----XX----XX----XXXXX--XX--
--XX----XX----XX----XX----XX----XX----XX----XX----XX----XX----XX----XX----XX----XXXXX--XX--
--XX----XX----XX----XX----XX----XX----XX----XX----XX----XX----XX----XX----XX----XX--
```

OOP

1. OOP เป็นเทคนิคการเขียนโปรแกรมแบบหนึ่ง ที่มุ่งทุกอย่างเป็น Object
2. การเขียน Code แบบ OOP ทำให้ง่าย & เป็นระเบียบ ในการ พัฒนา/ปรับปรุง

1. Attribute

```
In [1]: ┏━ 1 class Warrior:  
      2     def __init__(self, HP, ATK, DEF):  
      3         self.HP = HP  
      4         self.ATK = ATK  
      5         self.DEF = DEF
```

```
In [2]: ┏━ 1 warrior1 = Warrior(100, 45, 25)
```

```
In [3]: ┏━ 1 print("warrior1's HP =", warrior1.HP)  
      2 print("warrior1's ATK =", warrior1.ATK)  
      3 print("warrior1's DEF =", warrior1.DEF)
```

```
warrior1's HP = 100  
warrior1's ATK = 45  
warrior1's DEF = 25
```

```
In [4]: ┏━ 1 warrior2 = Warrior(100, 35, 30)
```

```
In [5]: ┏━ 1 print("warrior2's HP =", warrior2.HP)  
      2 print("warrior2's ATK =", warrior2.ATK)  
      3 print("warrior2's DEF =", warrior2.DEF)
```

```
warrior2's HP = 100  
warrior2's ATK = 35  
warrior2's DEF = 30
```

In [6]:

```
1 class Car:  
2     def __init__(self, brand, model, color):  
3         self.brand = brand  
4         self.model = model  
5         self.color = color
```

In [7]:

```
1 car1 = Car('Toyota', 'Camry', 'Black')
```

In [8]:

```
1 print("car1's brand =", car1.brand)  
2 print("car1's model =", car1.model)  
3 print("car1's color =", car1.color)
```

```
car1's brand = Toyota  
car1's model = Camry  
car1's color = Black
```

In [9]:

```
1 car2 = Car('Honda', 'Civic', 'White')
```

In [10]:

```
1 print("car2's brand =", car2.brand)  
2 print("car2's model =", car2.model)  
3 print("car2's color =", car2.color)
```

```
car2's brand = Honda  
car2's model = Civic  
car2's color = White
```

2. Method

In [11]:

```
1 class Warrior:  
2     def __init__(self, HP, ATK, DEF):  
3         self.HP = HP  
4         self.ATK = ATK  
5         self.DEF = DEF  
6  
7     def training(self):  
8         self.HP = self.HP + 5  
9         self.ATK = self.ATK + 10  
10        self.DEF = self.DEF + 5
```

In [12]:

```
1 warrior1 = Warrior(100, 45, 25)  
2 print('HP = %d, ATK = %d, DEF = %d' %(warrior1.HP, warrior1.ATK, warrior1.DEF))  
3 warrior1.training()  
4 print('HP = %d, ATK = %d, DEF = %d' %(warrior1.HP, warrior1.ATK, warrior1.DEF))
```

```
HP = 100, ATK = 45, DEF = 25  
HP = 105, ATK = 55, DEF = 30
```

In [13]:

```
1 class Car:  
2     def __init__(self, brand, model, color):  
3         self.brand = brand  
4         self.model = model  
5         self.color = color  
6  
7     def set_color(self, color):  
8         self.color = color
```

In [14]:

```
1 car1 = Car('Toyota', 'Camry', 'Black')  
2 print('old color = %s' %car1.color)  
3 car1.set_color('Red')  
4 print('new color = %s' %car1.color)
```

```
old color = Black  
new color = Red
```

HOMEWORK

IF-ELSE : Write Code and Flowchart

1) เขียนโปรแกรมรับอินพุต 1 ตัวที่เป็นจำนวนเต็ม และตรวจสอบว่าจำนวนที่รับมาเป็นเลขคู่หรือเลขคี่

- ถ้าเป็นเลขคู่ให้พิมพ์ 'เลขคู่'
- ถ้าเป็นเลขคี่ให้พิมพ์ 'เลขคี่'

2) เขียนโปรแกรมรับอินพุต 1 ตัวที่เป็นจำนวนจริง และตรวจสอบว่าจำนวนที่รับมาก็ไม่มากกว่า 0, น้อยกว่า 0 หรือเท่ากับ 0

- ถ้ามีค่ามากกว่า 0 ให้พิมพ์ 'positive'
- ถ้ามีค่าน้อยกว่า 0 ให้พิมพ์ 'negative'
- ถ้ามีค่าเท่ากับ 0 ให้พิมพ์ 'zero'

HOMEWORK IF-ELSE : Write Code and Flowchart

3) เขียนโปรแกรมรับอินพุต 1 ตัวเป็นอุณหภูมิในหน่วย华renไฮต์ (จำนวนจริง) และตรวจสอบว่า จำนวนที่รับมา มีค่ามากกว่าหรือเท่ากับ 32 หรือไม่

- ถ้ามีค่ามากกว่าหรือเท่ากับ 32 ให้คำนวณอุณหภูมิในหน่วยองศา เชลเซียส และพิมพ์ค่าออกมา

$$C = \frac{(5 \times (F - 32))}{9}$$

- ถ้ามีค่าน้อยกว่า 32 ให้พิมพ์ ‘cold’

4) เขียนโปรแกรมรับอินพุต 2 ตัวที่เป็นจำนวนจริง และตรวจสอบว่าจำนวนที่รับมา ตัวใดมีค่ามากกว่า และพิมพ์จำนวนนั้นออกมา แต่ถ้าจำนวนทั้งสองมีค่าเท่ากันให้พิมพ์ ‘มีค่าเท่ากัน’

HOMEWORK

IF-ELSE : Write Code and Flowchart

- 5) เขียนโปรแกรมรับอินพุต 3 ตัวที่เป็นจำนวนจริง และตรวจสอบว่าผลบวกของจำนวนที่หนึ่งและจำนวนที่สองมากกว่าจำนวนที่สามหรือไม่
- ถ้ามากกว่าให้พิมพ์ ' $a + b > c$ '
 - ถ้าไม่มากกว่า ไม่ดำเนินการใด ๆ

HOMEWORK

IF-ELSE : Write Code and Flowchart

6)

คุณเป็นพิตเนสเทรนเนอร์ในยิมซึ่อดังแห่งหนึ่ง ในแต่ละวันมีคนมาขอคำ

ปรึกษาเบื้องต้นเกี่ยวกับการดูแลรักษาสุขภาพโดยคุณสามารถ
คุณชมดูเวลาไปกับ
การให้คำปรึกษาและรู้สึกเห็นอย โชคดีที่คุณมีความรู้ในการเขียนโปรแกรม
คุณจึงวางแผนที่จะสร้างระบบอัตโนมัติเพื่อให้คำปรึกษาเบื้องต้นกับลูกค้า
ที่มาใช้บริการยิม ซึ่งมีรายละเอียดดังนี้

- รับอินพุต 3 ตัวที่เป็นสายอักขระ, จำนวนจริง, และจำนวนเต็ม (gender, weight, height) เพื่อให้คำแนะนำสุขภาพโดยมีเงื่อนไขดังนี้
 - ถ้าเป็นเพศชาย ให้พิจารณาว่าน้ำหนัก (กิโลกรัม) หากกว่าส่วนสูง (เซนติเมตร) - 100 หรือไม่
 - ถ้ามากกว่าให้พิมพ์ 'ควรออกกำลังกาย'
 - ถ้าไม่มากกว่าให้พิมพ์ 'คุณผู้ชายหุ่นดีเยี่ยม'
 - ถ้าเป็นเพศหญิง ให้พิจารณาว่าน้ำหนัก (กิโลกรัม) หากกว่าส่วนสูง (เซนติเมตร) - 110 หรือไม่
 - ถ้ามากกว่าให้พิมพ์ 'ควรออกกำลังกาย'
 - ถ้าไม่มากกว่าให้พิมพ์ 'คุณผู้หญิงหุ่นดีเยี่ยม'

HOMEWORK

For & WHILE Loop : Write Code and Flowchart

- 1) เขียนโปรแกรมพิมพ์ 19, 18, 17, ..., 0 โดยการใช้ for loop หรือ while loop
- 2) เขียนโปรแกรมพิมพ์ 18, 16, 14, ..., 0 โดยการใช้ for loop หรือ while loop
- 3) เขียนโปรแกรมพิมพ์ -3, -2, -1, 0, ..., 9 โดยการใช้ for loop หรือ while loop
- 4) เขียนโปรแกรมพิมพ์ -9, 8, -7, ..., 0 โดยการใช้ for loop หรือ while loop
- 5) เขียนโปรแกรมรับค่า n และคำนวณค่าต่อไปนี้ และพิมพ์ออกมา $1 + 2 + 3 + \dots + n$
- 6) เขียนโปรแกรมรับค่า n และคำนวณค่าต่อไปนี้ และพิมพ์ออกมา $1 \times 2 \times 3 \times \dots \times n$
- 7) เขียนโปรแกรมรับค่า n และคำนวณค่าต่อไปนี้ และพิมพ์ออกมา $\frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \dots + \frac{1}{n}$
- 8) เขียนโปรแกรมรับค่า n และคำนวณค่าต่อไปนี้ และพิมพ์ออกมา $\frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \dots + \frac{1}{n^2}$
- 9) เขียนโปรแกรมรับค่า n และคำนวณค่าต่อไปนี้ และพิมพ์ออกมา $\sqrt{6 \times \left(\frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \dots + \frac{1}{n^2} \right)}$

THE END