Class period 3

บทที่ 3 โปรแกรมวนซ้ำและการใช้เงื่อนไขในภาษาไพธอน

Function_Loop_Condition 2

ตัวอย่าง LOOP การวนซ้ำ

```
for i in 'Thanapong':
    print(f'{i} -> /')
```

- หมายความว่า ให้วนลูปอ่านค่าสมาชิกใน string 'Thanapong' โดยแทนค่าสมาชิกที่อ่านทีละตัวด้วยตัวแปร i
- ภายในลูป นำตัวแปร i ใส่ใน f' {i} -> / ' และ print string

ตัวอย่าง LOOP การวนซ้ำ

- ผลลัพธ์จะได้ จะเห็นว่า loop ทำการ print string ปกติ ตามค่า i คือตัวอักษรที่อยู่ใน string ทีละตัวตามลำดับจากลำดับที่ 0 ไปจนถึงลำดับสุดท้าย
- `T -> /`
- `h -> /`
- `a -> /`
- `n -> /`
- `a -> /`
- `p -> /`
- `o -> /`
- `n -> /`
- `g -> /`

การสั่ง loop ให้ print แบบไม่เว้นบรรทัด

- โดยคำสั่ง print จะมี key argument และค่า default สำหรับตัวมันเองคือ end='\n'
- print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

- Prints the values to a stream, or to sys.stdout by **default**.
- Optional keyword arguments:
- file: a file-like object (stream); defaults to the current sys.stdout.
- sep: string inserted between values, default a space.
- **end: string appended after the last value, default a newline.**
- flush: whether to forcibly flush the stream.

ตัวอย่างการสั่ง loop ให้ print แบบไม่เว้นบรรทัด

- `end='\n'`
- \n ในภาษา python สำหรับการพิมพ์ string หมายถึง การเว้นบรรทัด
- ถ้าต้องการ print แบบไม่เว้นบรรทัด ให้ใส่ input กำหนดค่า end ด้วยค่าที่เราต้องการ

- `for i in 'Thanapong':`
- `    print(f'{i} -> / ' ,end = '')`
- กำหนด end = ‘’ คือไม่ใส่อะไรเลย ผลลัพธ์จะได้
- `T -> / h -> / a -> / n -> / a -> / p -> / o -> / n -> / g -> /`

ตัวอย่างการสั่ง loop ให้ print แบบไม่เว้นบรรทัด

- `for i in 'Thanapong':`
- `    print(f'{i} -> / ' ,end = ',')`
- กำหนด end = ‘,’ คือ ใส่สัญลักษณ์ , ผลลัพธ์จะได้
- `T -> / ,h -> / ,a -> / ,n -> / ,a -> / ,p -> / ,o -> / ,n -> / ,g -> / ,`

range() การสร้าง list ตัวเลขแบบอัตโนมัติ

- range() คือคำสั่งที่ใช้สร้าง list ของตัวเลข เช่น
- `range5_output = range(5)`
- `print(list(range5_output))`
- หมายความว่า ให้สร้าง list ตัวเลขจำนวน 5 ตัว เริ่มจาก 0 และเก็บไว้ในตัวแปร range5_output
- จากนั้น print ตัวแปร range5_output ในรูปแบบของ list
- ผลลัพธ์จะได้
- [0, 1, 2, 3, 4]

ตัวอย่างการใช้งาน range() สร้าง list ตัวเลขในการวนลูป

```
for i in range(100):
    print('งง',end=' ')
```

- หมายความว่า ให้วนลูป 100 รอบ โดยอ่านค่าสมาชิกใน range(100) ซึ่งคือ list 0 ถึง 99
- ภายในลูปให้ print('งง',end=' ')
- ผลลัพธ์จะได้
- งง

Key argument ของ range()

- **range(stop)** -> range object
- **range(start, stop[, step])** -> range object
- Return an object that produces a sequence of integers from start (inclusive) to stop (exclusive) by step. range(i, j) produces i, i+1, i+2, ..., j-1. start defaults to 0, and stop is omitted! range(4) produces 0, 1, 2, 3. These are exactly the valid indices for a list of 4 elements. When step is given, it specifies the increment (or decrement).

- range() สามารถกำหนดตัวเลขที่ต้องการ เริ่ม(start), หยุด(stop) และ step ได้

ตัวอย่างการใช้งาน range() ด้วย key argument start, stop, step

- `list(range(1,11))`
- หมายความว่า ใช้ range() กำหนดให้สร้าง list ตัวเลข เริ่มจาก 1 ถึง 10
- ผลลัพธ์จะได้
- `[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]`

- `list(range(-3,20,4))`
- หมายความว่า ใช้ range() กำหนดให้สร้าง list ตัวเลข เริ่มจาก -3 ถึง 19 และให้ step=4
- ผลลัพธ์จะได้
- `[-3, 1, 5, 9, 13, 17]`

loop in loop

- สามารถสร้าง loop ภายใน loop ได้
- โดยลำดับการทำงาน จะทำงานตาม การเว้นวรรค (indent) โดยส่วนประมวณผลใน loop1 จะทำงานก่อน
- จากนั้นเมื่อมีการเขียน loop2 ภายใน loop1 loop2 จะถูกนับว่าเป็นส่วนประมวณผลของ loop1
- ดังนั้น loop2 จะทำงานวนลูปของตัวเองจนจบทุกรอบก่อน ถึงนับเป็นวนลูป 1 รอบของ loop1

ตัวอย่าง loop in loop

```
for loop1 in  range(2,5): #(2, 3, 4)
    print(f'now loop1 = {loop1}')
    for loop2 in  range(1,13):
        print(loop1,' x ', loop1,' = ', loop1 * loop2)
    print('the inner loop is end’)
```

- หมายความว่า ให้วนลูปอ่านค่าสมาชิกใน range(2,5) ซึ่งคือ [2, 3, 4] ทีละตัวแทนค่าด้วยตัวแปร loop1
- ภายในลูปให้ print(f'now loop1 = {loop1}’) ต่อด้วย ลูปที่ 2
- โดยลูปที่ 2 ให้วนลูปอ่านค่าสมาชิกใน range(1,13) ซึ่งคือ list 1 ถึง 12 ทีละตัวแทนค่าด้วยตัวแปร loop2
- ภายในลูปที่ 2 ให้ print(loop1,' x ', loop1,' = ', loop1 * loop2) วนลูปที่ 2 จนครบ 12
- เมื่อจบลูปที่ 2 ให้ print('the inner loop is end’) จากนั้นก็จะกลับไปวนลูปใหม่จนกว่าจะครบตาม range(2,5)

ตัวอย่าง loop in loop ผลลัพธ์จะได้

- now name1 = 2
- 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
- the inner loop is end

- now name1 = 3
- 3 x 1 = 3
- 3 x 2 = 6
- 3 x 3 = 9
- 3 x 4 = 12
- 3 x 5 = 15
- 3 x 6 = 18
- 3 x 7 = 21
- 3 x 8 = 24
- 3 x 9 = 27
- 3 x 10 = 30
- 3 x 11 = 33
- 3 x 12 = 36
- the inner loop is end

- now name1 = 4
- 4 x 1 = 4
- 4 x 2 = 8
- 4 x 3 = 12
- 4 x 4 = 16
- 4 x 5 = 20
- 4 x 6 = 24
- 4 x 7 = 28
- 4 x 8 = 32
- 4 x 9 = 36
- 4 x 10 = 40
- 4 x 11 = 44
- 4 x 12 = 48
- the inner loop is end

loop in function

- สามารถวนลูปในฟังก์ชั่นได้ โดยใช้ การเว้นวรรค (indent) กำหนดลำดับการทำงาน

```
def print_feelings(I,friendS,feeling = 'คิดถึง'):
  for friend in friendS:
        print(f'{I} {feeling} {friend}’)
```

- หมายความว่า กำหนดฟังก์ชั่นชื่อ print_feelings
- มี input 3 อย่าง คือ I,friendS,feeling = 'คิดถึง’ (default)
- ภายใน def ให้วนลูปอ่านค่าสมาชิกในตัวแปร friendS (list) ทีละตัวแทนค่าด้วยตัวแปร friend
- ภายในลูป print(f'{I} {feeling} {friend}’)

ตัวอย่างการใช้งาน loop in function

- `list_friends = ['พี่บูม','ดายน์','แตงโม','ฝน','พี่เจ็ท','น้องแคมป์']`
- `print_feelings('เจมส์',list_friends)`

- กำหนดตัวแปร list_friends เป็นข้อมูล list มีสมาชิก ['พี่บูม','ดายน์','แตงโม','ฝน','พี่เจ็ท','น้องแคมป์']
- ใช้งานฟังก์ชั่น print_feelings('เจมส์',list_friends)
- ชื่อฟังก์ชั่นตามด้วยค่าที่ต้องการ Input ตามลำดับตัวแปรที่กำหนดไว้ตอนสร้างฟังก์ชั่น ‘เจมส์’ = I, list_friends = friendS และตัวแปร feeling ไม่ได้ Input ค่าดังนั้นจะถูกใช้ค่า default คือ 'คิดถึง’

ตัวอย่างการใช้งาน loop in function

- ผลลัพธ์ของ `print_feelings('เจมส์',list_friends)` ไม่มี input ค่าตัวแปร feeling จะได้
- เจมส์ คิดถึง พี่บูม
- เจมส์ คิดถึง ดายน์
- เจมส์ คิดถึง แตงโม
- เจมส์ คิดถึง ฝน
- เจมส์ คิดถึง พี่เจ็ท
- เจมส์ คิดถึง น้องแคมป์

```
list friends = ['พี่บูม','ดายน์','แตงโม','ฝน','พี่เจ็ท','น้องแคมป์']

def print_feelings(I,friendS,feeling = 'คิดถึง'):
  for friend in friendS:
        print(f'{I} {feeling} {friend}')
```

ตัวอย่างการใช้งาน loop in function

- ผลลัพธ์ของ `print_feelings('เจมส์',list_friends,'รัก')` จะได้
- เจมส์ รัก พี่บูม
- เจมส์ รัก ดายน์
- เจมส์ รัก แตงโม
- เจมส์ รัก ฝน
- เจมส์ รัก พี่เจ็ท
- เจมส์ รัก น้องแคมป์

```
list_friends = ['พี่บูม','ดายน์','แตงโม','ฝน','พี่เจ็ท','น้องแคมป์']

def print_feelings(I,friendS,feeling = 'คิดถึง'):
  for friend in friendS:
        print(f'{I} {feeling} {friend}')
```

conditional คือ เงื่อนไข /ประโยคเงื่อนไข

```
if condition1 :
    do something
elif condition2 :  #elif คือ else if
    do another thing
else:
    do ...
```

- สามารถกำหนดเงื่อนไขก่อนเข้าการทำงานของส่วนประมวณผล โดยถ้าผ่านเงื่อนไขของ if condition1 โค้ดประมวณผลภายใน if จะทำงาน
- ถ้าไม่ผ่านจะไปเงื่อนไขถัดไป elif condition2 ตามลำดับ (elif สามารถมีได้มากกว่า 1)
- แต่ถ้าหากไม่ผ่านเงื่อนไขใดเลย โค้ดประมวณผลของ else จะทำงาน

ตัวอย่าง Loop แบบปกติ

- `list friends = ['พี่บูม','ดายน์','แตงโม','ฝน','พี่เจ็ท','น้องแคมป์']`

- `for name1 in list_friends :`
- `    for name2 in list_friends :`
- `        print(name1,'รัก',name2)`

- วนลูป for ปกติเพื่อ print(name1,'รัก',name2) เมื่อได้ผลลัพธ์ออกมา สังเกตว่าจะมีชื่อซ้ำกันรักกันเอง

ผลลัพธ์จะได้

- พี่บูม รัก พี่บูม
- พี่บูม รัก ดายน์
- พี่บูม รัก แตงโม
- พี่บูม รัก ฝน
- พี่บูม รัก พี่เจ็ท
- พี่บูม รัก น้องแคมป์
- ดายน์ รัก พี่บูม
- ดายน์ รัก ดายน์
- ดายน์ รัก แตงโม
- ดายน์ รัก ฝน
- ดายน์ รัก พี่เจ็ท
- ดายน์ รัก น้องแคมป์
- แตงโม รัก พี่บูม
- แตงโม รัก ดายน์
- แตงโม รัก แตงโม
- แตงโม รัก ฝน
- แตงโม รัก พี่เจ็ท
- แตงโม รัก น้องแคมป์
- ฝน รัก พี่บูม
- ฝน รัก ดายน์
- ฝน รัก แตงโม
- ฝน รัก ฝน
- ฝน รัก พี่เจ็ท
- ฝน รัก น้องแคมป์
- พี่เจ็ท รัก พี่บูม
- พี่เจ็ท รัก ดายน์
- พี่เจ็ท รัก แตงโม
- พี่เจ็ท รัก ฝน
- พี่เจ็ท รัก พี่เจ็ท
- พี่เจ็ท รัก น้องแคมป์
- น้องแคมป์ รัก พี่บูม
- น้องแคมป์ รัก ดายน์
- น้องแคมป์ รัก แตงโม
- น้องแคมป์ รัก ฝน
- น้องแคมป์ รัก พี่เจ็ท
- น้องแคมป์ รัก น้องแคมป์

ตัวอย่างการใช้งาน conditional

```
for name1 in list_friends:
    for name2 in list_friends:
        if name1 != name2: #ไม่ปริ้นชื่อคนเดียวกันซ้ำ
    print(name1,'รัก',name2)
```

- สามารถเพิ่มเงื่อนไข if name1 != name2: ไปส่วนประมวณผลภายในลูป เมื่อผ่านเงื่อนไขนี้จึงจะไปทำงานส่วนประมวณผลภายใน if คือ print(name1,'รัก',name2)
- ดังนั้น ตามเงื่อนไข name1 ไม่เท่ากับ name2 โค้ดจะปริ้นแต่ชื่อที่ไม่ซ้ำกันเท่านั้น

ผลลัพธ์จะได้

- พี่บูม รัก ดายน์
- พี่บูม รัก แตงโม
- พี่บูม รัก ฝน
- พี่บูม รัก พี่เจ็ท
- พี่บูม รัก น้องแคมป์
- ดายน์ รัก พี่บูม
- ดายน์ รัก แตงโม
- ดายน์ รัก ฝน
- ดายน์ รัก พี่เจ็ท
- ดายน์ รัก น้องแคมป์
- แตงโม รัก พี่บูม
- แตงโม รัก ดายน์
- แตงโม รัก ฝน
- แตงโม รัก พี่เจ็ท
- แตงโม รัก น้องแคมป์
- ฝน รัก พี่บูม
- ฝน รัก ดายน์
- ฝน รัก แตงโม
- ฝน รัก พี่เจ็ท
- ฝน รัก น้องแคมป์
- พี่เจ็ท รัก พี่บูม
- พี่เจ็ท รัก ดายน์
- พี่เจ็ท รัก แตงโม
- พี่เจ็ท รัก ฝน
- พี่เจ็ท รัก น้องแคมป์
- น้องแคมป์ รัก พี่บูม
- น้องแคมป์ รัก ดายน์
- น้องแคมป์ รัก แตงโม
- น้องแคมป์ รัก ฝน
- น้องแคมป์ รัก พี่เจ็ท

ข้อจำกัดของการใช้งาน conditional


```
for name1 in list_friends:
    for name2 in list_friends:
        if name1 == name2:
            'do nothing'
        else:
            print(name1,'รัก',name2)
```

- ภายในเงื่อนไข จำเป็นต้องมีส่วนประมวณผล ไม่สามารถปล่อยว่างไว้ได้

สัญลักษณ์ที่ใช้ใน conditional

- == คือ เท่ากับ
- != คือ ไม่เท่ากับ
- >= คือ มากกว่าหรือเท่ากับ ใช้ในกรณีตรวจสอบตัวเลข
- <= คือ น้อยกว่าหรือเท่ากับ ใช้ในกรณีตรวจสอบตัวเลข
- < คือ น้อยกว่า ใช้ในกรณีตรวจสอบตัวเลข
- > คือ มากกว่า ใช้ในกรณีตรวจสอบตัวเลข

Ture False ใน conditional

- หากหลังเงื่อนไข(if) เป็น True จะทำงานส่วนประมวณผลภายใน if
- if True:
- print('Yes’)
- ผลลัพธ์จะได้
- Yes
- หากหลังเงื่อนไข(if) เป็น False โค้ดจะไม่เข้าไปทำงานส่วนประมวณผลภายใน if เลย ดังนั้น
- if False:
- print('No’)
- ผลลัพธ์จะได้
- ไม่ปริ้นอะไรออกมาเลย เพราะโค้ดไม่เข้าไปทำงานส่วนประมวณผลภายใน if หรือก็คือปล่อยผ่าน

ตัวอย่าง Ture False ใน conditional

- 'พี่บูม' == 'พี่เจ็ท’
- ผลลัพธ์จะได้
- False

- 'พี่บูม' != 'พี่เจ็ท’
- ผลลัพธ์จะได้
- True

Homework class period 3

- เขียน function วนลูปตัดเกรด โดยที่ input เป็นคะแนน(0-100) , output เป็นเกรด (F-A)
- วนลูปตัดเกรด input = [1,50,65,90,101,-5,49]
- กำหนด
- ถ้าคะแนนอยู่ระหว่างมากกว่าหรือเท่ากับ 0 แต่น้อยกว่า 50 จะได้เกรด F
- ถ้าคะแนนอยู่ระหว่างมากกว่าหรือเท่ากับ 50 แต่น้อยกว่า 55 จะได้เกรด D
- ถ้าคะแนนอยู่ระหว่างมากกว่าหรือเท่ากับ 55 แต่น้อยกว่า 60 จะได้เกรด D+
- ถ้าคะแนนอยู่ระหว่างมากกว่าหรือเท่ากับ 60 แต่น้อยกว่า 65 จะได้เกรด C
- ถ้าคะแนนอยู่ระหว่างมากกว่าหรือเท่ากับ 65 แต่น้อยกว่า 70 จะได้เกรด C+
- ถ้าคะแนนอยู่ระหว่างมากกว่าหรือเท่ากับ 70 แต่น้อยกว่า 75 จะได้เกรด B
- ถ้าคะแนนอยู่ระหว่างมากกว่าหรือเท่ากับ 75 แต่น้อยกว่า 80 จะได้เกรด B+
- ถ้าคะแนนอยู่ระหว่างมากกว่าหรือเท่ากับ 80 จะได้เกรด A
- แต่ว่าค่าคะแนนจะต้องไม่ต่ำกว่า 0 และมากกว่า 100 ไม่งั้นจะเกิด error