

w14-Lab1

# Iterations

## Part III

Written for 204111

by Thapanapong Rukkanchanunt

# More on Loops

- **Topic 1: Nested Loops vs Helper Function**
- **Topic 2: Variable Scope in Loops**

# 1: Nested Loop vs Helper Function

```

02 def nestedLoop():
03
04     ROW = 5
05     COLUMN = 4
06
07     for i in range(1, ROW + 1):
08         print("\ni is now %d" % i)
09
10         for j in range(1, COLUMN + 1):
11             print("j = %d" % j, end=" ")

```

```

i is now 1
j = 1 j = 2 j = 3 j = 4
i is now 2
j = 1 j = 2 j = 3 j = 4
i is now 3
j = 1 j = 2 j = 3 j = 4
i is now 4
j = 1 j = 2 j = 3 j = 4
i is now 5
j = 1 j = 2 j = 3 j = 4

```

Independent

# 1: Nested Loop vs Helper Function

```

02 def nestedLoop_helper(COLUMN):
03     for j in range(1, COLUMN + 1):
04         print("j = %d" % j, end=" ")
05
06 def nestedLoop():
07
08     ROW = 5
09     COLUMN = 4
10
11     for i in range(1, ROW + 1):
12         print("\ni is now %d" % i)
13
14         nestedLoop_helper(COLUMN)
15         # for j in range(1, COLUMN + 1):
16             # print("j = %d" % j, end=" ")

```

```

i is now 1
j = 1 j = 2 j = 3 j = 4
i is now 2
j = 1 j = 2 j = 3 j = 4
i is now 3
j = 1 j = 2 j = 3 j = 4
i is now 4
j = 1 j = 2 j = 3 j = 4
i is now 5
j = 1 j = 2 j = 3 j = 4

```

ใช้ helper function เพื่อลดความซ้ำซ้อน

Use helper function instead

# 1: Nested Loop vs Helper Function

ให้เขียนโปรแกรมภาษา Python เพื่อหาผลรวมของยอดขายทั้งเดือน  
ผลรวมของยอดขายต่อสัปดาห์ ให้สอดคล้องกับตัวอย่างการ run  
โปรแกรมด้านล่าง โดยที่ค่าข้อมูลให้นำเข้าผ่านทาง keyboard

## ตัวอย่าง Output

```
SUM of Week (1) = 12800.00
SUM of Week (2) = 13050.00
SUM of Week (3) = 13728.57
SUM of Week (4) = 14500.00
SUM for All = 13519.64
```

```
02 def nestedLoop():
03
04     WEEK = 4
05     DAY = 7
06     WEEK_TOTAL = 0
07     ALL_TOTAL = 0
08     for i in range(1, WEEK + 1):
09         for j in range(1, DAY + 1):
10             SALE = float(input(""))
11             WEEK_TOTAL += SALE
12             ALL_TOTAL += WEEK_TOTAL
13             print("SUM of Week (%d) = %.2f" % (i, WEEK_TOTAL))
14             WEEK_TOTAL = 0
15     print("SUM for all = %.2f" % (ALL_TOTAL))
```

Dependent

# 1: Nested Loop vs Helper Function

```

02 def nestedLoop_helper(DAY):
03     WEEK_TOTAL = 0
04     for j in range(1, DAY + 1):
05         SALE = float(input(""))
06         WEEK_TOTAL += SALE
07     return WEEK_TOTAL

```

## ตัวอย่าง Output

```

SUM of Week (1) = 12800.00
SUM of Week (2) = 13050.00
SUM of Week (3) = 13728.57
SUM of Week (4) = 14500.00
SUM for All = 13519.64

```

ส่ง return value กลับมา

```

09 def nestedLoop():

```

```

10

```

```

11 WEEK = 4

```

```

12 DAY = 7

```

```

13 WEEK_TOTAL = 0

```

```

14 ALL_TOTAL = 0

```

```

15 for i in range(1, WEEK + 1):

```

```

16     WEEK_TOTAL = nestedLoop_helper(DAY)

```

```

17     ALL_TOTAL += WEEK_TOTAL

```

```

18     print("SUM of Week (%d) = %.2f" % (i, WEEK_TOTAL))

```

```

19     # WEEK_TOTAL = 0 ← No need to reset

```

```

20 print("SUM for all = %.2f" % (ALL_TOTAL))

```

Use RETURN to pass variable from helper function

ต้องส่งตัวแปรกลับด้วย

## 2: Variable Scope in Loops

```
08 SIZE = 5
09
10 for i in range(SIZE):
11     for j in range(SIZE - i):
12         print(" ", end="")
13
14     for j in range(i + 1):
15         print("* ", end="")
16
17     print("")
```



```

      *
     * *
    * * *
   * * * *
  * * * * *
```

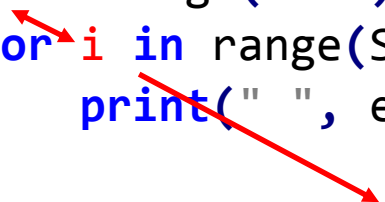
Variable can be reuse within the same level of loop without issue

## 2: Variable Scope in Loops

```

08 SIZE = 5
09
10 for i in range(SIZE):
11     for i in range(SIZE - i):
12         print(" ", end="")
13
14     for j in range(i + 1):
15         print("* ", end="")
16
17     print("")

```



```

      * * * * *
    * * * *
  * * *
* *
*

```

Be careful of using same variable between level of loop. In this example, the last value of *i* from the first inner loop will be used in the second inner loop instead of *i* from outer loop. Regardless, the outer loop still runs for 5 iterations.



## 2: Variable Scope in Loops

```

08 NUM = 25
09
10 for i in range(2, NUM):
11     if NUM%i == 0:
12         break
13 if i != NUM-1:
14     print("%d divides %d" % (i, NUM))

```

Loop variable can be used outside loop. However, be careful of the case where loop does 0 iteration.

5 divides 25

```

08 NUM = 2
09
10 for i in range(2, NUM):
11     if NUM%i == 0:
12         break
13 if i != NUM-1:
14     print("%d divides %d" % (i, NUM))

```

NameError: name 'i' is not defined

# Practice 1

ให้เขียนฟังก์ชัน `sum_parity_in_range(x, y, parity)` เพื่อคืน  
ค่าผลบวกของจำนวนที่มี **Parity** ตามที่กำหนดในช่วง  $x$  ถึง  $y$   
กำหนดให้ **Parity** = 0 สำหรับจำนวนคู่ และ 1 สำหรับจำนวนคี่

- ตัวอย่างการ run 1

```
Input first value: 7
Input last value: 21
Input parity: 0
Parity sum: 98
```

# Practice 2

ให้เขียนฟังก์ชัน `digit_count(x, d)` เพื่อคืนค่าจำนวนหลักของ  $x$  ที่มีค่าเท่ากับ  $d$

- ตัวอย่างการ run 1

```
Input x: 123456789
Input d: 2
Digit count: 1
```

- ตัวอย่างการ run 2

```
Input x: 111222333
Input d: 2
Digit count: 3
```

# Practice 3

ให้นำข้อมูลจำนวนแถว (row) และ พิมพ์ผลลัพธ์ดัง  
ตัวอย่างแสดงด้านล่าง

- ตัวอย่างการ run 1

Input row: 3

```
*
. *
* . *
```

- ตัวอย่างการ run 2

Input row: 4

```
*
. *
* . *
. * . *
```

# Practice 4

ให้เขียนโปรแกรมแสดงจำนวนตั้งแต่ 1 ถึง 1,000 ที่มีผลรวมทุกหลักมีค่าเท่ากับ  $n$  โดยเรียงจากมากไปน้อย

- ตัวอย่างการ run 1

Input n: 3

3 12 21 30 102 111 120 201 210 300

- ตัวอย่างการ run 2

Input n: 4

4 13 22 31 40 103 112 121 130 202 211 220 301 400

# Practice 5

ให้เขียนโปรแกรมแสดงจำนวนที่รับเข้ามาในรูปแบบ  
ของ **Mirror** ตามตัวอย่างด้านล่าง

- ตัวอย่างการ run 1

```
Input x: 123  
Mirror: 123321
```

- ตัวอย่างการ run 2

```
Input x: 3285  
Mirror: 32855823
```