2.14.2. For Loop

Fig. 2.14.2.1 Photo by Tine Ivanič on Unsplash

Note

Outline

1. For Loop
    a. Ex1: range()
    b. Ex2: Sum
    c. Ex3: From While Loop To For Loop
    d. Ex4: 6!
    e. Ex5: Split Me!
    f. Ex6: International Day of Mathematics
    g. Ex7: Star-typed Triangles
    h. ‘Ex8: Forever <#forever>`__
2. While Loop vs. For Loop
    a. Ex1: While vs. For: Star-typed Triangle

Note

Roadmap

1. This topic: Loop

> myMaze, myBoard, myBlock

<table>
<tr><td colspan="4">Loop</td></tr>
<tr><td rowspan="2">while Statement</td><td>for Statement</td><td rowspan="3">break Statement</td><td rowspan="3">continue Statement</td></tr>
<tr><td>Iterator</td></tr>
<tr><td>Nested while Loops</td><td>Nested for Loops</td></tr>
<tr><td colspan="4">Complex Loop</td></tr>
</table>

2. Course: Python 1
3. Subject: Programming
4. Field
    a. Software Engineering (SE)
    b. Computer Science and Information Engineering (CSIE)
    c. Electrical/Electronics Engineering (EE)

2.14.2.1. For Loop

1. for...in迴圈，一般也會簡稱為for迴圈，通常拿來處理有次序性且已知應該執行次數的問題。
2. 在執行for迴圈時需要下列三個條件，
    a. 設定控制變數的初始值。
    b. 設定控制變數的終止值，也就是迴圈的結束條件。
    c. 設定控制變數值的變動方向與量，也就是迴圈的疊代(Iteration)。
3. 一般來說，在for迴圈裡面的expression如果不是一個變數（如常用的i），就是一個變數序列（如list與tuple等）。
4. iterable是疊代的意思，在這裡要放置可以循序疊代的物件，例如list、tuple或range()函數。
5. 而else是選擇性設定，後面的程式碼會在for迴圈中止之後被執行。

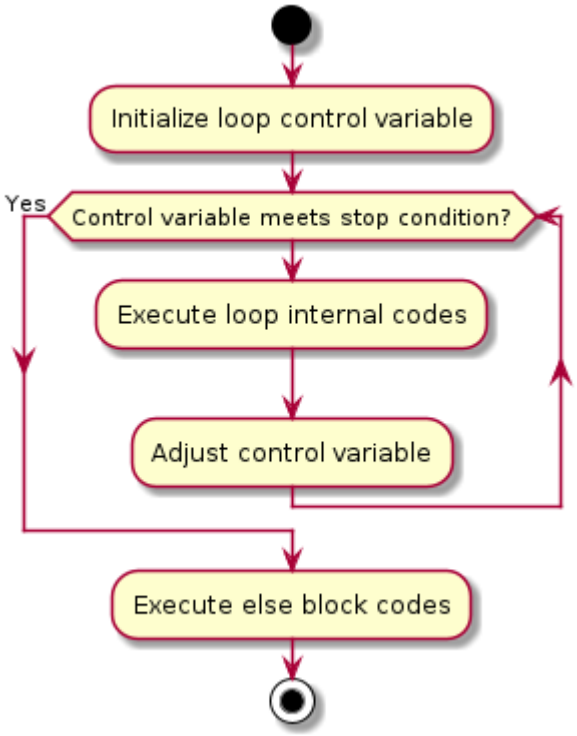
> **Note**
>
> a. iterator (n.) 疊代子
> b. iteration (n.)疊代
> c. iterable (a.) 可疊代的
> d. container (n.) 容器

6. Syntax

```
for expression in iterable:
    欲執行的程式碼1…
```

```
[else:]
    [欲執行的程式碼2…]
```

7. Activity Diagram



8. 從上圖可以看出for 迴圈的執行流程。
9. 一開始會先給定for 迴圈控制變數的初始值，然後判斷控制變數是否已符合迴圈終止條件；如果還沒符合終止條件，便會執行for 迴圈內部的程式碼，並且將控制變數進行疊代。
10. 執行完一次for 迴圈之後會回到條件判斷，看控制變數是否已符合迴圈終止條件，如果還不符合，則會再執行一次for 迴圈，直到控制變數符合終止條件為止。
11. 在前面的內容中，菲絲恩曾提到一個在for迴圈中常被使用的函數—range()。
12. 顧名思義，range就是範圍的意思。

函數	描述
range([start,]stop[, step])	建立整數序列

13. 如上表所示，range() 所必須要給予的傳入值是stop，也就是停止條件。至於start跟step都是選擇性選項屬性。
14. start是起始值，預設為0；step是疊代數字，預設為1。
15. 三者所需要的資料型態都是int。
16. 而range() 所傳回的資料是一個名稱為range 的物件，因此直接使用的可用性不高，通常會將其轉型成其他容器資料型態，如list、tuple 等。
17. 事實上，在for 迴圈中使用range() 時， 是將range() 放在上圖中的iterable，也就是疊代的位置。
18. 值得注意的是，range() 裡面的step 不僅能設定為正整數，也能設定為負整數（如-1、-2 等）。
19. 以下舉例說明之：

2.14.2.1.1. Ex1: range()

1. Source

Listing 2.14.2.1.1.1 /src/Loop/p0601Range.py

```
1  '''
2  Created on 2015年8月28日
```

```
3
4  @author: cph
5  '''
6  print(range(10))                # 直接印出range() 會印出range 物件，使用意義不大
7  print(list(range(10)))          # 將range 物件轉型為list
8  print(list(range(3, 15, 4)))    # 設定起始值為3，終止值為15，迭代數字為4
9  print(list(range(10, -14, -3))) # 設定起始值為10，終止值為-14，迭代數字為-3
10 print(tuple(range(-7, 24, 5)))  # 也可以將range 物件轉型為tuple
```

2. Output

```
1  range(0, 10)
2  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
3  [3, 7, 11]
4  [10, 7, 4, 1, -2, -5, -8, -11]
5  (-7, -2, 3, 8, 13, 18, 23)
```

3. 以起始值為10，終止值為-14，疊代數值為-3 的range() 為例，普羅可以發現range() 終止值的結果並沒有被輸出。

4. 也就是初始值疊代到小於或等於終止值時，疊代便會結束，然後將資料傳回給print() 進行列印的動作。

5. 請注意，上述例子中，若疊代數值為正數，則為大於或等於終止值時，疊代才會結束。

6. 如同上面所說，for 迴圈通常拿來執行已知執行次數的程式碼，例如：

7. Source 2

Listing 2.14.2.1.1.2 /src/Loop/p0601Range2.py

```
1  '''
2  Created on 2021年7月16日
3
4  @author: cph
5  '''
6  #
7  # range([start, ]stop[, step])
8  #
9  print(list(range(10)))
10 print(list(range(1, 10)))
11 print(list(range(1, 10, 1)))
12 print(list(range(0, 10, 1)))
13
14 print(list(range(10, 1, -1)))
15
16 print(list(range(0, 10, 2)))
17 print(list(range(10, 0, -2)))
18
19 print(list(range(1, 10, 3)))
20 print(list(range(10, 1, -3)))
```

8. Output 2

```
1  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
2  [1, 2, 3, 4, 5, 6, 7, 8, 9]
3  [1, 2, 3, 4, 5, 6, 7, 8, 9]
4  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
5  [10, 9, 8, 7, 6, 5, 4, 3, 2]
6  [0, 2, 4, 6, 8]
7  [10, 8, 6, 4, 2]
8  [1, 4, 7]
9  [10, 7, 4]
```

2.14.2.1.2. Ex2: Sum

1. Source

```
iSum = 0                # 將iSum給定初始值為0
for i in range(101):    # 設定for迴圈的執行條件，讓i從0疊代到100
    iSum += i           # 將i的值加到iSum
else:
    print("將0到100加總，所得總和為", iSum)    # 在迴圈執行結束時將iSum值印出
```

2. Output

```
將0到100加總，所得總和為 5050
```

2.14.2.1.3. Ex3: From While Loop To For Loop

1. Source 1: 1+2+...+8+9+10=55

Listing 2.14.2.1.3.1 /src/Loop/p14LoopWhileFor1/__init__.py

```
# 1+2+...+8+9+10=55
#== While Loop ==========================
iSum = 0                # 設定iSum為0
iCount = 1              # 設定iCount也就是起始條件為0
while (iCount <= 10):   # 當iCount < 100 時執行以下的程式碼
    iSum += iCount      # 將iCount的值加總到iSum
    iCount += 1         # 每次執行時iCount+1

print('While:', iSum)   # 利用格式化輸出

#== For Loop ============================
iSum = 0
for iCount in range(1, 11):
    iSum += iCount

print('For:', iSum)     # 利用格式化輸出

```

2. Output 1

```
While: 55
For: 55
```

3. Source 2: 10+9+...+2+1=55

Listing 2.14.2.1.3.2 /src/Loop/p14LoopWhileFor2/__init__.py

```
# 10+9+...+2+1=55
#== While Loop ==========================
iSum = 0
iCount = 10
while (iCount >= 1):
    iSum += iCount
    iCount -= 1

print('While:', iSum)  # 利用格式化輸出

#== For Loop ============================
iSum = 0
for iCount in range(10, 0, -1):

```

```
    iSum += iCount

print('For:', iSum)    # 利用格式化輸出
```

4. Output 2

```
While: 55
For: 55
```

5. Source 3: 1+3+5+7+9 = 25

Listing 2.14.2.1.3.3 /src/Loop/p14LoopWhileFor3/__init__.py

```
# 1+3+5+7+9 = 25
#== While Loop ========================
iSum = 0
iCount = 1
while (iCount <= 10):
    iSum += iCount
    iCount += 2

print('While:', iSum)

#== For Loop ==========================
iSum = 0
for iCount in range(1, 10, 2):
    iSum += iCount

print('For:', iSum)
```

6. Output 3

```
While: 25
For: 25
```

7. Source 3a: 9+7+5+3+1 = 25

Listing 2.14.2.1.3.4 /src/Loop/p14LoopWhileFor3a/__init__.py

```
# 9+7+5+3+1 = 25
#== While Loop ========================
iSum = 0
iCount = 9
while (iCount >= 1):
    iSum += iCount
    iCount -= 2

print('While:', iSum)

#== For Loop ==========================
iSum = 0
for iCount in range(9, 0, -2):
    iSum += iCount

print('For:', iSum)
```

8. Output 3a

```
While: 25
For: 25
```

9. Source 3b: 9+7+5+3+1 = 25

Listing 2.14.2.1.3.5 /src/Loop/p14LoopWhileFor3b/__init__.py

```
# 9+7+5+3+1 = 25
#== While Loop ========================
iSum = 0
iCount = 10
while (iCount >= 1):
    if (iCount % 2 == 1):
        iSum += iCount
    iCount -= 1

print('While:', iSum)

#== For Loop ==========================
iSum = 0
for iCount in range(9, 0, -1):
    if (iCount % 2 == 1):
        iSum += iCount

print('For:', iSum)
```

10. Output 3b

```
While: 25
For: 25
```

11. Code+Output 3c: 10+8+6+4+2 = 30

Code Output

```
While: 30
For: 30
```

12. Code+Output 3d: 10+7+4+1 = 22

Code Output

```
While: 22
For: 22
```

13. Code+Output 3e: 3+6+9+12=30

Code Output

```
While: 30
For: 30
```

14. Code+Output 4: 1+4+7+10 = 22

Code Output

```
While: 22
For: 22
```

2.14.2.1.4. Ex4: 6!

1. 複賽開始了， 第一關主考官要求普羅計算出6的階層 $6!$， 這點難度的題目對於他們來說簡直易如反掌， 普羅在彈指之間就算出來了。
2. Code+Output

Code Output

```
720
720
```

2.14.2.1.5. Ex5: Split Me!

1. 輕鬆地通過了第一關， 普羅正得意著。 但緊接著，下一關就是要普羅利用魔法把句子裡的字母給一一分開， 普羅頓時陷入難題之中。
2. 這時，菲絲恩信心喊話道：「這題目不難呀！腦筋轉個彎，答案就出來了。」
3. Code+Output

Code Output

```
sLine = "Split me!"
for i in sLine:
   print(i + "_", end = "")
```

2.14.2.1.6. Ex6: International Day of Mathematics

1. 每年的03/14是國際數學日， 有人說物理的極致是宗教，數學的極致是哲學，是有一定道理的。 輸出以下有趣的算式。

```
1 x 8 + 1 = 9
12 x 8 + 2 = 98
123 x 8 + 3 = 987
1234 x 8 + 4 = 9876
12345 x 8 + 5 = 98765
123456 x 8 + 6 = 987654
1234567 x 8 + 7 = 9876543
12345678 x 8 + 8 = 98765432
123456789 x 8 + 9 = 987654321
```

Code Output

Listing 2.14.2.1.6.1 /src/Loop/p0604LoopForMathDay.py

```
'''
Created on 20210722
@author: cph
'''
sCat = ''
for i in range(1, 10):
    sCat += str(i)
    print(sCat, 'x 8 +', i, '=', (int(sCat) * 8 + i))
```

Code Output

```
1 x 9 + 2 = 11
12 x 9 + 3 = 111
123 x 9 + 4 = 1111
1234 x 9 + 5 = 11111
12345 x 9 + 6 = 111111
123456 x 9 + 7 = 1111111
1234567 x 9 + 8 = 11111111
12345678 x 9 + 9 = 111111111
123456789 x 9 +10= 1111111111
```

Code Output

```
9 x 9 + 7 = 88
98 x 9 + 6 = 888
987 x 9 + 5 = 8888
9876 x 9 + 4 = 88888
98765 x 9 + 3 = 888888
987654 x 9 + 2 = 8888888
9876543 x 9 + 1 = 88888888
98765432 x 9 + 0 = 888888888
```

Code Output

```
1 x 1 = 1
11 x 11 = 121
111 x 111 = 12321
1111 x 1111 = 1234321
11111 x 11111 = 123454321
111111 x 111111 = 12345654321
1111111 x 1111111 = 1234567654321
11111111 x 11111111 = 123456787654321
111111111 x 111111111 = 12345678987654321
```

2.14.2.1.7. Ex7: Star-typed Triangles

1. Code+Output

Code Output

```
8
*                 *   ********   ********
**               **   *******     *******
***             ***   ******       ******
****           ****   *****         *****
```

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

2. Note that the shortest space between each triangle is with 3 spaces.

2.14.2.1.8. Ex8: Forever

1. Code+Output

Case 1: itertools.repeat() | Case 2: itertools.count() | Case 3a: itertools.cycle() | Case 3b: itertools.cycle() | Case 3c: itertools.cycle() | Case 3d: itertools.cycle() | Case 4: asyncio | Output

Listing 2.14.2.1.8.1 /src/Loop/p0604LoopForever1ItertoolsRepeat.py

```
'''
Created on 20230724
@author: cph
'''
import itertools

def counter():
    global iCnt, iMOD

    if (iCnt % iMOD == 0):
        print(f'.', end='')
    iCnt += 1

if __name__ == '__main__':
    iMOD = 10 ** 7
    iCnt = 0
    for _ in itertools.repeat(None):
        if (iCnt > iMOD * 10):
            break
        counter()
    print(f'End of infinite loop...')
```

Note

1. Start: 20170719
2. System Environment:

Listing 2.14.2.1.8.8 requirements.txt

```
sphinx>=6.1.3                           # Sphinx
graphviz>=0.20.1                        # Graphviz
sphinxbootstrap4theme>=0.6.0            # Theme: Bootstrap
sphinx-material>=0.0.35                 # Theme: Material
sphinxcontrib-plantuml>=0.25            # PlantUML
sphinxcontrib.bibtex>=2.5.0             # Bibliography
sphinx-autorun>=1.1.1                   # ExecCode: pycon
sphinx-execute-code-python3>=0.3        # ExecCode
btd.sphinx.inheritance-diagram>=2.3.1   # Diagram
sphinx-copybutton>=0.5.1                # Copy button
sphinx_code_tabs>=0.5.3                 # Tabs
sphinx-immaterial>=0.11.3               # Tabs

#-----------------------------------------------------
#-- Library Upgrade Error by Library Itself
#   >> It needs to fix by library owner
#   >> After fixed, we need to try it later
#-----------------------------------------------------
pydantic==1.10.10                       # 2.0: sphinx compiler error, 20230701

#-----------------------------------------------------
#-- Minor Extension
#-----------------------------------------------------
sphinxcontrib.httpdomain>=1.8.1         # HTTP API

#sphinxcontrib-blockdiag>=3.0.0         # Diagram: block
#sphinxcontrib-actdiag>=3.0.0           # Diagram: activity
#sphinxcontrib-nwdiag>=2.0.0            # Diagram: network
#sphinxcontrib-seqdiag>=3.0.0           # Diagram: sequence

#-----------------------------------------------------
#-- Still Wait For Upgrading Version
#-----------------------------------------------------

#-----------------------------------------------------
#-- Still Under Testing
#-----------------------------------------------------
#numpy>=1.24.2                          # Figure: numpy

#-----------------------------------------------------
#-- NOT Workable
#-----------------------------------------------------
#sphinxcontrib.jsdemo==0.1.4    # ExecCode: Need replace add_js_file()
#jupyter-sphinx==0.4.0          # ExecCode: Need gcc compiler
#sphinxcontrib.slide==1.0.0     # Slide: Slideshare
#hieroglyph==2.1.0              # Slide: make slides
#matplotlib>=3.7.1              # Plot: Need Python >= v3.8
#manim==0.17.2                  # Diagram: scipy, numpy need gcc
#sphinx_diagrams==0.4.0         # Diagram: Need GKE access
#sphinx-tabs>=3.4.1                     # Tabs: Conflict w/ sphinx-material
```