

Chapter 7

迴圈設計

7.1 基本 for 迴圈

- 計算 1 加到 10 的總和：

```
1 sum = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10
2 print(f"sum of 1 to 10 is {sum}")
```

sum of 1 to 10 is 55

- 計算 1 加到 100 的總和：？？？
- for 迴圈可以讓程式將整個物件內的元素**遍歷**（也可稱為**迭代**）。在迭代期間可以紀錄或輸出當下的**狀態**或稱為**軌跡**。

for **var** in **可迭代物件**:

程式碼區塊

- **可迭代物件**(iterable object)可以是**串列(list)**、**元組(Tuple)**、**字典(Dict)**與**集合(Set)**或 **range()** 函數產生的區間。

```
1 sum = 0
2 for val in range(1,101):    # 1 to <101
3     sum += val
4 print(f"sum of 1 to 100 is {sum}")
```

sum of 1 to 100 is 5050

7.1 基本 for 迴圈

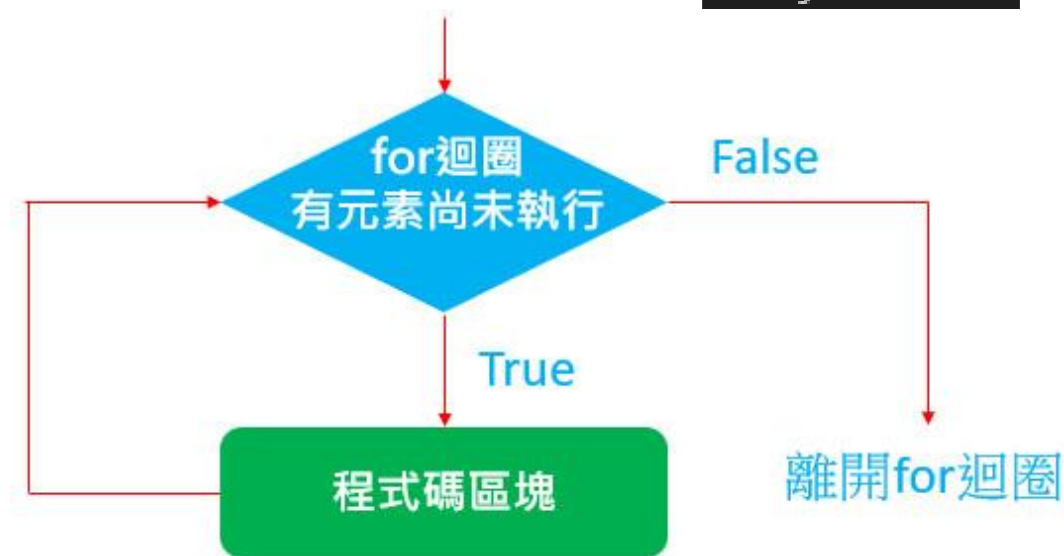
- for 的基本運作：

```
1 actresses = ["Nicole Kidman", "Angelina Jolie", "Gal Gadot", "Meryl Streep", "Emily Blunt"]
2 for actress in actresses:
3     print(actress)
```

```
Nicole Kidman
Angelina Jolie
Gal Gadot
Meryl Streep
Emily Blunt
```

- 若 for 的程式碼區塊只有一行：
for var in 可迭代物件: 程式碼區塊

```
1 actresses = ["Nicole Kidman", "Angelina Jolie"]
2 for actress in actresses: print(actress)
```



7.1 基本 for 迴圈

- for 的基本運作：
 - 迭代整個串列：

```
1 actresses = ["Nicole Kidman", "Angelina Jolie", "Gal Gadot", "Meryl Streep", "Emily Blunt"]
2 for actress in actresses:
3     print(f'What a great show {actress.split(" ")[0]}')
4     print(f"I can't wait to see your next show!!!")
```

```
What a great show Nicole
I can't wait to see your next show!!!
What a great show Angelina
I can't wait to see your next show!!!
What a great show Gal
I can't wait to see your next show!!!
What a great show Meryl
I can't wait to see your next show!!!
What a great show Emily
I can't wait to see your next show!!!
```

- 迭代部份串列：

```
1 actresses = ["Nicole Kidman", "Angelina Jolie", "Gal Gadot", "Meryl Streep", "Emily Blunt"]
2 for actress in actresses[1:4]:
3     print(f"{actress.split(' ')}")
```

```
['Angelina', 'Jolie']
['Gal', 'Gadot']
['Meryl', 'Streep']
```

7.1 基本 for 迴圈

- for 的基本運作：
 - 搭配 if 做篩選

```
1 actresses = ["Nicole Kidman", "Angelina Jolie", "Gal Gadot", "Meryl Streep", "Emily Blunt"]
2 for actress in actresses:
3     if actress.endswith('t'):
4         print(f"{actress}")
```

Gal Gadot
Emily Blunt

```
1 sum = 0
2 for val in range(1,101):    # 1 to <101
3     if (val % 3 == 0) and (val % 5 == 0) :    # val % 15 == 0
4         sum += val
5 print(f"Conditional sum is {sum}")
```

Conditional sum is 315

7.1 基本 for 迴圈

- for 的基本運作：
 - 兩個串列的比較

```
1 fruits1 = ["Banana", "Apple", "Pineapple", "Papaya", "Peach"]
2 fruits2 = ["Pineapple", "Papaya", "Watermelon", "Passion Fruit"]
3 print(f"Befort: {fruits2 = }")
4 for fruit in fruits1:
5     if fruit in fruits2:
6         fruits2.remove(fruit)
7         print(f"Remove {fruit} in fruits2")
8 print(f"After: {fruits2 = }")
```

```
Befort: fruits2 = ['Pineapple', 'Papaya', 'Watermelon', 'Passion Fruit']
Remove Pineapple in fruits2
Remove Papaya in fruits2
After: fruits2 = ['Watermelon', 'Passion Fruit']
```

7.2 range() 函數

- 可產生一個 range 物件，內容為一等差數列，也是個可迭代物件 (iterable object)。
- range(start, stop, step) :
 - start, stop, step 與串列的 slice 用法類似。
 - 若 step 為 1，會產生 start ~ stop - 1 的遞增數列 (stop > start)。
 - 若 step 為 -1，會產生 start ~ stop + 1 的遞減數列 (start > stop)。
 - range(n) = range(0, n) = range(0, n, 1) = 0~n-1 的遞增數列。

```
1 for x in range(3):
2     print(x)
```

0
1
2

```
1 for x in range(3, 0, -1):
2     print(x)
```

3
2
1

```
1 for x in range(0, 6, 2):
2     print(x)
```

0
2
4

7.2 range() 函數

- 跟 if 搭配：

```
1 print(f"{'-'*5}{'*'*5}{'-'*5}{'*'*5}{'-'*5}")
```

```
1 for x in range(5):
2     for y in range(5):
3         if x % 2:
4             print("*", end="")
5         else:
6             print("-", end="")
```

```
-----*****-----*****-----
```

- 將每次迭代的結果取出來：

<pre>1 money = 50000 2 rate = 0.005 3 years = 5 4 for year in range(years): 5 money *= (1 + rate) 6 print(f"第 {year + 1} 年本金和為 {int(money)}")</pre>	<pre>第 1 年本金和為 50249 第 2 年本金和為 50501 第 3 年本金和為 50753 第 4 年本金和為 51007 第 5 年本金和為 51262</pre>
---	--

7.2 range() 函數

- 活用 range() 函數：

```
1 start, end = eval(input("請輸入起始與結束數字："))
2 sum = 0
3 for val in range(start, end+1):
4     sum += val
5 print(f"Sum from {start} to {end} is {sum}")
```

```
請輸入起始與結束數字：5, 20
Sum from 5 to 20 is 200
```

```
1 start, end = eval(input("請輸入起始與結束數字："))
2 sum = sum(range(start, end+1))
3 print(f"Sum from {start} to {end} is {sum}")
```

- range 會產生一個 <class 'range'> 的物件，而不是串列
- 用這個方法，並不會預留出 start ~ end 的記憶體空間，而是只用一個記憶體空間，將每次迭代的結果放在這個空間中，然後執行 sum()。可以增加工作效率與節省記憶體空間。

7.2 range() 函數

- 用 range() 照一定規則新增串列特定元素：

```
1 squares = []
2 count = eval(input("請輸入一個數值："))
3 for idx in range(1, count+1):
4     squares.append(idx * idx)    # squares.append(idx ** 2)
5 print(f"squares = ")
```

請輸入一個數值：6

squares = [1, 4, 9, 16, 25, 36]

```
1 count = eval(input("請輸入一個數值："))
2 squares = [None] * count
3 for idx in range(1, count+1):
4     squares[idx - 1] = (idx * idx)    # squares[idx - 1](idx ** 2)
5 print(f"squares = ")
```

7.2 range() 函數

- 用 range() 刪除串列所有元素：
 - 串列沒有提供刪除整個串列元素的方法。list.clear()???

```
1 fruits1 = ["Banana", "Apple", "Pineapple", "Papaya", "Peach"]
2 print(f"Fruits in list are: {fruits1}")
3 for fruit in fruits1[:]:
4     fruits1.remove(fruit)
5     print(f"Remove {fruit}.\nFruits left in list are:{fruits1}")
```

```
Fruits in list are: ['Banana', 'Apple', 'Pineapple', 'Papaya', 'Peach']
Remove Banana.
Fruits left in list are:['Apple', 'Pineapple', 'Papaya', 'Peach']
Remove Apple.
Fruits left in list are:['Pineapple', 'Papaya', 'Peach']
Remove Pineapple.
Fruits left in list are:['Papaya', 'Peach']
Remove Papaya.
Fruits left in list are:['Peach']
Remove Peach.
Fruits left in list are:[]
```

```
1 fruits1 = ["Banana", "Apple", "Pineapple", "Papaya", "Peach"]
2 print(fruits1)
3 fruits1.clear()
4 print(fruits1)
```

```
['Banana', 'Apple', 'Pineapple', 'Papaya', 'Peach']
[]
```

```
1 fruits1 = ["Banana", "Apple", "Pineapple", "Papaya", "Peach"]
2 print(f"Fruits in list are: {fruits1}")
3 for fruit in fruits1:
4     fruits1.remove(fruit)
5     print(f"Remove {fruit}.\nFruits left in list are:{fruits1}")
```

輸出會是什麼？

7.2 range() 函數

- 產生一個內容有規則的串列：

```
1  xList = []
2  xList.append(0)
3  xList.append(1)
4  xList.append(2)
5  xList.append(3)
6  xList.append(4)
7  xList.append(5)
8
9  print(xList) [0, 1, 2, 3, 4, 5]
```

```
1  xList = []
2  for n in range(6):
3      xList.append(n)
4
5  print(xList)
```

```
1  xList = list(range(6))
2
3  print(xList)
```

7.2 range() 函數

- List generator: 用迭代方式產生串列資料的方法。

新串列 = [運算式 for 項目 in 可迭代物件]

```
1 xList = [ n for n in range(6)]
2
3 print(xList) [0, 1, 2, 3, 4, 5]
```

```
1 count = eval(input("請輸入一個數值："))
2 squares = []
3 for idx in range(1, count+1):
4     squares.append(idx * idx) # squares.append(idx ** 2)
5 print(f"squares = ")
6
7 newSquares = [idx ** 2 for idx in range(1, count+1)]
8 print(f"newSquares = ")
```

```
請輸入一個數值：6
squares = [1, 4, 9, 16, 25, 36]
newSquares = [1, 4, 9, 16, 25, 36]
```

- 攝氏轉華氏：

```
1 celsius = [21, 25, 30]
2 fahrenheit = [(x * 9 / 5 + 32) for x in celsius]
3 print(f"fahrenheit = ") fahrenheit = [69.8, 77.0, 86.0]
```

7.2 range() 函數

- List generator: 用迭代方式產生串列資料的方法。

搭配 if :

```
1 count = eval(input("請輸入一個數值："))
2 squares = []
3 for idx in range(1, count+1):
4     if count <= 7:
5         squares.append(idx * idx) # squares.append(idx ** 2)
6 print(f"squares = ")
7
8 newSquares = [idx ** 2 for idx in range(1, count+1) if count <= 7]
9 print(f"newSquares = ")
```

```
squares = [1, 4, 9, 16, 25, 36, 49]
newSquares = [1, 4, 9, 16, 25, 36, 49]
```

7.2 range() 函數

- List generator: 用迭代方式產生串列資料的方法。
 - 搭配多重指定與 if：列出邊長小於16的直角三角形的三邊長：

```
1 len = 16
2 tri90 = []
3 for a in range(1, len):
4     for b in range(1, len):
5         for c in range(1, len):
6             if a ** 2 + b ** 2 == c ** 2:
7                 tri90.append([a, b, c])
8 print(tri90)
9
10 tri90 = [[a, b, c] for a in range(1, len) for b in range(1, len) for c in range(1, len)
11           if a ** 2 + b ** 2 == c ** 2]
12 print(tri90)
```

```
[[3, 4, 5], [4, 3, 5], [5, 12, 13], [6, 8, 10], [8, 6, 10], [9, 12, 15], [12, 5, 13], [12, 9, 15]]
[[3, 4, 5], [4, 3, 5], [5, 12, 13], [6, 8, 10], [8, 6, 10], [9, 12, 15], [12, 5, 13], [12, 9, 15]]
```

7.2 range() 函數

- List generator: 用迭代方式產生串列資料的方法。
 - 搭配多重指定與 if：列出邊長小於16的直角三角形的三邊長。

```

1  len = 16
2  tri90 = [[a, b, c] for a in range(1, len) for b in range(1, len) for c in range(1, len)
3  |      |      if a ** 2 + b ** 2 == c ** 2]
4  print(tri90)
5
6  tri90 = [[a, b, c] for a in range(1, len) for b in range(a, len) for c in range(b, len)
7  |      |      if a ** 2 + b ** 2 == c ** 2]
8  print(tri90)
9
10 tri90 = [[a, b, c] for a in range(1, len) for b in range(a, len) for c in range(b, len)
11 |      |      if a ** 2 + b ** 2 == c ** 2 and a % 2 == 0]
12 print(tri90)

```

```

[[3, 4, 5], [4, 3, 5], [5, 12, 13], [6, 8, 10], [8, 6, 10], [9, 12, 15], [12, 5, 13], [12, 9, 15]]
[[3, 4, 5], [5, 12, 13], [6, 8, 10], [9, 12, 15]]
[[6, 8, 10]]

```


7.2 range() 函數

- List generator: 用迭代方式產生串列資料的方法。
 - 兩串列內容相乘。
 - $A * B = \{(a, b)\} : a \text{ 屬於 } A \text{ 元素}, b \text{ 屬於 } B \text{ 元素}$

```

1  listColors = ["Red", "Green", "Blue"]
2  listShape = ["Circle", "Triangle", "Square"]
3  result = []
4  for color in listColors:
5      for shape in listShape:
6          result.append([color, shape])
7  print(result)
8
9  result = [[color, shape] for color in listColors for shape in listShape]
10 print(result)

```

```

[['Red', 'Circle'], ['Red', 'Triangle'], ['Red', 'Square'], ['Green', 'Circle'], ['Green', 'Triangle'], ['Green', 'Square'], ['Blue', 'Circle'], ['Blue', 'Triangle'], ['Blue', 'Square']]
[['Red', 'Circle'], ['Red', 'Triangle'], ['Red', 'Square'], ['Green', 'Circle'], ['Green', 'Triangle'], ['Green', 'Square'], ['Blue', 'Circle'], ['Blue', 'Triangle'], ['Blue', 'Square']]

```

7.2 range() 函數

- 列印含串列元素的串列：

```
1 listColors = ["Red", "Green", "Blue"]
2 listShape = ["Circle", "Triangle", "Square"]
3 result = [[color, shape] for color in listColors for shape in listShape]
4 for color, shape in result:
5     print(color, shape)
```

Red Circle
Red Triangle
Red Square
Green Circle
Green Triangle
Green Square
Blue Circle
Blue Triangle
Blue Square

7.3 進階的 for 迴圈應用

- 巢狀for迴圈：

for var1 in 可迭代物件:

外層for迴圈

...外層程式區塊

 for var2 in 可迭代物件:

內層for迴圈

 內層程式區塊

7.3 進階的 for 迴圈應用

- 巢狀for迴圈：
 - 99乘法表：

```
1  for i in range(1, 10):
2      for j in range(1, 10):
3          res = i * j
4          print(f"{j}*{i}={i*j:>2d}", end=" ")
5      print("") #換行
```

1*1= 1	2*1= 2	3*1= 3	4*1= 4	5*1= 5	6*1= 6	7*1= 7	8*1= 8	9*1= 9
1*2= 2	2*2= 4	3*2= 6	4*2= 8	5*2=10	6*2=12	7*2=14	8*2=16	9*2=18
1*3= 3	2*3= 6	3*3= 9	4*3=12	5*3=15	6*3=18	7*3=21	8*3=24	9*3=27
1*4= 4	2*4= 8	3*4=12	4*4=16	5*4=20	6*4=24	7*4=28	8*4=32	9*4=36
1*5= 5	2*5=10	3*5=15	4*5=20	5*5=25	6*5=30	7*5=35	8*5=40	9*5=45
1*6= 6	2*6=12	3*6=18	4*6=24	5*6=30	6*6=36	7*6=42	8*6=48	9*6=54
1*7= 7	2*7=14	3*7=21	4*7=28	5*7=35	6*7=42	7*7=49	8*7=56	9*7=63
1*8= 8	2*8=16	3*8=24	4*8=32	5*8=40	6*8=48	7*8=56	8*8=64	9*8=72
1*9= 9	2*9=18	3*9=27	4*9=36	5*9=45	6*9=54	7*9=63	8*9=72	9*9=81

7.3 進階的 for 迴圈應用

- 巢狀for迴圈：
 - 直角三角形：

```
1  for i in range(1, 10):  
2      for j in range(1, 10):  
3          if j <= i:  
4              print("aa", end="")  
5      print()
```

for j in range(1, i+1):

```
aa  
aaaa  
aaaaaa  
aaaaaaaa  
aaaaaaaaaa  
aaaaaaaaaaaa  
aaaaaaaaaaaaaa  
aaaaaaaaaaaaaaaa  
aaaaaaaaaaaaaaaaaa  
aaaaaaaaaaaaaaaaaa
```

```
1  for i in range(1, 10):  
2      symbol = "aa" * i  
3      print(symbol)
```

7.3 進階的 for 迴圈應用

- break 指令：
 - 強制離開 for 迴圈：

for var in 可迭代物件:

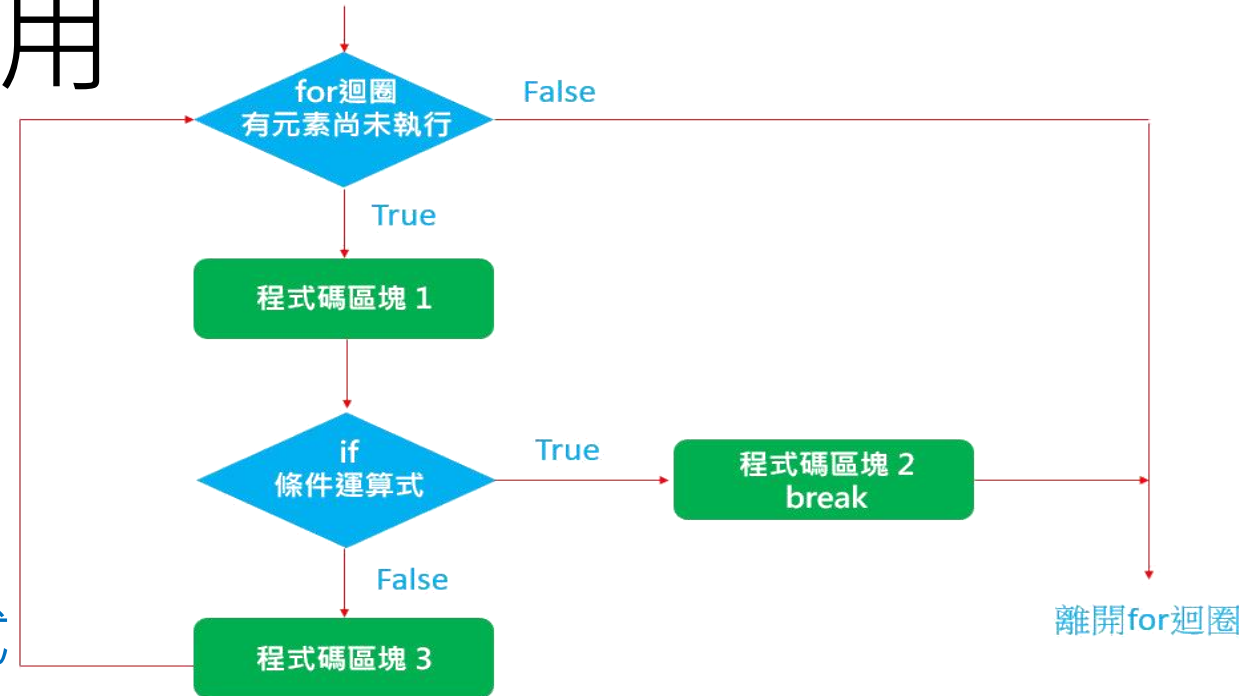
 程式碼區塊1

 if 條件運算式: # 判斷條件運算式

 程式碼區塊2

 break # 如果條件運算式是True則離開for迴圈

 程式碼區塊3



7.3 進階的 for 迴圈應用

- break 指令：

```

1  print("Test 1:")
2  for digit in range(1, 11):
3      if digit == 5:
4          break
5      print(digit, end=" ")
6  print()
7
8  print("Test 2:")
9  for digit in range(0, 11, 2):
10     if digit == 5:
11         break
12     print(digit, end=" ")

```

```

Test 1:
1 2 3 4
Test 2:
0 2 4 6 8 10

```

```

1  scores = [90, 83, 96, 92, 59, 88, 91]
2  scores.sort(reverse = True)
3  count = 0
4  for score in scores:
5      print(score, end = " ")
6      count += 1
7      if count == 5:
8          break

```

```
96 92 91 90 88
```

```

1  scores = [90, 83, 96, 92, 59, 88, 91]
2  scores.sort(reverse = True)
3  count = 0
4  for score in scores[:5]:
5      print(score, end = " ")

```

7.3 進階的 for 迴圈應用

- continue 指令：
 - 某些條件發生時，本次迴圈剩餘的程式碼區塊可以不繼續執行，直接進入下一次迭代：

for var in 可迭代物件:

 程式碼區塊1

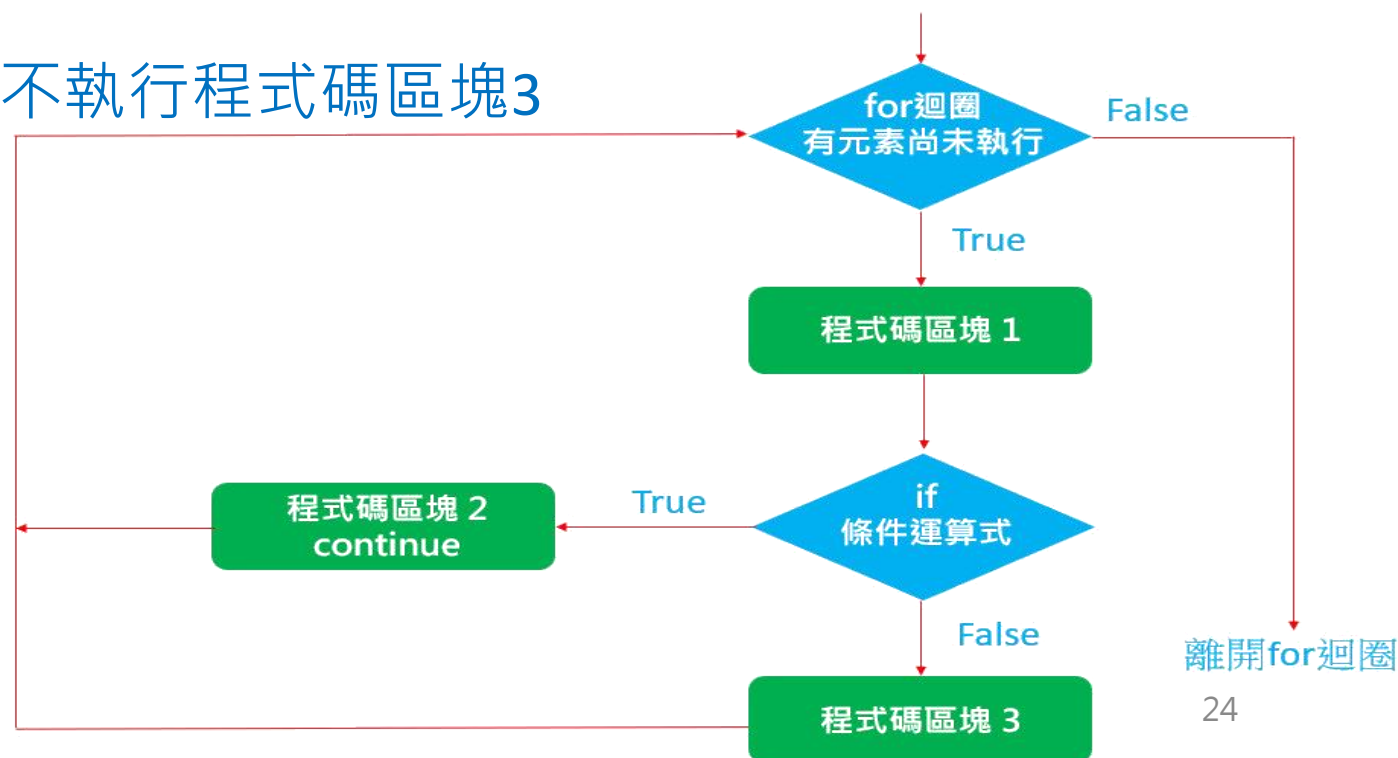
 #如果條件運算式是True則不執程式碼區塊3

 if 條件運算式:

 程式碼區塊2

 continue

 程式碼區塊3



7.3 進階的 for 迴圈應用

- continue 指令：

```
1 scores = [90, 83, 96, 92, 59, 88, 91]
2 count = 0
3 for score in scores:
4     if score < 90:
5         continue
6     print(score, end = " ")
7     count += 1
8 print(f"\nThere are {count} times that score > 90")
```

90 96 92 91
There are 4 times that score > 90

```
1 scores = [90, 83, 96, 92, 59, 88, 91]
2 scores.sort(reverse = True)
3 count = 0
4 for score in scores:
5     if score < 90:
6         break
7     print(score, end = " ")
8     count += 1
9 print(f"\nThere are {count} times that score > 90")
```

96 92 91 90
There are 4 times that score > 90

7.3 進階的 for 迴圈應用

- for ... else ... :
 - 通常跟 if, break 一起使用。

for var in 可迭代物件

if 條件運算式

程式碼區塊1

break

else:

程式碼區塊2

#如果條件為 True 就離開迴圈，不執行 else 部份。

#如果迴圈沒有被 break 就執行

7.3 進階的 for 迴圈應用

- for ... else ... :
 - 檢查一個數是否為質數：

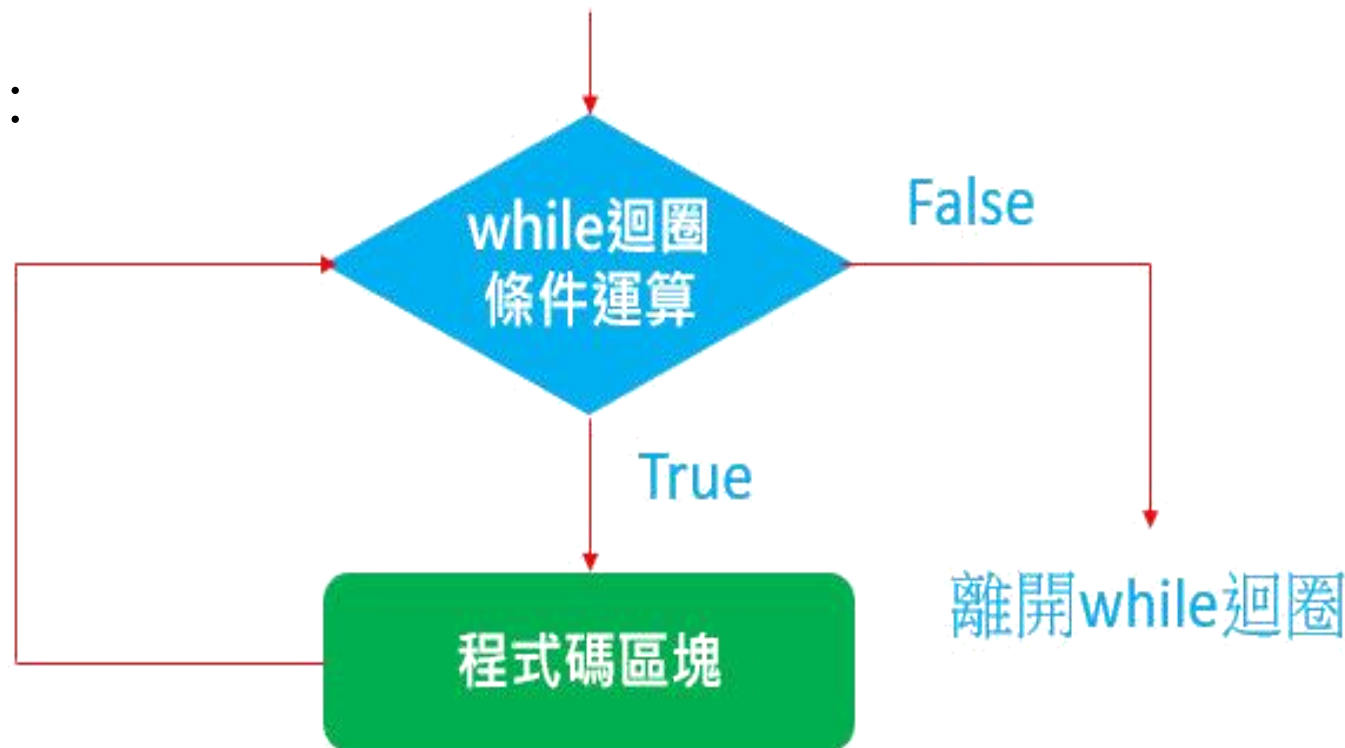
```
1  num = int(input("Input a integer larger than 1: "))
2  if num == 2:
3      print("2 is a prime number.")
4  else:
5      for n in range(2, num):
6          if num % n == 0:
7              print(f"{num} is not a prime number.")
8              break
9  else:
10     print(f"{num} is a prime number.")
```

```
Input a integer larger than 1: 5
5 is a prime number.
```

7.4 while 迴圈

- for 迴圈是一種計數迴圈，會重複執行一定次數。
- while 迴圈是一種條件控制迴圈，會執行到條件滿足才結束。

while 條件運算：
程式區塊



7.4 while 迴圈

- 印出輸入的字串，直到輸入 q 結束 while 迴圈。
- q 又稱作哨兵值(Sentinel value)

```
1 inputMsg = ""
2 while inputMsg != "q":
3     inputMsg = input("Input q to break loop: ")
4     print(inputMsg)
```

Input q to break loop: abc
abc
Input q to break loop: def
def
Input q to break loop: q
q

```
1 inputMsg = ""
2 while inputMsg != "q":
3     inputMsg = input("Input q to break loop: ")
4     if inputMsg != "q":
5         print(inputMsg)
```

Input q to break loop: abc
abc
Input q to break loop: def
def
Input q to break loop: q

```
1 inputMsg = ""
2 loopContinue = True
3 while loopContinue:
4     inputMsg = input("Input q to break loop: ")
5     if inputMsg == "q":
6         loopContinue = False
7     else:
8         print(inputMsg)
```

Input q to break loop: abc
abc
Input q to break loop: def
def
Input q to break loop: q

7.4 while 迴圈

- 猜一個 1 ~ 100 的數字：

```
1  answer = 63
2  guessInt = -1
3  while guessInt != answer:
4      guessInt = int(input("Choose a integer between 1 to 100: "))
5      if guessInt < answer:
6          print("Guess a \"LARGER\" ingeger.")
7      elif guessInt > answer:
8          print("Guess a \"SMALLER\" ingeger.")
9      else:
10         print("Right guess~~~~~")
```

```
Choose a integer between 1 to 100: 50
Guess a "LARGER" ingeger.
Choose a integer between 1 to 100: 75
Guess a "SMALLER" ingeger.
Choose a integer between 1 to 100: 63
Right guess~~~~~
```

- 預測學費：

```
1  tuition = 50000
2  year = 0
3  while tuition < 60000:
4      tuition = int(tuition * 1.05)
5      year += 1
6      print(f"經過 {year} 年，學費是 {tuition}")
7  print(f"經過 {year} 年，學費會超過60000")
```

```
經過 1 年，學費是 52500
經過 2 年，學費是 55125
經過 3 年，學費是 57881
經過 4 年，學費是 60775
經過 4 年，學費會超過60000
```

7.4 while 迴圈

- 巢狀 while 迴圈

while 條件運算：

...

while 條件運算：

...

- 99乘法表

外層while迴圈

內層while迴圈

```
1  i = 1
2  while i <= 9:
3      j = 1
4      while j <= 9:
5          res = i * j
6          print(f"{j}*{i}={i*j:>2d}", end=" ")
7          j += 1
8      print() #換行
9      i += 1
```

1*1= 1	2*1= 2	3*1= 3	4*1= 4	5*1= 5	6*1= 6	7*1= 7	8*1= 8	9*1= 9
1*2= 2	2*2= 4	3*2= 6	4*2= 8	5*2=10	6*2=12	7*2=14	8*2=16	9*2=18
1*3= 3	2*3= 6	3*3= 9	4*3=12	5*3=15	6*3=18	7*3=21	8*3=24	9*3=27
1*4= 4	2*4= 8	3*4=12	4*4=16	5*4=20	6*4=24	7*4=28	8*4=32	9*4=36
1*5= 5	2*5=10	3*5=15	4*5=20	5*5=25	6*5=30	7*5=35	8*5=40	9*5=45
1*6= 6	2*6=12	3*6=18	4*6=24	5*6=30	6*6=36	7*6=42	8*6=48	9*6=54
1*7= 7	2*7=14	3*7=21	4*7=28	5*7=35	6*7=42	7*7=49	8*7=56	9*7=63
1*8= 8	2*8=16	3*8=24	4*8=32	5*8=40	6*8=48	7*8=56	8*8=64	9*8=72
1*9= 9	2*9=18	3*9=27	4*9=36	5*9=45	6*9=54	7*9=63	8*9=72	9*9=81

7.4 while 迴圈

- break 指令：強制離開 while 迴圈

while 運算條件式 A:

程式碼區塊 1

if 運算條件式 B:

程式碼區塊 2

break

程式碼區塊 3

- 猜一個 1 ~ 100 的數字：

```
1  answer = 63
2
3  while True:
4      inputInt = int(input("Choose a integer between 1 to 100: "))
5      if inputInt < answer:
6          print("Guess a \"LARGER\" ingeger.")
7      elif inputInt > answer:
8          print("Guess a \"SMALLER\" ingeger.")
9      else:
10         print("Right guess~~~~~")
11         break
```

```
Choose a integer between 1 to 100: 50
Guess a "LARGER" ingeger.
Choose a integer between 1 to 100: 75
Guess a "SMALLER" ingeger.
Choose a integer between 1 to 100: 63
Right guess~~~~~
```


7.4 while 迴圈

- break 指令：強制離開 while 迴圈
 - 隨機產生一個1000 ~ 9999的數，其中每一個位數值都不相同。

```
1  import random
2
3  answer = -1
4  listDigit = []
5  while True:
6      breakFlag = True
7      answer = random.randint(1000, 9999)
8      listDigit = list(str(answer))
9      for iIdx1 in range(len(listDigit)):
10         for iIdx2 in range(iIdx1+1, len(listDigit)):
11             if listDigit[iIdx1] == listDigit[iIdx2]:
12                 breakFlag = False
13         if breakFlag:
14             break
15     print(f"{answer:04d}")
```

6238
5690
0586

7.4 while 迴圈

- `continue` 指令：某些條件發生時，本次迴圈剩餘的程式碼區塊可以不繼續執行，直接進入下一次迭代：

while 運算條件式 A:

程式碼區塊1

#如果條件運算式 B是True則不執行程式碼區塊3

if 條件運算式 B:

程式碼區塊2

`continue`

程式碼區塊3

- 印出 1~10 之間的偶數：

```
1 index = 0
2 while index <= 10:
3     index += 1
4     if index % 2 == 1:
5         continue
6     print(index, end = " ")
    2 4 6 8 10
```

7.4 while 迴圈

- 條件運算式與可迭代物件：

while var in 可迭代物件:
 程式碼區塊

如果 var in 可迭代物件是 True 就繼續

while 可迭代物件:
 程式碼區塊

如果可迭代物件空的才結束

7.4 while 迴圈

- 將串列中的 Apple 刪除：

```
1 fruits1 = ["Banana", "Apple", "Pineapple", "Apple", "Papaya", "Peach", "Apple"]
2 print(f"Fruits in list are: {fruits1}")
3 fruit = "Apple"
4 #for fruit in fruits1[:]:
5 while fruit in fruits1:
6     fruits1.remove(fruit)
7     print(f"Remove {fruit}.\nFruits left in list are:{fruits1}")
```

```
Fruits in list are: ['Banana', 'Apple', 'Pineapple', 'Apple', 'Papaya', 'Peach', 'Apple']
Remove Apple.
Fruits left in list are:['Banana', 'Pineapple', 'Apple', 'Papaya', 'Peach', 'Apple']
Remove Apple.
Fruits left in list are:['Banana', 'Pineapple', 'Papaya', 'Peach', 'Apple']
Remove Apple.
Fruits left in list are:['Banana', 'Pineapple', 'Papaya', 'Peach']
```

- 有一串列，請依消費金額將消費者分成 VIP 與 Golden 兩個串列。

```
1 allBuyers = [{"James", 1030},
2             ["Curry", 2000],
3             ["Durant", 803],
4             ["Jordon", 3029],
5             ["David", 2502]]
6 goldenBuyer = []
7 vipBuyer = []
8 while allBuyers:
9     buyer = allBuyers.pop()
10    if buyer[1] > 2500:
11        vipBuyer.append(buyer)
12    else:
13        goldenBuyer.append(buyer)
14 print(f"VIP: {vipBuyer}")
15 print(f"Golden: {goldenBuyer}")
```

```
VIP: [['David', 2502], ['Jordon', 3029]]
Golden: [['Durant', 803], ['Curry', 2000], ['James', 1030]]
```

7.4 while 迴圈

- 無限迴圈：

```
while True:          #while 1
    程式碼區塊
```

- pass 指令是什麼也不做。

```
While True:
    pass
```

- 不過不建議這麼做。這個有時候會用在設計一個迴圈或函數，在尚未完成時，先放 pass，等未來再用完整的程式碼取代。

7.5 enumerate 物件使用 for 迴圈解析

- enumerate 物件是由索引值與元素值配對出現。
- 可以將串列用 enumerate 建立成 enumerate 物件，再搭配 for 迴圈將每個物件的索引值與元素值解析出來。

<pre> 1 drinks = ["coffee", "milk", "tea"] 2 for drink in enumerate(drinks): 3 print(drink) 4 for count, drink in enumerate(drinks): 5 print(count, drink) 6 print("*****") 7 for drink in enumerate(drinks, 10): 8 print(drink) 9 for count, drink in enumerate(drinks, 10): 10 print(count, drink) </pre>	<pre> (0, 'coffee') (1, 'milk') (2, 'tea') 0 coffee 1 milk 2 tea ***** (10, 'coffee') (11, 'milk') (12, 'tea') 10 coffee 11 milk 12 tea </pre>
--	--

7.5 enumerate 物件使用 for 迴圈解析

- 傳統寫法：

```
1 scores = [21, 29, 18, 33, 12, 17, 26]
2 gameIdx = 1
3 for score in scores:
4     if score >= 20:
5         print(f"Game {gameIdx}, scores: {score}")
6     gameIdx += 1
```

```
Game 1, scores: 21
Game 2, scores: 29
Game 4, scores: 33
Game 7, scores: 26
```

- 符合 Python 精神寫法：

```
1 scores = [21, 29, 18, 33, 12, 17, 26]
2 for gameIdx, score in enumerate(scores, 1):
3     if score >= 20:
4         print(f"Game {gameIdx}, scores: {score}")
```

```
Game 1, scores: 21
Game 2, scores: 29
Game 4, scores: 33
Game 7, scores: 26
```

7.6 動手練習

- 使用迴圈計算成績的總分、平均與排名：

座號	姓名	國文	英文	數學	總分	平均	名次
1	王小明	80	95	88			
2	蘇小花	98	97	96			
3	謝大樹	94	93	85			
4	李大呆	91	94	95			
5	陳小比	92	97	90			

```

1  scoreTable = [ ["王小明", 80, 95, 88, 0],
2                  ["蘇小花", 98, 97, 96, 0],
3                  ["謝大樹", 94, 93, 85, 0],
4                  ["李大呆", 91, 94, 95, 0],
5                  ["陳小比", 92, 97, 90, 0],
6                  ]
7
8  scoreTable[0][4] = sum(scoreTable[0][1:4])
9  scoreTable[1][4] = sum(scoreTable[1][1:4])
10 scoreTable[2][4] = sum(scoreTable[2][1:4])
11 scoreTable[3][4] = sum(scoreTable[3][1:4])
12 scoreTable[4][4] = sum(scoreTable[4][1:4])
13
14 print(f"  姓名  , 國文, 英文, 數學, 總分")
15 print(f"{scoreTable[0]}")
16 print(f"{scoreTable[1]}")
17 print(f"{scoreTable[2]}")
18 print(f"{scoreTable[3]}")
19 print(f"{scoreTable[4]}")

```

姓名 , 國文, 英文, 數學, 總分
['王小明', 80, 95, 88, 263]
['蘇小花', 98, 97, 96, 291]
['謝大樹', 94, 93, 85, 272]
['李大呆', 91, 94, 95, 280]
['陳小比', 92, 97, 90, 279]

7.6 動手練習

- 使用迴圈計算成績的總分、平均與排名：
 - 可能會用到二維串列的 sort：二維串列.sort(key=lambda x:x[n], reverse=True)
 - 對二維陣列中第n 個元素做升冪 (reverse=False, 小到大) /降冪 (reverse=True,大到小) 的排序

```

1  scoreTable = [[1, "王小明", 80, 95, 88, 0],
2                [2, "蘇小花", 98, 97, 96, 0],
3                [3, "謝大樹", 94, 93, 85, 0],
4                [4, "李大呆", 91, 94, 95, 0],
5                [5, "陳小比", 92, 97, 90, 0],
6                ]
7  print("BeforeSort:")
8  for idx in range(len(scoreTable)):
9      print(scoreTable[idx])
10 scTable = scoreTable[:]
11 scTable.sort(key=lambda x: x[2], reverse=True)
12 print("After Sort:")
13 for idx in range(len(scTable)):
14     print(scTable[idx])
    
```

```

BeforeSort:
[1, '王小明', 80, 95, 88, 0]
[2, '蘇小花', 98, 97, 96, 0]
[3, '謝大樹', 94, 93, 85, 0]
[4, '李大呆', 91, 94, 95, 0]
[5, '陳小比', 92, 97, 90, 0]
After Sort:
[2, '蘇小花', 98, 97, 96, 0]
[3, '謝大樹', 94, 93, 85, 0]
[5, '陳小比', 92, 97, 90, 0]
[4, '李大呆', 91, 94, 95, 0]
[1, '王小明', 80, 95, 88, 0]
    
```

7.6 動手練習

- 使用迴圈計算雞兔同籠的問題：
 - 雞兔同籠，共有100隻腳，請列出所有可能的雞兔隻數的結果？
- 計算圓周率：使用萊布尼茲公式計算圓週率，使 $i = 1$ to 100萬，每計算20萬次就輸出一當前 pi 值。

$$pi = 4(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \dots + \frac{(-1)^{i+1}}{2i-1})$$

```
When i = 200000, Pi = 3.1415876535897618
When i = 400000, Pi = 3.141590153589744
When i = 600000, Pi = 3.1415909869230147
When i = 800000, Pi = 3.1415914035897172
When i = 1000000, Pi = 3.1415916535897743
```