

Python

技術者們 實踐!

第 3 章 Python 的流程控制

本投影片（下稱教用資源）僅授權給採用教用資源相關之旗標書籍為教科書之授課老師（下稱老師）專用，老師為教學使用之目的，得摘錄、編輯、重製教用資源（但使用量不得超過各該教用資源內容之80%）以製作為輔助教學之教學投影片，並於授課時搭配旗標書籍公開播放，但不得為網際網路公開傳輸之遠距教學、網路教學等之使用；除此之外，老師不得再授權予任何第三人使用，並不得將依此授權所製作之教學投影片之相關著作物移作他用。

3-0 文字輸入與輸出的技巧

- 文字輸入

```
InputString = input("提示文字")
```

傳回一個字串

```
name = input("請問你是?")  
print(name + " 你好!")
```



請問你是?大雄 ← 輸入：大雄, 然後按 **Enter** 鍵
大雄 你好! ← 程式即會顯示這一行

IPython console

Console 1/A

請問你是?

請先用滑鼠在 Spyder 右下的 IPython console 窗格中點一下, 然後才能輸入

Python



文字輸出

```
print(資料1, 資料2, ..., sep=資料分隔字串, end=結束時要加的字串)
```

要輸出的資料

指定輸出格式

```
print('蘋果', '柳丁', '香蕉', sep=',', end='很好吃\n') # \n 為換行字元
```



蘋果、柳丁、香蕉很好吃 ← ',' 為分隔字串, '很好吃\n' 為結尾

Python



文字的格式化

```
a, b = 2, 3
print(f'{a} 和 {b} 相加的結果是 {a+b}')
print(f'{a} 和 {b} 相乘的結果是 {a*b}')
```

輸出 → 2 和 3 相加的結果是 5
輸出 → 2 和 3 相乘的結果是 6

在字串前加 *f*
字串裡用大括號嵌入變數和運算式
在輸出時即會套用運算的結果

```
a, b = 2, 3
print(f'{a} 和 {b} 相乘的結果是{a*b:+5.1f}')
```

輸出 → 2 和 3 相乘的結果是 +6.0

在運算式之後加：
+ 號表示要顯示正負號，5.1 表示長度至少 5 個字元，
小數保留 1 位，最後的 *f* 表示以浮點數輸出。

運算結果：前面空 1 格
(讓總長度 5 個字)，有
+ 號，小數 1 位



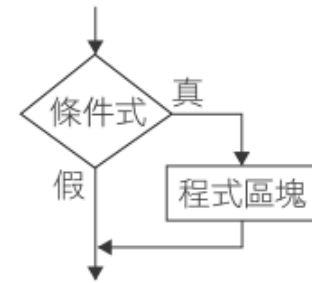
3-1 if 判斷式

- if 判斷式可以在程式中做「如果...就...」的判斷, 寫法如下：

if 條件式：← 注意最後要加：

```

    程式區塊 } ← 可以有多行程式, 每行都要向右縮排
    ...
  
```



```

if a < 1: ← if 判斷式
    a += 1
    b = a + 3 } ← 程式區塊
print(b) ← 接下來的程式未縮排, 不屬於 if 區塊了
...
  
```



- 如果區塊中只有一行程式, 那麼也可併到 `if` 的冒號之後

```
if a < 1: a += 1    # 合併為一行的 if  
print(a)
```

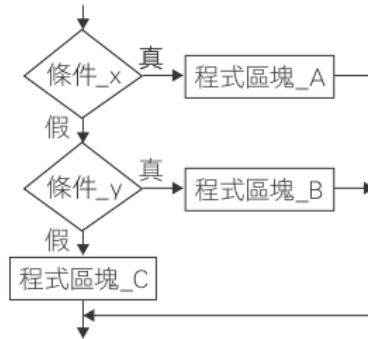
Python



if...elif...else...

```

if 條件_x:      ← 如果 x
    程式區塊_A
elif 條件_y:   ← 否則如果 y
    程式區塊_B
else:          ← 否則
    程式區塊_C
  
```



```

a = 70
if a >= 90:    # 如果 >= 90
    grade = 'A'
elif a >= 80:  # 否則如果 >= 80
    grade = 'B'
elif a >= 70:  # 否則如果 >= 70
    grade = 'C'
elif a >= 60:  # 否則如果 >= 60
    grade = 'D'
else:          # 否則
    grade = 'E'
print(f'{a} 分, 等第為 {grade}')
  
```



70 分, 等第為 C

```

a = 70
if a >= 90: grade = 'A'
elif a >= 80: grade = 'B'
elif a >= 70: grade = 'C'
elif a >= 60: grade = 'D'
else:       grade = 'E'

print(f'{a} 分, 等第為 {grade}')
  
```



多層的 if

```
if 條件_x:
```

```
    if 條件_y: ← 在 x 為真的狀況下, 再來判斷如果 y 就...
```

```
        程式區塊_A
```

```
    else: ← 否則...
```

```
        程式區塊_B
```

```
else:
```

```
    程式區塊_C
```

if 指令如果只是用來打成績就顯不出它的重要性了! 各位可以參考 9-17 頁的程式 9-2 看如何用 if、elif 做股票、虛擬貨幣的多空決策!

Python



實例：使用 2 層 if 檢查輸入字串

```

01 s = input("請輸入溫度:")
02 if s.count('.') > 1:                # 判斷是否超過 1 個小數點
03     print('只能有一個小數點')
04 elif s[1:].replace('.', '').isdigit(): # 判斷除了第一個字元，其餘字元去
                                         除小數點之後是否均為數字
05     if s[0] == '-' or s[0].isdigit(): # 判斷開頭字元是否為負號或數字
06         temp = float(s) ← 大括號裡放變數
07         print(f'攝氏 {temp} 度等於華氏 {(temp*9/5)+32 :+5.1f} 度')
08         print(f'華氏 {temp} 度等於攝氏 {(temp-32)*9/5 :+5.1f} 度')
09     else:
10         print("只能以數字或負號開頭")
11 else:
12     print("輸入的溫度無法轉換!")

```

執行結果

```

請輸入溫度：a          ← 輸入 a
輸入的溫度無法轉換！ ← 顯示錯誤訊息

請輸入：15.5.2        ← 重新執行，然後輸入 15.5.2
只能有一個小數點    ← 顯示提示訊息

請輸入溫度：+15      ← 重新執行，然後輸入 +15
只能以數字或負號開頭 ← 顯示提示訊息

請輸入溫度：-15.5    ← 重新執行，然後輸入 -15.5
攝氏 -15.5 度等於華氏 +4.1 度
華氏 -15.5 度等於攝氏 -85.5 度 } ← 轉換成功

```



條件運算式

傳統寫法

```
if c > 5:  
    a = 1  
else:  
    a = 2
```

條件運算式：
一行就搞定！

```
a = 1 if c>5 else 2 # 這裡 else 2 就是  
                    else a = 2 的意思
```

真 假

TIPS  條件算符 if...else 的優先順序是最低的，因此以上 c>5 會優先運算。

```
price = 100  
price = price*0.8 if price > 99 else price*0.9  
print(price)  80.0
```

Python



- 注意!!! 使用 `==` 做條件運算時, 最好不要用浮點數, 因為浮點數運算的小數點精度不易拿捏

一步一腳印

```
In [1]: a = 1.1 * 3
```

```
In [2]: b = 3.3 * 1
```

```
In [3]: a == b
```

```
Out[3]: False
```

```
In [4]: a
```

```
Out[4]: 3.3000000000000003
```

```
In [5]: b
```

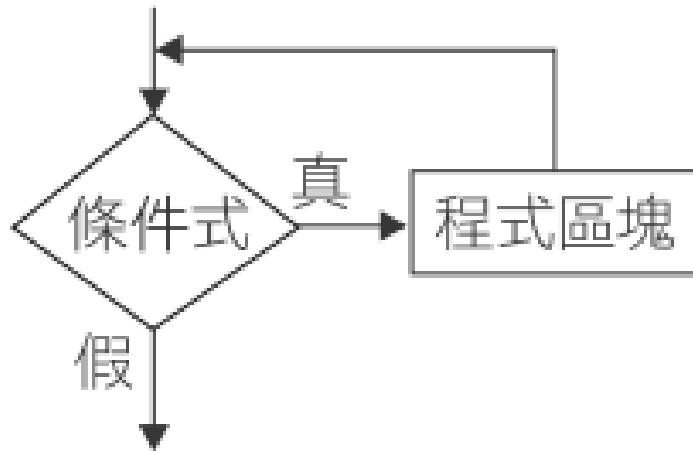
```
Out[5]: 3.3
```



3-2

while 迴圈：依條件重複執行

while 條件式：
程式區塊



Python



```
n = int(input('請輸入一個正整數:'))
k = n          # k 和 n 兩個變數名綁到同一個數值上
while(n > 1):
    n = n-1    # 再複習一下, n=n-1 會讓 n 這個變數名綁一個新的數值上
    k = k*n    # 所以現在的 n 和 k 是脫鉤的
print(k)      # 注意!為了聚焦解說 while 迴圈的用法, 本程式
              # 不處理過濾 0、負數、非整數字串的輸入
```

這裡要用 *float* 而不是 *int* 轉型, 否則
當使用者輸入有小數點的字串就會錯誤



```
i = float(input('請輸入一個正數:'))
j = 0.0
while j*j < i:
    j = j+0.00001 # 每次將 j 加一個微小的量
print(f'{i} 的平方根是: {j}')
```

Python



使用 break 與 continue

while 條件式：

....

continue

....

break

....

....

跳到下一圈的開頭
(略過後面程式敘述)

跳出迴圈

```
i = 1
while True:
    if i == 5:           # 若 i==5 就
        i += 1         # 先將 i 加 1,
        continue      # 然後直接跳到迴圈開頭
    print(i, end=' ')
    if i == 10:        # 若 i==10 就
        break          # 跳出迴圈
    i += 1
print('結束')
```



1 2 3 4 6 7 8 9 10 結束 ← 5 跳過不印了



while...else...

- while 也可以有 else, 它跟 if 的 else 很像

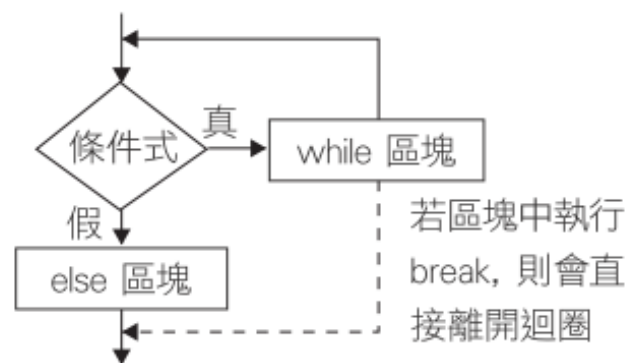
while 條件式 :

while 區塊

else :

else 區塊

...



Python



```
s = ''
while s != '喵喵':
    if s != '': print('不對喔！')
    s = input('請輸入通關密語：')
    if s == 'out':          # 如果輸入 'out' 則用 break 跳出迴圈
        break
else:
    print('恭喜你過關了')  # 正常結束時才顯示成功訊息
print('再見！')
```

輸入喵喵, 通關成功

請輸入通關密語：喵喵
恭喜你過關了
再見！

輸入 out, 放棄通關

請輸入通關密語：out
再見！

Python

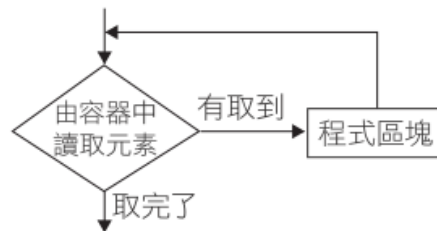


3-3

for 迴圈：走訪容器的每個元素

- for 迴圈可將容器中的元素一一讀取出來做處理, 其語法如下：

for 變數 in 容器
程式區塊



```
s = [0, 1, 2, 3]
for i in s:          # 每次由 s 中讀取一個元素，並指定給 i
    print(i, end=' ') # 印出 i 的值並空一格
```



0 1 2 3



3-3

for 迴圈：走訪容器的每個元素

- 像以上這種「一一讀取出來」的動作，就稱為走訪 (或迭代, **Iterate**)。
- 底下分別走訪字串、集合、與字典

Python



```
for a in 'abc':  
    print(a, end=' ')  
print('in str')    输出 a b c in str  
  
for a in {0,1,2}:  
    print(a, end=' ')  
print('in set')    输出 0 1 2 in set  
  
for a in {'a': 0, 'b': 1, 'c': 2}:  
    print(a, end=' ')  
print('in dict')    输出 a b c in dict
```

```
d = {'a': 0, 'b': 1, 'c': 2}  
for a in d.values():  
    print(a, end=' ')  
print('in dict.values()') 输出 0 1 2 in dict.values  
  
for a in d.items():  
    print(a, end=' ')  
print('in dict.items()') 输出 ('a', 0) ('b', 1) ('c', 2) in dict.items
```

Python



用多個變數走訪

```
d = {'a':0, 'b':1, 'c':2}
for a in d.items():
    print(a[0], a[1])
```



```
a 0
b 1
c 2
```

用索引存取：可讀
性差、索引易弄錯

```
d = {'a':0, 'b':1, 'c':2}
for key, value in d.items():
    print(key, value)
```



```
a 0
b 1
c 2
```

用多變數走訪：可讀
性佳、不需要索引

Python



使用 range() 來走訪數列

別忘了「有頭無尾！」的口訣！

```
for i in range(10): # 走訪由 0 到 10 但不包含 10 的數列
    print(i, end=' ') 輸出 0 1 2 3 4 5 6 7 8 9
```

```
for i in range(1, 11): # 走訪由 1 到 11 但不包含 11 的數列
    print(i, end=' ') 輸出 1 2 3 4 5 6 7 8 9 10
```

```
for i in range(1, 10, 2): # 只產生 1~9 的奇數數列
    print(i, end=' ') 輸出 1 3 5 7 9
```

```
for i in range(9, 0, -2): # 遞增量為負數時，m 要大於 n
    print(i, end=' ') 輸出 9 7 5 3 1
```

Python



使用 break、continue 與 for...else...

- for 和 while 一樣, 也可使用 break 來跳出迴圈, 或是用 continue 來跳到下一迴圈的開頭。

```
for i in range(1, 20):  
    if i == 5:          # 若 i==5 就略過不印  
        continue  
    print(i, end=' ')  
    if i == 10:       # 若 i==10 就跳出迴圈  
        break  
print('結束')
```

輸出 → 1 2 3 4 6 7 8 9 10 結束

- `for` 也可以搭配 `else`, 當 `for` 走訪完所有的元素後, 會接著執行 `else` 區塊然後才結束迴圈。

```
s = "asxldjf;laszdjf"
for c in s:
    if c == 'k':      # 如果找到第一個 k 就顯示找到訊息, 然後跳出迴圈
        print('找到 k 了')
        break;
else:
    print('沒找到 k') # 如果全部找完仍沒找到, 就顯示沒找到, 然後結束迴圈
```



沒找到 k



多層的 for 迴圈

```
a = [[1,4,3,2], [5,3,6], [4,7,3,8,3], [8,3]]
cnt = 0
for s in a:
    for n in s:
        if n == 3: cnt += 1
print('共有', cnt, '個 3')
```

輸出 共有 5 個 3

Python



實例：九九乘法表

```
IPython console
Console 1/A
1x1= 1  2x1= 2  3x1= 3  4x1= 4  5x1= 5  6x1= 6  7x1= 7  8x1= 8  9x1= 9
1x2= 2  2x2= 4  3x2= 6  4x2= 8  5x2=10  6x2=12  7x2=14  8x2=16  9x2=18
1x3= 3  2x3= 6  3x3= 9  4x3=12  5x3=15  6x3=18  7x3=21  8x3=24  9x3=27
1x4= 4  2x4= 8  3x4=12  4x4=16  5x4=20  6x4=24  7x4=28  8x4=32  9x4=36
1x5= 5  2x5=10  3x5=15  4x5=20  5x5=25  6x5=30  7x5=35  8x5=40  9x5=45
1x6= 6  2x6=12  3x6=18  4x6=24  5x6=30  6x6=36  7x6=42  8x6=48  9x6=54
1x7= 7  2x7=14  3x7=21  4x7=28  5x7=35  6x7=42  7x7=49  8x7=56  9x7=63
1x8= 8  2x8=16  3x8=24  4x8=32  5x8=40  6x8=48  7x8=56  8x8=64  9x8=72
1x9= 9  2x9=18  3x9=27  4x9=36  5x9=45  6x9=54  7x9=63  8x9=72  9x9=81
In [2]:
```

```
01 for i in range(1, 10):      # 外層迴圈 i 由 1 到 9
02     for j in range(1, 10): # 內層迴圈 j 由 1 到 9
03         print(f'{j}x{i}={j*i:2d}', end=' ')
04     print()
```

for 的多變數走訪

```
person = {22:('張', '天才'), 23:('王', '子帥'), 24:('陳', '美美')}  
for person_id, (lastname, firstname) in person.items():  
    print(str(person_id) + '號-' + lastname + firstname)
```



```
22號-張天才  
23號-王子帥  
24號-陳美美
```

Python



用 enumerate() 產生元素，有序號的容器

Ch3-21

- 內建函式 `enumerate` (容器, 序號起始值) 可傳回一個將元素加了序號的新容器
- 例如原來的第 0 個元素 'a' 會變成 (0, 'a'), 第 1 個元素 'b' 會變成 (1, 'b')...

```
drinks = ('紅茶', '咖啡', '果汁')
print(list(enumerate(drinks, 5)))    # 先轉為 list 再輸出內容
                                     輸出 → [(5, '紅茶'), (6, '咖啡'), (7, '果汁')] ←
for sn, drink in enumerate(drinks): # 用 for 走訪
    print(sn, drink, end=' ')        由 5 開始加序號
                                     輸出 → 0 紅茶 1 咖啡 2 果汁 ← 由 0 開始加序號
```

Python



```
drinks = ('紅茶', '咖啡', '果汁')
prices = (35, 50, 65)
matches = ('餅乾', '蛋糕', '三明治', '鬆餅') # 鬆餅不會被 zip() 走訪

for drink, price, match in zip(drinks, prices, matches):
    print(drink, price, '元, 建議甜點: '+match)
```



```
紅茶 35 元, 建議甜點: 餅乾
咖啡 50 元, 建議甜點: 蛋糕
果汁 65 元, 建議甜點: 三明治
```

Python



用 zip() 同時走訪多個容器

```
drinks = ('紅茶', '咖啡', '果汁')
prices = (35, 50, 65)
matches = ('餅乾', '蛋糕', '三明治', '鬆餅') # 鬆餅不會被 zip() 走訪

for drink, price, match in zip(drinks, prices, matches):
    print(drink, price, '元, 建議甜點: '+match)
```



```
紅茶 35 元, 建議甜點: 餅乾
咖啡 50 元, 建議甜點: 蛋糕
果汁 65 元, 建議甜點: 三明治
```



for 的容器生成式

- 生成串列的就稱為串列生成式 (List Comprehensions), 生成字典的就叫做字典生成器, 其他以此類推。各種生成式的語法如下：

從這個容器中 ... 取出元素 ... 運算後產生新元素

串列生成式：[運算式 for 變數 in 容器] ← 用中括號

集合生成式：{ 運算式 for 變數 in 容器 } ← 用大括號

字典生成式：{ 運算式_k: 運算式_v for 變數 in 容器 } ← 注意裡頭有：

```
print([i*i for i in range(1, 6)])    輸出 [1, 4, 9, 16, 25]    # 串列
print({i*i for i in range(1, 6)})   輸出 {1, 4, 9, 16, 25}    # 集合
print({i: i*i for i in range(1, 6)}) 輸出 {1: 1, 2: 4, 3: 9, 4: 16,
                                     5: 25}    # 字典
```

除了用 range() 外, 也可以用其他容器哦!



for 的容器生成式

- 如果將 `for` 生成式放在小括號中, 並不會產生 `tuple`, 而是會建立一個產生器 (`generator`), 它和 `range()` 類似, 都是一種動態容器, 也就是在需要元素時才會動態產生出元素來。

```

a = (i*i for i in range(1, 6)) ← 生成式使用小括號, 會生成一個 generator
print(type(a)) 輸出 <class 'generator'> ← 它是 generator 型別
print(tuple(a)) 輸出 (1, 4, 9, 16, 25) ← tuple() 會取用 generator 的
print(tuple(a)) 輸出 () ← 元素來建立一個 tuple
                    generator 中沒元素了, 因為只能用一次
  
```

- 在生成式中還可以加上 **if** 來做篩選

```
s = [i*i for i in range(1, 11) if i%2 == 0]  
print(s) ➡ [4, 16, 36, 64, 100]
```

Python



for 的容器生成式

- 其實在 **for** 之後可以加上任意數量的 **for** 或 **if**, 依由左而右的順序, 形成由外而內的多層巢狀結構。

```
a = [[1,2,3], [4,5,6], [7,8,9]]
print([e2 for e1 in a for e2 in e1 if e2 != 4])
```

外層 for 內層 for 最內層 if



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

↓ 相當於

```
a = [[1,2,3], [4,5,6], [7,8,9]]
b = []
for e1 in a:
    for e2 in e1:
        if e2 != 4:
            b.append(e2) # 將 e2 附加到串列 b 中
print(b)
```



3-4 例外處理

- Python 一開始執行程式時，會先全面檢查語法，如果是語法錯誤 (Syntax Error)，例如 if 之後忘了加冒號、或是 print 沒加小括號 (例如 print 1) 等，那麼只要修正語法即可。
- 如果語法沒問題，但在執行過程中發生系統無法處理的誤錯，則稱為執行期錯誤 (Runtime Error)，例如：用 int() 轉換非整數的字串、用索引讀取不存在的元素、要開啟的檔案不存在、或做了除以 0 的計算等等，此時系統會產生一個例外 Exception)



3-4 例外處理

- 如果語法沒問題,但在執行過程中發生系統無法處理的誤錯,則稱為執行期錯誤 (Runtime Error)
- 例如: 用 `int()` 轉換非整數的字串、用索引讀取不存在的元素、要開啟的檔案不存在、或做了除以 0 的計算等等,此時系統會產生一個例外 **Exception**)

Python



一步一腳印

In [1]: `int('a')` ← 用 `int()` 轉換非數字的字串, 會發生轉換資料錯誤的例外
Traceback (most recent call last): ← Traceback 的意思就是追溯程式呼叫的歷程 (最後被呼叫的會列在最後面)

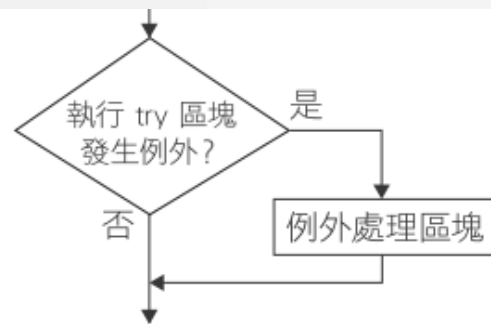
```
File "<ipython-input-2-233884bacd4e>", line 1, in <module>  
    int('a')
```

```
ValueError: invalid literal for int() with base 10: 'a'
```

↑
例外的類型

例外的原因說明

```
try :  
    可能發生例外的區塊  
except :  
    例外處理區塊
```



```
s = 'a'
```

```
try:
```

```
    i = int(s)
```

← 發生 ValueError 例外, 直接跳到 except 區塊去

```
    print('沒發生例外')
```

← 因前面發生例外, 這行不會被執行

```
except:
```

```
    print('發生例外了!')
```

```
print('程式結束')
```



發生例外了! ← 在 except 中印出的訊息(表示有處理了)

程式結束 ← except 處理完, 接下來就回歸主程式, 印出 '程式結束'

還是覺得很抽象嗎? 請參考
10-6 頁的程式 10-0 就會
有比較紮實的感覺了。

Python



捕捉特定的例外

except 特定例外類型 as 變數名稱：

```
while True:
    s = input('請輸入100的除數：')
    try:
        i = 100 / float(s)           # 將輸入字串轉為 float 做為除數
        print('100 除', s, '=', i)
        break                       # 若沒發生例外，則跳離迴圈
    except ValueError as e:         ← 捕捉值錯誤的例外，並將例外存到 e
        print('發生 ValueError 例外：', e) # 看看 e 的內容
    except ZeroDivisionError:      ← 捕捉除以零的例外，省略 as
        print('發生 ZeroDivisionError 例外')
    except:                         ← 還可用空的 except: 來捕捉所有其他的例外，但必須放在最後
        print('其他例外')
    print('進入下一迴圈')          # 抓到不正常的輸入，所以進入下一迴圈要求再輸入
print('程式正常結束')
```





請輸入100的除數：a ← 輸入 a

發生 ValueError 例外：`could not convert string to float: 'a'`

進入下一迴圈

`e` 的訊息內容

請輸入100的除數：0 ← 輸入 0

發生 ZeroDivisionError 例外

進入下一迴圈

請輸入100的除數：5 ← 輸入 5

100 除 5 = 20.0

程式正常結束

你怎知道有 ValueError
這種東西呢？



是內建的啦！Python 有近
30 個內建的例外，詳見官網：
[docs.python.org/3/library/
exceptions.html](https://docs.python.org/3/library/exceptions.html)



Python



try...except...else...finally...

try

可能發生例外的區塊

except 例外類型_A as e :

發生「例外類型_A」時要執行的區塊

except :

發生前面都未捕捉到的例外時, 要執行的區塊

else :

未發生例外時要執行的區塊

finally :

無論如何都會執行的區塊

Python



```

while True:
    s = input('請輸入100的除數:')
    try:
        i = 100 / float(s)
    except ValueError:          # 只捕捉 ValueError 的例外
        print('發生 ValueError 例外')
    else:                       # 未發生例外時要執行的區塊
        print('100 除', s, '=', i)
        break
    finally:                     # 不管是否發生例外都會執行的區塊
        print('你輸入的值是', s)
    print('進入下一迴圈')
print('程式正常結束')

```



請輸入100的除數:0 ← 輸入 0, 會發生除以零的例外

你輸入的值是 0 ← 因例外沒被捕捉到, 所以會先執行 *finally* 區塊印出此訊息

Traceback (most recent call last): ← 然後一層層往上傳, 由 Python 來顯示
錯誤訊息並結束程式 (非正常結束程式)

```
File "<ipython-input-18-82300098aeae>", line 1, in <module>
```

```
.....
```

```
File "D:/0Book/FT700/範例/untitled3.py", line 16, in <module>
```

```
    i = 100 / float(s)
```

```
ZeroDivisionError: float division by zero
```

Python

