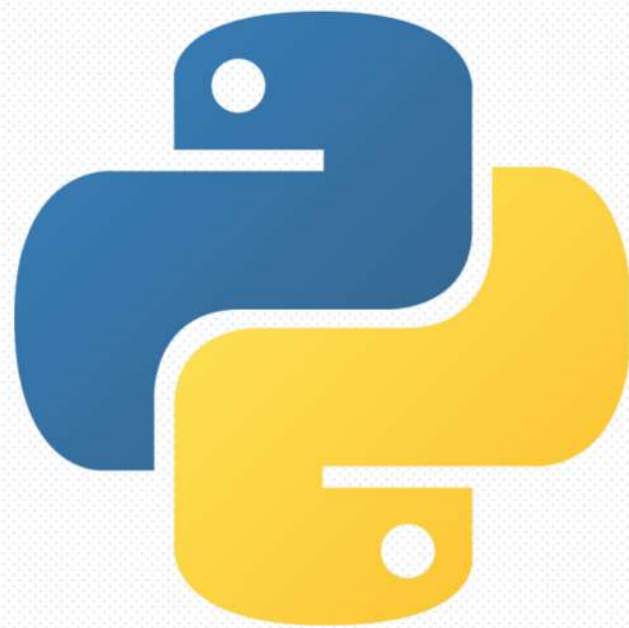


例外處理

Exception Handling



Python Fundamental



認識例外

- ◆ 大部分執行中的錯誤，Python 直譯器 (interpreter) 會以例外 (exception) 的方式來中斷程式的執行。
- ◆ 我們在很多實際情況下需要自行控制可能會產生例外的程式碼。因為例外並不全然是程式的邏輯錯誤，
例如程式中打算開啟檔案，然而實際檔名並不存在。在這種情況下，我們需要的是例外發生後的處理動作，而非中止程式的執行。
- ◆ 凡是可能會產生例外的程式碼，Python 利用 try - except 陳述 (try-except statement) 讓程式設計師自行處理例外。
- ◆ try-except 為關鍵字 (keyword) 之一，專門來例外處理(exception handling)。



程式設計錯誤類型

◆ 程式設計錯誤的類型

- 語法錯誤 (syntax error)
- 執行期間錯誤 (runtime error)
- 邏輯錯誤 (logic error)

```
File Edit Shell Debug Options Window Help
>>>
>>>
>>> if x > y
SyntaxError: invalid syntax
>>>
```

Ln: 38 Col: 4

```
File Edit Shell Debug Options Window Help
>>>
>>> prin("Hello, World!")
Traceback (most recent call last):
  File "<pyshell#22>", line 1, in <module>
    prin("Hello, World!")
NameError: name 'prin' is not defined
>>>
```

Ln: 46 Col: 4

```
Python 3.5.2 Shell
File Edit Shell Debug Options Window Help
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18) [MSC v
.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print("hello, World!")
SyntaxError: EOL while scanning string literal
>>> print("Hello, World!")
SyntaxError: unexpected indent
>>>
```

Ln: 9 Col: 4

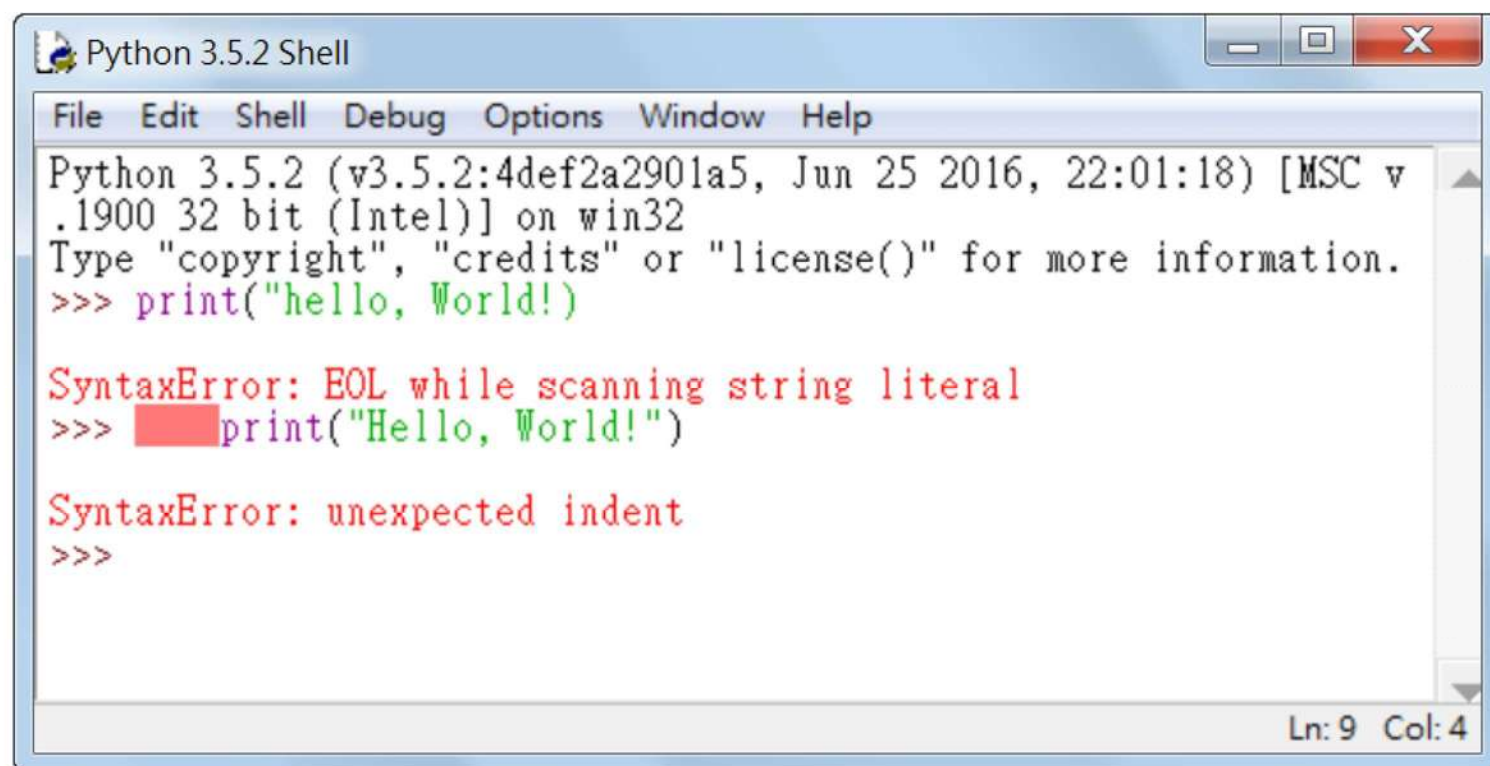
```
Python 3.5.2 Shell
File Edit Shell Debug Options Window Help
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18) [MSC v
.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> X = 1 # 將變數X的值設定為1
>>> Y = 0 # 將變數Y的值設定為0
>>> print(X / Y) # 印出變數X除以變數Y的結果
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    print(X / Y) # 印出變數X除以變數Y的結果
ZeroDivisionError: division by zero
>>>
```

Ln: 10 Col: 4



語法錯誤 (syntax error)

- ◆ 這是在寫程式時最容易發生的錯誤。每個程式語言都有其專屬語法，如果誤用，就會發生錯誤，例如不當的縮排，拼字錯誤，漏掉必要的符號等。
- ◆ 針對語法錯誤，Python直譯器會顯示錯誤之處及其原因，依提示修正即可。



```
Python 3.5.2 Shell
File Edit Shell Debug Options Window Help
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18) [MSC v
.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print("hello, World!)
SyntaxError: EOL while scanning string literal
>>> print("Hello, World!")
SyntaxError: unexpected indent
>>>
```

Ln: 9 Col: 4



執行期間錯誤 (runtime error)

- ◆ 這是在程式執行期間發生的錯誤。會產生這樣的錯誤，大多不是語法問題，而是一個看起來正確，但執行上發生錯誤的程式碼。
- ◆ 以下圖為例，其出錯原因在於除數不得為0的限制，導致程式終止執行。
- ◆ 針對執行期間錯誤，只要依錯誤訊息進行修正即可。

```
Python 3.5.2 Shell
File Edit Shell Debug Options Window Help
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18) [MSC v
.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> X = 1                # 將變數X的值設定為1
>>> Y = 0                # 將變數Y的值設定為0
>>> print(X / Y)         # 印出變數X除以變數Y的結果
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    print(X / Y)          # 印出變數X除以變數Y的結果
ZeroDivisionError: division by zero
>>>
```

Ln: 10 Col: 4

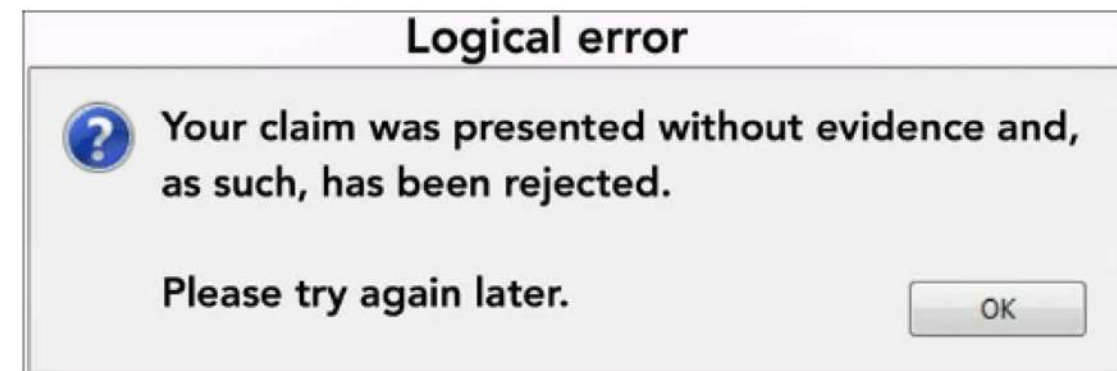


邏輯錯誤 (logical error)

- ◆ 這是在使用程式時發生的錯誤，例如輸入不符合要求的資料，但程式沒有設計如何處理。
- ◆ 邏輯錯誤是最難修正的錯誤類型，因為不容易找出導致錯誤的真正原因，但還是可以從執行結果不符合預期來加以判斷。

LOGICAL ERROR

- * *A logic error (or logical error) is a mistake in a program's source code that results in incorrect or unexpected behavior. It is a type of runtime error that may simply produce the wrong output or may cause a program to crash while running.*
- * *Many different types of programming mistakes can cause logic errors. For example, assigning a value to the wrong variable may cause a series of unexpected program errors*





認識例外

◆ 例外的類型

系統會根據不同的錯誤丟出不同的例外。

- ImportError
- IndexError
- MemoryError
- NameError
- OverflowError
- RuntimeError
- SyntaxError
- IndentationError
- ZeroDivisionError
- SystemError
- TypeError
- ValueError
- ZeroDivisionError
- ConnectionError、ConnectionAbortedError、
ConnectionRefusedError、ConnectionResetError
- FileNotFoundError
- TimeoutError



例外範例

```
X = eval(input("請輸入被除數X : "))
```

```
Y = eval(input("請輸入除數Y : "))
```

```
Z = X / Y
```

```
print("X除以Y的結果等於", Z)
```

```
請輸入被除數X : 100  
請輸入除數Y : 10  
X除以Y的結果等於 10.0
```

```
>>>
```

```
===== RESTART: J:\Jean\Python3\Samples\Ch09\except1.py =====
```

```
請輸入被除數X : 100  
請輸入除數Y : 0
```

```
Traceback (most recent call last):
```

```
  File "J:\Jean\Python3\Samples\Ch09\except1.py", line 3, in <module>
```

```
    Z = X / Y
```

```
ZeroDivisionError: division by zero
```

```
>>>
```

```
===== RESTART: J:\Jean\Python3\Samples\Ch09\except1.py =====
```

```
請輸入被除數X : 100  
請輸入除數Y : a
```

```
Traceback (most recent call last):
```

```
  File "J:\Jean\Python3\Samples\Ch09\except1.py", line 2, in <module>
```

```
    Y = eval(input("請輸入除數Y : "))
```

```
  File "<string>", line 1, in <module>
```

```
NameError: name 'a' is not defined
```

```
>>>
```

```
===== RESTART: J:\Jean\Python3\Samples\Ch09\except1.py =====
```

```
請輸入被除數X : 100,  
請輸入除數Y : 1
```

```
Traceback (most recent call last):
```

```
  File "J:\Jean\Python3\Samples\Ch09\except1.py", line 3, in <module>
```

```
    Z = X / Y
```

```
TypeError: unsupported operand type(s) for /: 'tuple' and 'int'
```

```
>>>
```




例外範例

```
numbers = input('輸入數字 (空白區隔): ').split(' ')
print('平均', sum(int(number) for number in numbers) / len(numbers))
```

```
輸入數字 (空白區隔): 10 20 30 40
平均 25.0
```

```
輸入數字 (空白區隔): 10 20 3o 40
Traceback (most recent call last):
  File "C:/workspace/exceptions/average.py", line 2, in <module>
    print('平均', sum((int(number) for number in numbers)) / len(numbers))
  File "C:/workspace/exceptions/average.py", line 2, in <genexpr>
    print('平均', sum((int(number) for number in numbers)) / len(numbers))
ValueError: invalid literal for int() with base 10: '3o'
```



例外捕捉

- ◆ 例外的捕捉可以使用 `try ... except` 區塊進行處理。
- ◆ 在 `try suite` 中的程式碼會被 Python 首先執行。一旦發生狀況，產生例外，執行流程就會跳離例外發生點，轉由 `except suite` 中的程式碼來進行後續的處理。

```
try:  
    try suite  
except:  
    except suite
```

```
>>> try:  
    a=0  
    b.split() ←  
    b=0 ←  
except:  
    c=0  
  
>>> a  
0  
>>> b  
Traceback (most recent call last):  
  File "<pyshell#23>", line 1, in <module>  
    b  
NameError: name 'b' is not defined  
>>> c  
0
```

此處發生例外

此運算未被執行

跳至except區塊



try...except

- ◆ `except` 之後可以使用數組(tuple) 指定多個物件，也可以有多個 `except`。
- ◆ 如果沒有指定 `except` 後的物件型態，表示捕捉所有引發的物件。

```
import time

try:
    time.sleep(10) # 模擬一個耗時流程
    num = int(input('輸入整數：'))
    print('{0} 爲 {1}'.format(num, '奇數' if num % 2 else '偶數'))
except ValueError:
    print('請輸入阿拉伯數字')
except (EOFError, KeyboardInterrupt):
    print('使用者中斷程式')
except:
    print('其他程式例外')
```



更多的控制敘述

◆ `try...except` 區塊還支援 `else` 和 `finally` 敘述句可以提供更有彈性的控制

- 若 `try` 區塊中並未引發任何例外，則 `else` 區塊則會在 `try` 區塊執行完後被執行，反之則不執行。
- `finally` 區塊不論例外發生與否、捕捉與否，最後一定會執行的敘述。

```
try :  
    try suite  
except :  
    except suite  
else :  
    else suite  
finally :  
    finally suite
```




try...except

- ◆ 我們可以使用try...except將前面的例子改寫成如下，令它捕捉ZeroDivisionError (除數為0的除法運算) 和其它例外，然後針對不同的例外做不同的處理。

```
01 try:
02     X = eval(input("請輸入被除數X : "))
03     Y = eval(input("請輸入除數Y : "))
04     Z = X / Y
05 except ZeroDivisionError:
06     print("除數不得為0")
07 except Exception as e1:
08     print(e1.args)
09 else:
10     print("沒有捕捉到例外！X除以Y的結果等於", Z)
11 finally:
12     print("離開try...except區塊")
```

```
請輸入被除數X : 100
請輸入除數Y : 10
沒有捕捉到例外！X除以Y的結果等於 10.0
離開try...except區塊
```

```
>>>
===== RESTART: J:\Jean\Python3\Samples\Ch09\except2.py
```

```
請輸入被除數X : 100
請輸入除數Y : 0
除數不得為0
離開try...except區塊
```

```
>>>
===== RESTART: J:\Jean\Python3\Samples\Ch09\except2.py
```

```
請輸入被除數X : 100
請輸入除數Y : a
('name 'a' is not defined',)
離開try...except區塊
```

```
>>>
===== RESTART: J:\Jean\Python3\Samples\Ch09\except2.py
```

```
請輸入被除數X : 100,
請輸入除數Y : 1
('unsupported operand type(s) for /: 'tuple' and 'int',)
離開try...except區塊
>>>
```


Q & A