

Week 2-AEDT操作錄製與修改



範例解說：將物件複製到選定的位置

定義需求

- 需求
 - 需要將物件複製到選定的位置
- 輸入
 - 那些物件？
 - 位置該如何表示？
- 輸出
 - 物件複製到選定的座標位置



程式開發流程

- 在HFSS當中原點任意建立多個物件
- 選取物件並啟動錄製
- 執行”複製”的動作並結束錄製
- 刪除註解，或註解當中的中文字
- 將SetActiveProject(“”)改成GetActiveProject，將SetActiveDesign(“”)改成GetActiveDesign()
- 將座標放在list of tuple當中
- 加入迴圈並修改DuplicateAlongLine()的座標點，測試代碼
- 加入選取oEditor.GetSelections()
- 將座標移到CSV當中，並利用open()讀取值到list of tuple

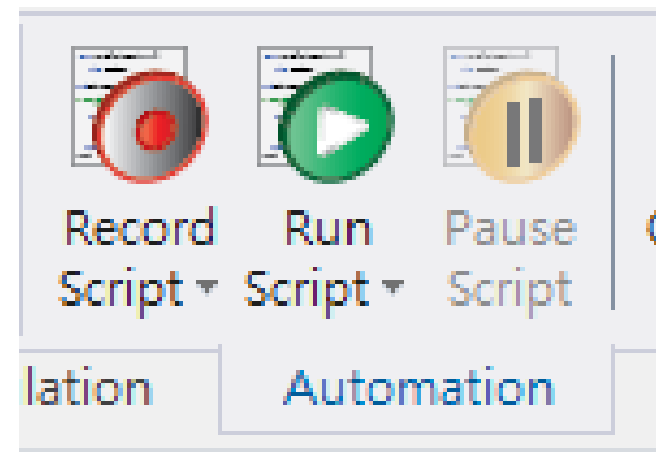
程式改善

- 針對未選擇物件的的狀況加入，加入警告訊息
- 加入選擇**CSV**檔對話框的功能
- 改為中文警告訊息
- 加入註解

錄製及修改

錄製腳本對於開發AEDT自動化程式有相對大的幫助。由於腳本的函式碼對應使用者的操作，減少了許多查詢及試錯的時間。要注意的是腳本的檔頭如果有中文字符，必須先移除。project與design的部分也要改成GetActiveProject()及GetActiveDesign()。使其可以使用在其他的專案。

如果要加上迴圈及判斷式，最好先將錄製的函式進行封裝簡化其輸出輸入。要留意保持正確的輸入參數型別。可以加上AddWarningMessage(str(X))適時的將執行過程中的變數值(X)輸出到訊息視窗，這有助於除錯。



完成的程式碼

```
1 # coding=UTF-8
2 # -----
3 # Script Recorded by ANSYS Electronics Desktop Version 2020.1.0
4 # 9:40:07 Jul 02, 2020
5 # -----
6 import ScriptEnv
7 import sys
8 import clr
9 clr.AddReference("System.Windows.Forms")
10 from System.Windows.Forms import DialogResult, OpenFileDialog
11 ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")
12 oDesktop.RestoreWindow()
13 oProject = oDesktop.GetActiveProject()
14 oDesign = oProject.GetActiveDesign()
15 oEditor = oDesign.SetActiveEditor("3D Modeler")
16
17 if oEditor.GetSelections():
18     dialog = OpenFileDialog()
19     dialog.Filter = "text files (*.txt)|*.txt"
20
21     if dialog.ShowDialog() == DialogResult.OK:
22         txt_path = dialog.FileName
23         AddWarningMessage(txt_path)
24     else:
25         sys.exit()
26     with open(txt_path) as f:
27         text = f.readlines()
28     w = []
29     for i in text:
30         y = i.split(',')
31         w.append([float(j) for j in y])
```

```
32
33 for x, y, z in w:
34
35     oEditor.DuplicateAlongLine(
36         [
37             "NAME:Selections",
38             "Selections:=", ', '.join(oEditor.GetSelections()),
39             "NewPartsModelFlag:=", "Model"
40         ],
41         [
42             "NAME:DuplicateToAlongLineParameters",
43             "CreateNewObjects:=", True,
44             "XComponent:=", "{mm}".format(x),
45             "YComponent:=", "{mm}".format(y),
46             "ZComponent:=", "{mm}".format(z),
47             "NumClones:=", "2"
48         ],
49         [
50             "NAME:Options",
51             "DuplicateAssignments:=", False
52         ],
53         [
54             "CreateGroupsForNewObjects:=", False
55         ])
56 else:
57     oDesktop.ClearMessages("", "", 2)
58     AddErrorMessage("記得先選擇物件!")
```

自動化程式開發流程

找出模擬工作的瓶頸並定義題目

首先試著在問題中加入明確的輸入及輸出。舉例來說，“如何更快的畫出3D結構”就是屬於一個大問題，沒有明確的輸入及輸出。加入輸出入的問題會像是這樣子：

- “如何給定一條曲線（輸入），沿著線產生等間距的via（輸出）”，或是
- “如何給定座標點（輸入），將物件（輸入）複製到座標點上面（輸出）”或是，
- “選擇焊片（輸入），如何在上面自動產生錫球（輸出）”。

加入了輸入及輸出的描述讓整個問題變得具體也更容易理解。

分析題目並拆解為更小的問題

接下來要試著將上面的問題變成是（輸入—輸出）—（輸入—輸出）—（輸入—輸出）的格式，拆解問題的同時也是在回答問題，如同抽絲剝繭一般，這時候已經開始進入了程式開發的工作。舉例來說，“如何給定座標點”變成是“如何將儲存在csv檔案的座標點讀到陣列當中”以及“如何將選擇的物件輸出名稱”和“如何將輸出物件根據陣列的座標位置進行複製”。

要注意的是，除非你已經是有經驗的開發者，否則這些問題並不是你在程式開發初期就可以想的出來的，而是在程式碼開發過程當中，透過不斷嘗試去摸索出來的。過程當中答案也可能會一變再變，不斷嘗試不同的小程式碼片段去解決其中一個小問題，直到找到答案為止。

將所有的代碼拼湊出來

等到把所有的拼圖碎片找齊，最後便是拼圖的工作。這個階段不斷進行拼湊程式碼，測試，除錯的工作循環。就像拼圖一樣，先拼出邊邊角角比較容易的部分。將四個邊角完成之後在設法將這些大塊連結起來。

開發自動化程式很少是一氣呵成，一筆而就的。實際的過程更像是摸著石子過河，走到半途退回去另尋他路更是常有的事。就算是百轉千折，只要最終能到達對岸就是成功的達成任務，這也是學習程式開發的必經階段。至於對有開發經驗的設計師來說，可以快速的找出最短路徑，加上手上現有的工具，很快就能完成開發的工作。



Python基本語法簡介(2/4)

Python - Basic Operators

Python - Decision Making

Python - Loops

Python - Numbers

Python - Strings

/ 常用內建函式

- `abs()`
- `complex()`
- `dict()`
- `dir()`
- `enumerate()`
- `filter()`
- `float()`
- `int()`
- `len()`
- `list()`
- `map()`
- `max()`
- `min()`
- `open()`
- `pow()`
- `print()`
- `range()`
- `round()`
- `str()`
- `sum()`
- `tuple()`
- `type()`
- `zip()`

Built-in Functions				
<code>abs()</code>	<code>delattr()</code>	<code>hash()</code>	<code>memoryview()</code>	<code>set()</code>
<code>all()</code>	<code>dict()</code>	<code>help()</code>	<code>min()</code>	<code>setattr()</code>
<code>any()</code>	<code>dir()</code>	<code>hex()</code>	<code>next()</code>	<code>slice()</code>
<code>ascii()</code>	<code>divmod()</code>	<code>id()</code>	<code>object()</code>	<code>sorted()</code>
<code>bin()</code>	<code>enumerate()</code>	<code>input()</code>	<code>oct()</code>	<code>staticmethod()</code>
<code>bool()</code>	<code>eval()</code>	<code>int()</code>	<code>open()</code>	<code>str()</code>
<code>breakpoint()</code>	<code>exec()</code>	<code>isinstance()</code>	<code>ord()</code>	<code>sum()</code>
<code>bytearray()</code>	<code>filter()</code>	<code>issubclass()</code>	<code>pow()</code>	<code>super()</code>
<code>bytes()</code>	<code>float()</code>	<code>iter()</code>	<code>print()</code>	<code>tuple()</code>
<code>callable()</code>	<code>format()</code>	<code>len()</code>	<code>property()</code>	<code>type()</code>
<code>chr()</code>	<code>frozenset()</code>	<code>list()</code>	<code>range()</code>	<code>vars()</code>
<code>classmethod()</code>	<code>getattr()</code>	<code>locals()</code>	<code>repr()</code>	<code>zip()</code>
<code>compile()</code>	<code>globals()</code>	<code>map()</code>	<code>reversed()</code>	<code>__import__()</code>
<code>complex()</code>	<code>hasattr()</code>	<code>max()</code>	<code>round()</code>	

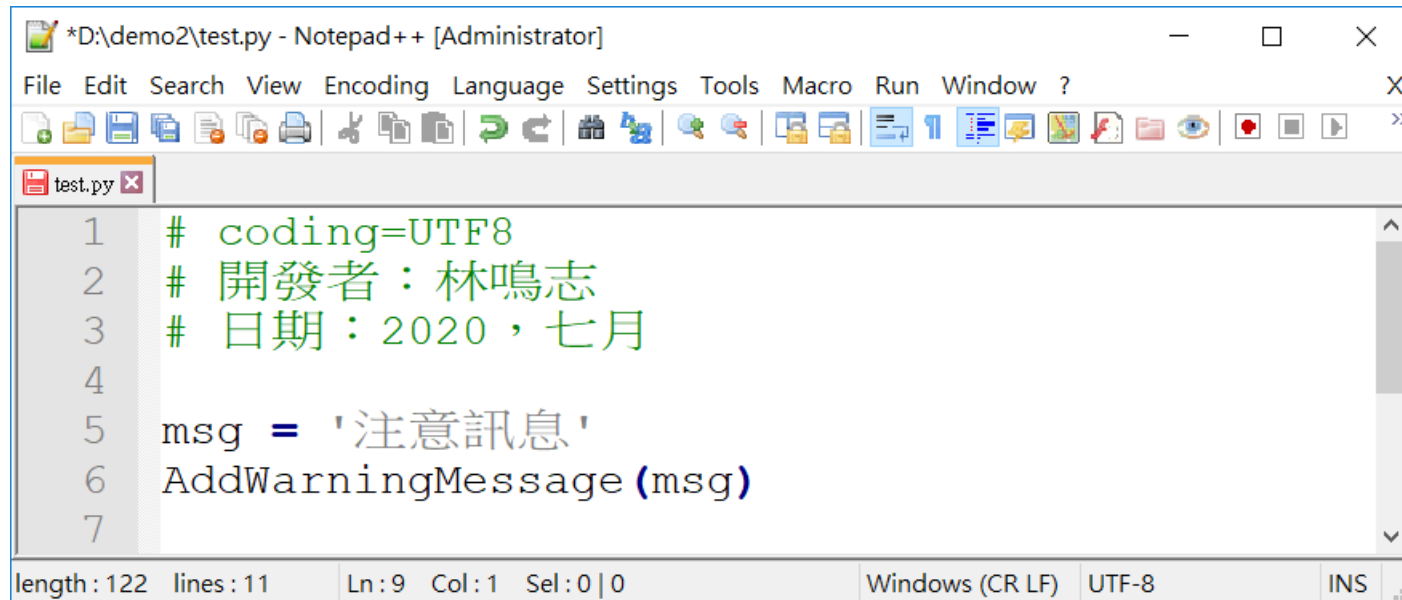
 `f = open()` 與 `with open()` as `f`:

```
>>>
>>> f = open('d:/demo2/locations.csv')
>>> text = f.readlines()
>>> text
['1, 0 ,0\n', '0, 1, 0\n', '0,0,1\n', '2,0,0\n', '3,0,0\n']
>>> f.close()
>>> with open('d:/demo2/locations.csv') as f:
...     text2 = f.readlines()
...
>>> text2
['1, 0 ,0\n', '0, 1, 0\n', '0,0,1\n', '2,0,0\n', '3,0,0\n']
>>> |
```

map(), filter() 與 list comprehension

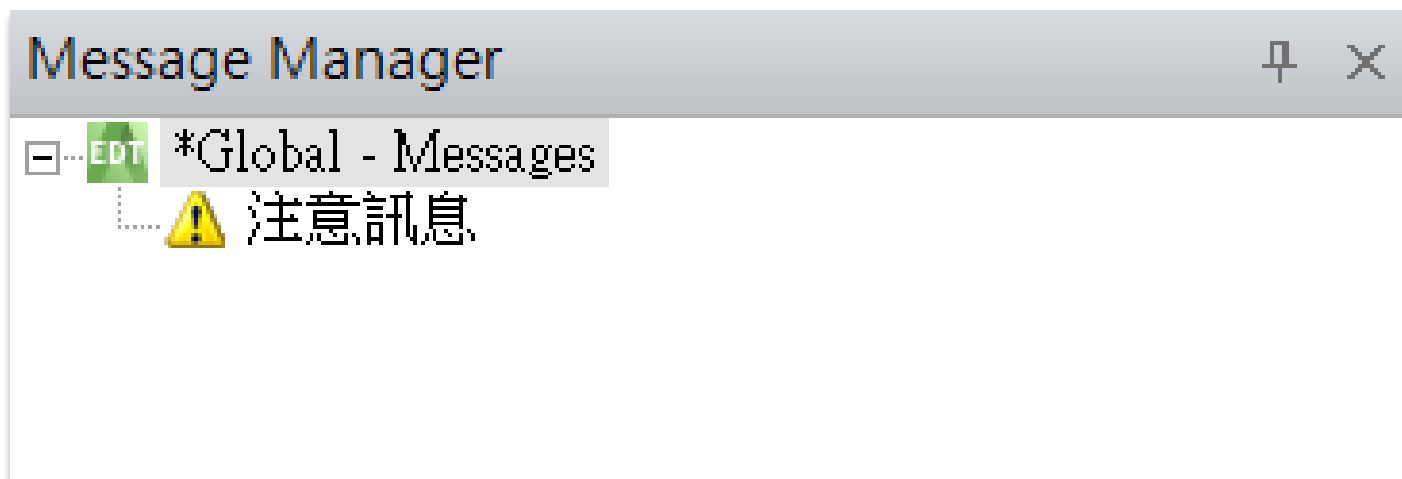
```
>>> x = ['-2', '3', '5', '-6', '1', '-4', '0']
>>> y = map(float, x)
>>> y
[-2.0, 3.0, 5.0, -6.0, 1.0, -4.0, 0.0]
>>> def greater_than_zero(num):
...     return num > 0
...
>>> z = filter(greater_than_zero, y)
>>> z
[3.0, 5.0, 1.0]
>>> w = [float(i) for i in x if float(i)>0]
>>> w
[3.0, 5.0, 1.0]
>>>
```


第一行加入 #coding=UTF8以支援中文註解及訊息輸出



```
*D:\demo2\test.py - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Tools Macro Run Window ?
test.py x
1 # coding=UTF8
2 # 開發者：林鳴志
3 # 日期：2020，七月
4
5 msg = '注意訊息'
6 AddWarningMessage(msg)
7

length: 122 lines: 11 Ln: 9 Col: 1 Sel: 0 | 0 Windows (CR LF) UTF-8 INS
```



/ IronPython 啟動開檔瀏覽對話框

```
import sys
import clr
clr.AddReference("System.Windows.Forms")

from System.Windows.Forms import DialogResult, OpenFileDialog
dialog = OpenFileDialog()
dialog.Filter = "text files (*.txt)|*.txt"

if dialog.ShowDialog() == DialogResult.OK:
    txt_path = dialog.FileName
    AddWarningMessage(txt_path)
else:
    pass
```

字串的索引與切片

切片是相當方便的字串處理工具之一，利用中括號來取出部分字串或倒置

- `x[start:stop:step]`

```
In [8]: x='channel.s4p'
```

```
In [9]: x[0]
```

```
Out[9]: 'c'
```

```
In [10]: x[1:7]
```

```
Out[10]: 'hannel'
```

```
In [11]: x[-3:]
```

```
Out[11]: 's4p'
```

```
In [12]: x[:-4]
```

```
Out[12]: 'channel'
```

```
In [13]: x[::-1]
```

```
Out[13]: 'p4s.lennahc'
```

```
In [14]: x[::2]
```

```
Out[14]: 'canlsp'
```

Escape Characters

```
>>> print('Hello World')
Hello World
>>> print('Hello\nWorld')
Hello
World
>>> print('Hello\\World')
Hello\World
>>> print('1\t2\t3')
1      2      3
>>> |
```

Code	Result	Try it
\'	Single Quote	Try it »
\\	Backslash	Try it »
\n	New Line	Try it »
\r	Carriage Return	Try it »
\t	Tab	Try it »
\b	Backspace	Try it »
\f	Form Feed	
\ooo	Octal value	Try it »
\xhh	Hex value	Try it »

常用字串運算

```
>>> x='Channel.s36p'
>>> x.split('.')
['Channel', 's36p']
>>> 'd:/demo/' + x
'd:/demo/Channel.s36p'
>>> x.replace('.s', '_new.s')
'Channel_new.s36p'
>>> x.upper()
'CHANNEL.S36P'
>>> x.lower()
'channel.s36p'
>>> '.s' in x
True
>>> len(x)
12
>>> dir(x)
['Chars', 'Clone', 'Compare', 'CompareOrdinal', 'CompareTo', 'Concat', 'Contains', 'Copy', 'CopyTo',
'Empty', 'EndsWith', 'Equals', 'Format', 'GetEnumerator', 'GetHashCode', 'GetType', 'GetTypeCode',
'IndexOf', 'IndexOfAny', 'Insert', 'Intern', 'IsInterned', 'IsNormalized', 'IsNullOrEmpty',
'IsNullOrWhiteSpace', 'Join', 'LastIndexOf', 'LastIndexOfAny', 'Length', 'MemberwiseClone', 'Normalize',
'PadLeft', 'PadRight', 'ReferenceEquals', 'Remove', 'Replace', 'Split', 'StartsWith', 'Substring',
'ToArray', 'ToLower', 'ToLowerInvariant', 'ToString', 'ToUpper', 'ToUpperInvariant', 'Trim',
'TrimEnd', 'TrimStart', '__add__', '__class__', '__contains__', '__delattr__', '__doc__', '__eq__',
'__format__', '__ge__', '__getattr__', '__getitem__', '__getnewargs__', '__getslice__', '__gt__',
'__hash__', '__init__', '__le__', '__len__', '__lt__', '__mod__', '__mul__', '__ne__', '__new__',
'__radd__', '__reduce__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '__setattr__',
'__sizeof__', '__str__', '__subclasshook__', '_formatter_field_name_split', '_formatter_parser',
'capitalize', 'center', 'count', 'decode', 'encode', 'endswith', 'expandtabs', 'find', 'format',
'index', 'isalnum', 'isalpha', 'isdecimal', 'isdigit', 'islower', 'isnumeric', 'isspace', 'istitle',
'isunicode', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'partition', 'replace', 'rfind', 'rindex',
'rjust', 'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines', 'startswith', 'strip', 'swapcase',
'title', 'translate', 'upper', 'zfill']
```

/ 分解與組合字串

```
>>> x = ['box1', 'cylinder1', 'sphere1']
>>> y = ','.join(x)
>>> print(y)
box1,cylinder1,sphere1
>>> type(y)
<type 'str'>
>>> z = y.split(',')
>>> z
['box1', 'cylinder1', 'sphere1']
>>> |
```

邏輯判斷if, else, elif

Python 提供了 if 、 else 、 elif 這三種語法來協助我們實現各種條件判斷和流程控制。Python 程式語言是一行一行執行的，所以當我們想要所寫的程式在某些條件下跳過某幾行敘述、不再照單全收的時候，就可以使用條件判斷。也就是說，如果要讓程式可以自動檢查所處理資料的內容，而且根據資料內容來決定是否執行某一個敘述或指令，那就需要用到條件判斷式來控制流程。

求解公倍數

```
for i in range(1000):  
    if i%21 == 0 and i%24 == 0:  
        print(i)
```

三元運算子(ternary operator)

- 三元運算子將變量指定及判斷整合在一行當中
- 好處是程式碼比較簡短。

- 一般

if $x > y$:

$z = x$

else:

$z = y$

- 三元運算子

$z = x$ if $x > y$ else y

/ 迴圈控制 for...in...:

- 單純的for迴圈會依序對列表當中每個元素做運算，直到所有元素都完成處理。加上break直接脫離迴圈，而continue敘述則是中斷這個元素後續工作進到下一個元素。這兩個敘述通常會搭配if-else使用。這裡用兩個例子來說明break以及continue的使用時機。

```
In [40]: for i in range(1, 1000):  
...:     if i%21==0 and i%24==0:  
...:         print(i)  
...:         break  
...:  
168
```

```
In [42]: for i in range(1, 1000):  
...:     if i%21==0 and i%24==0:  
...:         continue  
...:     else:  
...:         print(i)
```

/ 拆包

```
>>> x, y, z = (1, 2, 3)
>>> x
1
>>> y
2
>>> z
3
>>> shopping_list = [('Apple', 5.4, 6), ('Banana', 3.8, 12), ('lemon', 1.2, 30)]
>>> for name, price, number in shopping_list:
...     print(name+' : $'+str(price*number))
...
Apple: $32.4
Banana: $45.6
lemon: $36.0
>>>
```

函數range()與enumerate()

這兩個函數經常與for合併使用

- 函數range()產生等差的整數數列。
- 當需要index來操作list時可使用range()來建立index。
- enumerate多用於在for循環中得到計數，利用它可以同時獲得索引和值，即需要index和value值的時候可以使用enumerate。

```
In [44]: list(range(10))  
Out[44]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
In [45]: list(range(2, 10))  
Out[45]: [2, 3, 4, 5, 6, 7, 8, 9]
```

```
In [46]: list(range(2, 10, 2))  
Out[46]: [2, 4, 6, 8]
```

```
In [50]: names = ['John', 'Mary', 'Anna', 'Ken']
```

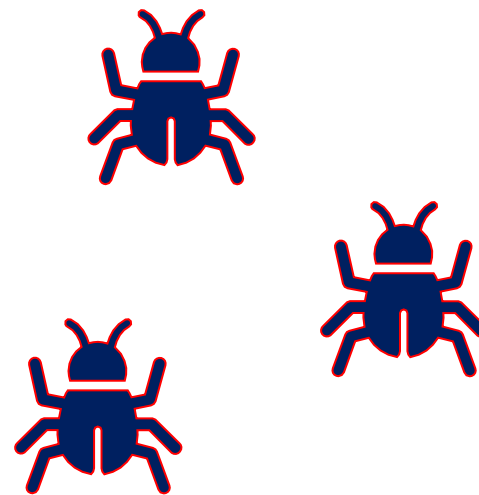
```
In [51]: for n, i in enumerate(names):  
...:     print(str(n) + ':' + i)  
...:
```

```
0:John  
1:Mary  
2:Anna  
3:Ken
```

除錯 (Debug)

寫程式是不斷的嘗試及除錯，程式錯誤基本上可分成三種：

- Syntax Error(語法錯誤)
- Run Time Error(執行時期錯誤)
- Logic Error(邏輯錯誤)



一般的IDE環境在編寫階段即可偵測Syntax Error。Run Time Error通常是輸入資料的瑕疵所造成，通常是在執行過程會被檢查出來。可以透過例外處理來解決。至於Logic Error則是輸出結果不如預期，這個錯誤要修正則是困難許多。一般需要設定斷點觀察變量的變動來追蹤可能的邏輯錯誤。在AEDT當中只能透過輸出變數值或log檔來協助除錯。

/ Syntax Error (語法錯誤)

- 圖一的程式碼是要計算x當中所有數字的總和。在第10行for迴圈的最後少了冒號，這違反了Python的語法規則，因此編輯器在行數10之前顯示了叉叉符號，提醒開發者語法錯誤。
- 如果不修正直接執行，則會出現如圖二的語法錯誤訊息
- 在第十行for迴圈之後補上了冒號之後(如圖三)，叉叉符號消失，代表沒有語法錯誤。

```
8 x = [1, 2, 3, 'John', 5, 6, 7]
9 total = 0
10 for i in x
11     total += i
12
13 print(total)
14
```

```
File "C:/Users/mlin/AppData/Roaming/SPB_Data/untitled16.py", line 10
    for i in x
              ^
SyntaxError: invalid syntax
```

```
8 x = [1, 2, 3, 'John', 5, 6, 7]
9 total = 0
10 for i in x:
11     total += i
12
13 print(total)
14
```

/ Run Time Error(執行時期錯誤)

- 此時再次執行程式，另一個錯誤出現(圖左)
- 這個錯誤是因為x當中混入了字串'John'，而字串是無法與數字相加的。由於這是無法事先預期的錯誤且在執行時期才發生，所以又稱作例外。這種錯誤可以用例外處理機制來解決，程式碼修改如圖右：

```
8 x = [1, 2, 3, 'John', 5, 6, 7]
9 total = 0
10 for i in x:
11     total += i
12
13 print(total)
14
```

`TypeError: unsupported operand type(s) for +=: 'int' and 'str'`

```
8 x = [1, 2, 3, 'John', 5, 6, 7]
9 total = 0
10 for i in x:
11     try:
12         total += i
13     except:
14         pass
15 print(total)
16
```

```
In [12]: runfile('C:/Users/mlin/AppData/Roaming/SPB_Data/untitled16.py',
wdir='C:/Users/mlin/AppData/Roaming/SPB_Data')
24
```

Logic Error (邏輯錯誤)

- 邏輯錯誤指的是代碼可以完成執行而不會返回錯誤訊息，然而執行結果卻不符合預期。左圖的程式碼目的是求取奇數項的和，即 $1+3+5+7$ ，總和應為16。然而計算結果卻是12。
- 原因是Python的引數是從0開始，所以程式碼必須改成`sum(x[0::2])`，才能得到預期的結果16。邏輯錯誤一般來說要遠比語法錯誤及執行時期錯誤難處理的多。

```
8 x = [1, 2, 3, 4, 5, 6, 7]
9 total = 0
10 print(sum(x[1::2]))
11
```

```
In [15]: runfile('C:/Users/mlin/AppData/Roaming/SPB_Data/untitled16.py',
               wdir='C:/Users/mlin/AppData/Roaming/SPB_Data')
12
```

```
8 x = [1, 2, 3, 4, 5, 6, 7]
9 total = 0
10 print(sum(x[0::2]))
11
```

```
In [16]: runfile('C:/Users/mlin/AppData/Roaming/SPB_Data/untitled16.py',
               wdir='C:/Users/mlin/AppData/Roaming/SPB_Data')
16
```

常見Exception

```
IronPython Command Window
=====
>>> 'abc'+123
TypeError: unsupported operand type(s) for +: 'str' and 'int'
>>> 123/0
ZeroDivisionException: Attempted to divide by zero.
>>> x = [1, 2, 3]
>>> x[4]
IndexOutOfRangeException: index out of range: 4
>>> x.append(4)
MissingMemberException: 'list' object has no attribute 'append'
>>> z = x + y
UnboundNameException: name 'y' is not defined
>>> y = {'A':1, 'B':2}
>>> y['a']
KeyNotFoundException: a
>>> u, v = (1, 2, 3)
ValueError: too many values to unpack
```


腦力激盪

- 分享想要透過自動化解決的題目及緣由，並設想輸入及輸出。
- 每位3分鐘時間，利用板書搭配口頭說明。
- 講師針對題目給出建議及方向，並評估難易度。
- 分享供所有學員參考，不一定為最後專題題目。
- 待第四周所有學員完成分享之後，再從當中選擇最後的專題題目。

