


# Week 3-檔案處理與格式化輸出

# Python基本語法簡介(3/4)

 [Python - Lists](#)

 [Python - Tuples](#)


 [Python - Dictionary](#)

 [Python - Date & Time](#)

 [Python - Functions](#)

 [Python - Modules](#)

 [Python - Files I/O](#)

 [Python - Exceptions](#)

- Accessing Values in Lists
- Updating Lists
- Delete List Elements
- Basic List Operations
- Indexing, Slicing, and Matrixes
- Built-in List Functions & Methods
- List Comprehension
- Accessing Values in Tuples
- Accessing Values in Dictionary
- Updating Dictionary
- Built-in Dictionary Functions
- Opening and Closing Files
- Reading and Writing Files
- Directories in Python

## / list切片

```
In [7]: x = ['A', 'B', 'C', 'D', 'E', 'F']
```

```
In [8]: x[0]
```

```
Out[8]: 'A'
```

```
In [9]: x[3]
```

```
Out[9]: 'D'
```

```
In [10]: x[0:3]
```

```
Out[10]: ['A', 'B', 'C']
```

```
In [11]: x[1:3]
```

```
Out[11]: ['B', 'C']
```

```
In [12]: x[::2]
```

```
Out[12]: ['A', 'C', 'E']
```

```
In [13]: x[1::2]
```

```
Out[13]: ['B', 'D', 'F']
```

```
In [14]: x[::-1]
```

```
Out[14]: ['F', 'E', 'D', 'C', 'B', 'A']
```

```
In [15]: x[-2::-2]
```

```
Out[15]: ['E', 'C', 'A']
```

```
In [16]: x[-3:]
```

```
Out[16]: ['D', 'E', 'F']
```

```
In [17]: y = x[:]
```

```
In [18]: y
```

```
Out[18]: ['A', 'B', 'C', 'D', 'E', 'F']
```

# list運算

宣告空list

**x = [ ]**

宣告list

**x = [4,3,1,5,6,7,2]**

加入

**x.append(8)**

排序

**x.sort()**

返回list長度

**len(x)**

返回最大值

**max(x)**

返回最小值

**min(x)**

list相加

**x + y**

元素運算

**y = [i\*i for i in x]**

元素運算+判斷

**y = [i for i in x if i%2 == 0]**

# / tuple運算

在Python中，Tuple就像是串列（List），不過串列是可變動（Mutable）物件，而Tuple是不可變動（Immutable）物件。你可以使用()來建立Tuple物件，也可以直接逗號區隔元素來建立Tuple物件。

tuple主要用來記錄不同屬性的資料，比方說  
(name, gender, age)，(id, size, color), (x, y, z)

```
In [1]: x = (1,2,3)

In [2]: x
Out[2]: (1, 2, 3)

In [3]: x = 4,5

In [4]: x
Out[4]: (4, 5)

In [5]: x[0]
Out[5]: 4

In [6]: a, b = x

In [7]: a
Out[7]: 4

In [8]: b
Out[8]: 5
```

# / dictionary運算

- 在 Python 的dictionary當中，每一個元素都由鍵 (key) 和值 (value) 構成，結構為key: value。不同的元素之間會以逗號分隔，並且以大括號 {}圍住。字典提供了非常快的查詢速度。

```
In [9]: x = {}

In [10]: x
Out[10]: {}

In [11]: x = {'John': ('Male', 23)}

In [12]: x
Out[12]: {'John': ('Male', 23)}

In [13]: x['Mary'] = ('Female', 18)

In [14]: x
Out[14]: {'John': ('Male', 23), 'Mary': ('Female', 18)}

In [15]: x.keys()
Out[15]: dict_keys(['John', 'Mary'])

In [16]: gender, age = x['Mary']

In [17]: gender
Out[17]: 'Female'

In [18]: age
Out[18]: 18
```



## / 以頻率對應複數的CSV為例，列舉了幾種不同的資料結構

```
S11_freq=[1,2,3,4,5]
```

```
S11_real=[6,7,8,9,10]
```

```
S11_imag=[11,12,13,14,15]
```

```
S11_a=([1,2,3,4,5], [6,7,8,9,10], [11,12,13,14,15])
```

```
S11_b=[(1,6,11),(2,7,12),(3,8,13),(4,9,14),(5,10,15)]
```

```
S11_c=[(1, 6+11j), (2, 7+12j), (3, 8+13j), (4, 9+14j), (5, 10+15j)]
```

```
S11_d={1:6+11j, 2:7+12j, 3:8+13j, 4:9+14j, 5:10+15j}
```

# 資料結構的選擇

資料結構的選擇沒有絕對的好壞，完全取決於要執行的操作。適合A資料結構的操作對於B資料結構可能相當困難，反之亦然。必要的時候我們可以做資料結構轉換，以適應不同的操作程序。

下面是Python常用於儲存大量資料的資料結構.

- List
- List of tuple
- Tuple of list
- Dictionary


## / w = zip(x , y)

zip可以將多個數值list打包成一個list of tuple，舉例來說，我們將freq, gain和溫度放到list of tuple當中

```
8 freq = [1e9, 2e9, 3e9, 4e9]
9 gain = [4, 5, 6, 7]
10 temp=[30, 25, 20, 18]
11 data = zip(freq, gain, temp)
```

將數值透過zip關連起來之後，可以容易在for loop當中做篩選處理，比方說，找出滿足gain大於5，溫度小於27所有的頻率點及溫度，可以寫成

```
13 result=[]
14 for freq, gain, temp in data:
15     if gain>5 and temp<27:
16         result.append((freq, temp))
```

 and `x, y = zip(*w)`

- 可以透過`zip(*)`的方法將list of tuple拆成多個list

```
18 freq_list, temp_list = zip(*result)
19 print(freq_list)
20 print(temp_list)
```

# 數字格式化輸出

為了容易閱讀或是要將輸出字串對齊或置中，就可以利用字串的format方法

數字	格式	輸出	描述
3.1415926	{:.2f}	3.14	保留小数点后两位
3.1415926	{:+.2f}	+3.14	带符号保留小数点后两位
-1	{:+.2f}	-1.00	带符号保留小数点后两位
2.71828	{:.0f}	3	不带小数
5	{:0>2d}	05	数字补零 (填充左边, 宽度为2)
5	{:x<4d}	5xxx	数字补x (填充右边, 宽度为4)
10	{:x<4d}	10xx	数字补x (填充右边, 宽度为4)
1000000	{:,}	1,000,000	以逗号分隔的数字格式
0.25	{:.2%}	25.00%	百分比格式
1000000000	{:.2e}	1.00e+09	指数记法
13	{:>10d}	13	右对齐 (默认, 宽度为10)
13	{:<10d}	13	左对齐 (宽度为10)
13	{:^10d}	13	中间对齐 (宽度为10)

```
>>>
>>> import math
>>> '{}'.format(math.pi)
'3.1416'
>>> '{:.6f}'.format(math.pi)
'3.141593'
>>> '{:>12.6f}'.format(math.pi)
'      3.141593'
>>> '{:<12.6f}'.format(math.pi)
'3.141593      '
>>> '{:^12.6f}'.format(math.pi)
'  3.141593  '
>>> '{:12.3e}'.format(math.pi)
'      3.142e+00'
>>> '{:+12.3e}'.format(math.pi)
'    +3.142e+00'
>>> '{:+12.3e}'.format(-math.pi)
'    -3.142e+00'
>>> '{:+12.3%}'.format(-math.pi)
'    -314.159%'
>>>
```

# / IronPython啟動存檔瀏覽對話框

```
1 import sys
2 import clr
3 clr.AddReference("System.Windows.Forms")
4
5 from System.Windows.Forms import DialogResult, SaveFileDialog
6 dialog = SaveFileDialog()
7 dialog.Filter = "text files (*.txt)|*.txt"
8
9 if dialog.ShowDialog() == DialogResult.OK:
10     txt_path = dialog.FileName
11     AddWarningMessage(txt_path)
12 else:
13     sys.exit()
14
```

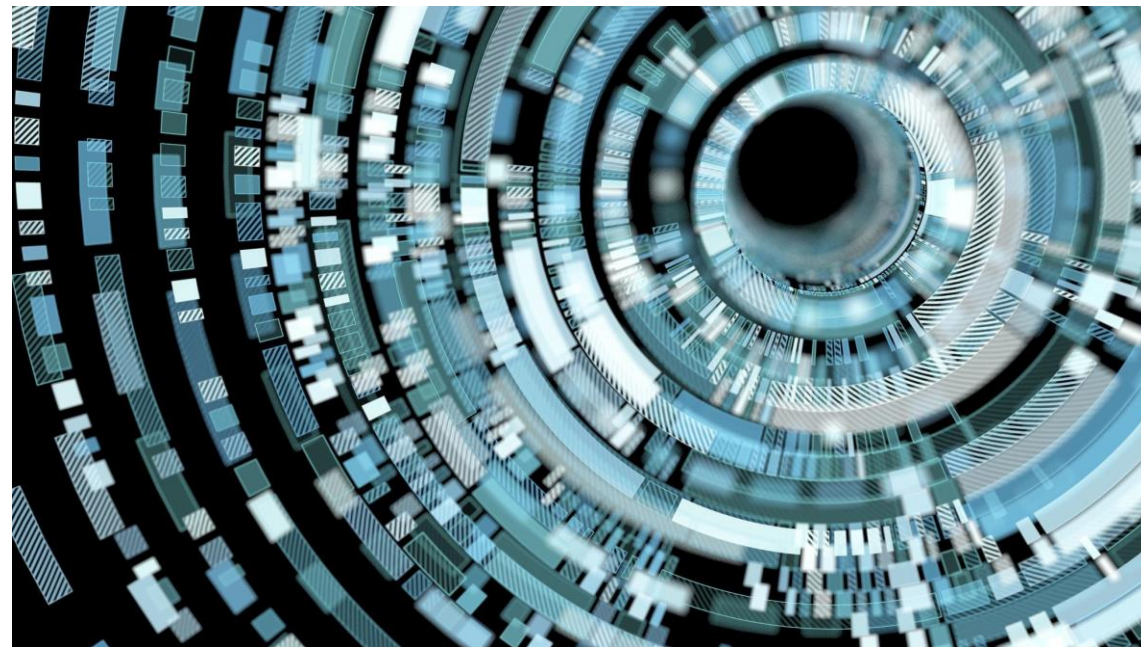
## / 讀/寫文字檔

將每一行讀到list當中

```
with open(file_name) as f:  
    text = f.readlines()
```

將list當中的字串寫到檔案當中

```
with open(file_name, 'w') as f:  
    for i in string_list:  
        f.writelines(i + '\n')
```



# 概念解説



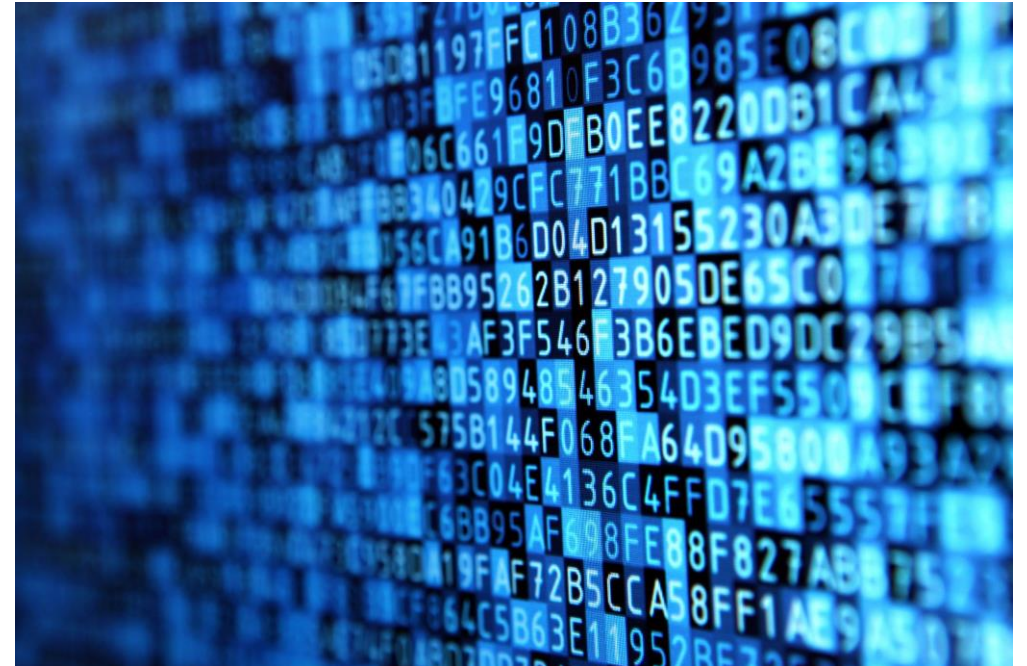
## 熟悉檔案的讀寫為何重要？

AEDT存在許多的設定檔，比方說是材料設定檔，激發設定檔，堆疊設定檔等等。以文字檔格式紀錄。對前處理來說，讀取設定檔便可以取得必要的訊息，也可以透過修改檔案來更新設計參數。對後處理來說，模擬結果都可以匯出到檔案當中。因此熟悉檔案的讀寫，對模擬自動化是必要的技能。

檔案採用的資料種類及方法，根據資料的格式及之後要進行的運算，我們必須選擇適當的資料結構來儲存，才能有效的完成後續的工作。解析檔案並儲存至資料結構的程式稱之為**parser**。最常見的**csv**檔為例，**csv parser**便是將**csv**檔案讀到**list of list**的資料結構，使用者可以輕鬆的使用切片來取得某一行，某一列的資料。

## 如何從AEDT當中匯出模擬資料

當完成模擬之後，多數的資料是以二進制的格式儲存在專案檔中，無法直接取用。我們可以先行產生表格報告之後為，再以csv檔案格式匯出。在另行以python做處理。有些常用的模擬結果可以支援特定檔案格式的直接輸出，像是snp，ffd，nfd，spice等。這些匯出動作都有AEDT函式可以支援。



# / CSV檔

CSV是最為普遍的資料儲存格式，每一列當中不同屬性的資料以分隔號區分開來。這種格式不但可讀性高，程式碼也容易處理。除了資料本身，最前面的行數也會用來記錄相關訊息，比方說是日期或是單位等等。為了與資料區分，檔頭的這些訊息前面多以特殊字元表示，以利區隔。各位所熟悉的**S**參數就是屬於這種格式。

要讀取**csv**格式，首先要先分離檔頭及資料，檔頭的資訊可以透過字串處理或是正規表示法來擷取資訊，並存到變數當中。資料的部分就簡單的多，讀取每一列，並根據分隔號分割資料並存到**list**當中。

csv所儲存的資料格式通常較為單純。複雜度較高的資料一般透過xml檔或json檔紀錄。AEDT的堆疊設定便是xml格式。json檔對於python而言，較易處理。也是這幾年較受歡迎的格式。xml格式基於歷史因素仍大量存在於AEDT當中。要完整剖析xml檔可以使用內建的parser。如果只需要擷取部分資料可以使用正規表示法。正規表示法留到之後再作介紹。

```
1 <?xml version="1.0" encoding="UTF-8" standalone="no" ?>
2 <c:Control xmlns:c="http://www.ansys.com/control" schemaVersion="1.0">
3
4   <Stackup schemaVersion="1.0">
5     <Materials>
6       <Material Name="AIR">
7         <Permittivity>
8           <Double>1</Double>
9         </Permittivity>
10        <Permeability>
11          <Double>1</Double>
12        </Permeability>
13        <Conductivity>
14          <Double>0</Double>
15        </Conductivity>
```

## / pickle讀寫

基本輸入輸出除了利用基本 I/O，自行決定要保存的運算結果之外，如果想直接保存物件狀態，在下次重新執行程式時讀取以恢復運算時必要的資料，這類的技術稱為物件序列化（Object serialization），在 Python 中，提供標準模組 pickle來進行這方面的支援。



# 範例解說：如何讀取 CSV 檔計算資料並格式化輸出

匯出資料

- CSV
- XML
- TXT

開檔讀取

- 計算
- 篩選

輸出

- 格式化
- 儲存

# 專題討論

## / 個人專題分享

- 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5分鐘時間並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。

題目:

緣由:

輸入:

輸出: