

# Week 3-檔案處理與格式化輸出



# 範例解說：如何讀取檔案 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

# / 定義需求

- 需求
  - 找出所有S參數的頻率範圍
- 輸入
  - 指定目錄
- 輸出
  - 紀錄檔名及其頻率範圍的.csv檔



# 程式開發流程

- 研究S參數檔案格式
- 開啟S參數檔案並讀取資料
- 讀取單位
- 讀取數值資料段
- 從數值資料段擷取頻率點並根據單位轉換正確數值
- 包裝成函數
- 讀取特定目錄底下的所有檔案
- 輸出每個檔案的頻段到CSV當中

# 程式改善

- 加上例外處理
- 加上開啟目錄瀏覽對話窗
- 處理編碼錯誤問題

# 取得特定路徑底下所有S參數的頻率範圍



outputfreqrange.py.txt

Spyder (Python 3.7)

File Edit Search Source Run Debug Consoles Projects Tools View Help

Editor - D:\demo\outputfreqrange.py

```
1 import os
2 import codecs
3
4 def getFrequency(file):
5     N = int(file.split('.')[1][1:-1])
6     with codecs.open(file, 'r', encoding='utf-8', errors='ignore') as f:
7         text = f.readlines()
8
9     scale_map={'hz':1e0, 'khz':1e3, 'mhz':1e6, 'ghz':1e9, 'thz':1e12}
10    value = []
11    for i in text:
12        if i[0] == '!':
13            continue
14        elif i[0] == '#':
15            unit = i.split()[1].lower()
16        else:
17            value += list(map(float, i.split()))
18    return [f*scale_map[unit] for f in value[:2*(N*N)+1]]
19
20 folder = "D:/OneDrive - ANSYS, Inc/Models/S_Parameter"
21 output = 'd:/demo/info.csv'
22
23 with open(output, 'w') as f:
24     for file in os.listdir(folder):
25         try:
26             freq = getFrequency(os.path.join(folder, file))
27             f.writelines(file+'\n')
28             f.writelines('{}GHz, {}GHz\n'.format(min(freq)/1e9, max(freq)/1e9))
29         except:
30             print(file)
31
```

Variable explorer

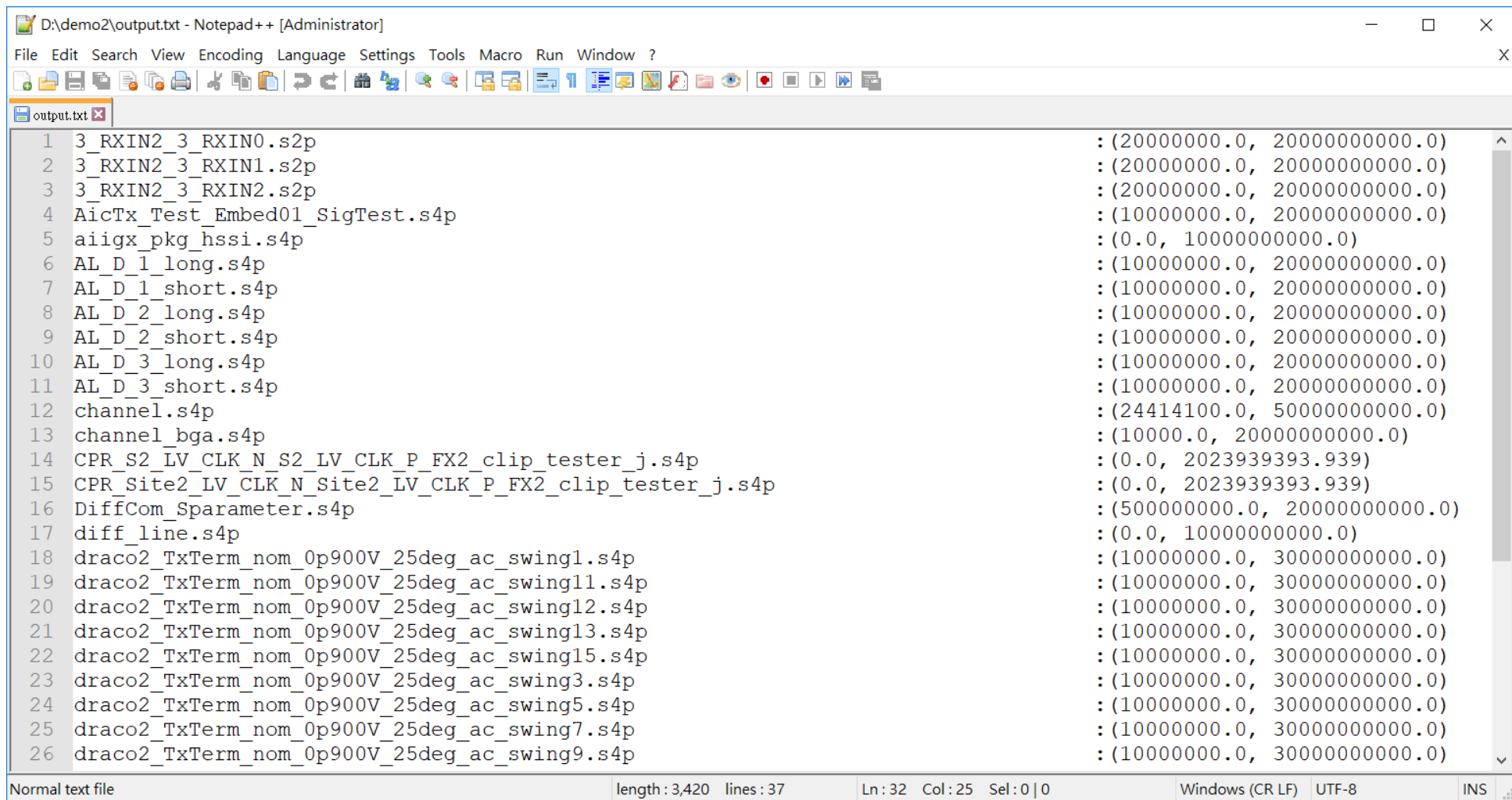
Name	Type	Size	Value
N	int	1	4
codecs	module	1	module object of builtins
f	TextIOWrapper	1	TextIOWrapper object of _io
file	str	1	Report
folder	str	1	D:/OneDrive - ANSYS, Inc/Models/S_Parameter
freq	list	2090	[10.0, 11.6591, 13.5936, 15...

IPython console

Console 1/A

Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 33 Column: 9 Memory: 35 %

# 輸出S參數檔檔名及其頻率範圍



D:\demo2\output.txt - Notepad++ [Administrator]

File Edit Search View Encoding Language Settings Tools Macro Run Window ?

output.txt

1	3_RXIN2_3_RXIN0.s2p	:(20000000.0, 20000000000.0)
2	3_RXIN2_3_RXIN1.s2p	:(20000000.0, 20000000000.0)
3	3_RXIN2_3_RXIN2.s2p	:(20000000.0, 20000000000.0)
4	AicTx_Test_Embed01_SigTest.s4p	:(10000000.0, 20000000000.0)
5	aiigx_pkg_hssi.s4p	:(0.0, 10000000000.0)
6	AL_D_1_long.s4p	:(10000000.0, 20000000000.0)
7	AL_D_1_short.s4p	:(10000000.0, 20000000000.0)
8	AL_D_2_long.s4p	:(10000000.0, 20000000000.0)
9	AL_D_2_short.s4p	:(10000000.0, 20000000000.0)
10	AL_D_3_long.s4p	:(10000000.0, 20000000000.0)
11	AL_D_3_short.s4p	:(10000000.0, 20000000000.0)
12	channel.s4p	:(24414100.0, 50000000000.0)
13	channel_bga.s4p	:(10000.0, 20000000000.0)
14	CPR_S2_LV_CLK_N_S2_LV_CLK_P_FX2_clip_tester_j.s4p	:(0.0, 2023939393.939)
15	CPR_Site2_LV_CLK_N_Site2_LV_CLK_P_FX2_clip_tester_j.s4p	:(0.0, 2023939393.939)
16	DiffCom_Sparameter.s4p	:(5000000000.0, 20000000000.0)
17	diff_line.s4p	:(0.0, 10000000000.0)
18	draco2_TxTerm_nom_0p900V_25deg_ac_swing1.s4p	:(10000000.0, 30000000000.0)
19	draco2_TxTerm_nom_0p900V_25deg_ac_swing11.s4p	:(10000000.0, 30000000000.0)
20	draco2_TxTerm_nom_0p900V_25deg_ac_swing12.s4p	:(10000000.0, 30000000000.0)
21	draco2_TxTerm_nom_0p900V_25deg_ac_swing13.s4p	:(10000000.0, 30000000000.0)
22	draco2_TxTerm_nom_0p900V_25deg_ac_swing15.s4p	:(10000000.0, 30000000000.0)
23	draco2_TxTerm_nom_0p900V_25deg_ac_swing3.s4p	:(10000000.0, 30000000000.0)
24	draco2_TxTerm_nom_0p900V_25deg_ac_swing5.s4p	:(10000000.0, 30000000000.0)
25	draco2_TxTerm_nom_0p900V_25deg_ac_swing7.s4p	:(10000000.0, 30000000000.0)
26	draco2_TxTerm_nom_0p900V_25deg_ac_swing9.s4p	:(10000000.0, 30000000000.0)

Normal text file length: 3,420 lines: 37 Ln: 32 Col: 25 Sel: 0 | 0 Windows (CR LF) UTF-8 INS

# 觀念解說



# 資料分析的重要性

- 模擬的結果必須搭配有效的分析，模擬才有意義
- 分析要回答的問題包括：
  - 特性是否符合規格要求
  - 特性是否有改善或惡化
  - 量測與模擬結果是否吻合
- 隨著模擬複雜度的增加，分析的難度也大幅提升，包括：
  - 同一個設計port數的增加
  - 不同模擬結果的比較
  - 設計的製程變異評估
- 傳統手動的資料處理已經不足以有效處理如此龐大的數據量。透過程式來分析已經勢在必行。

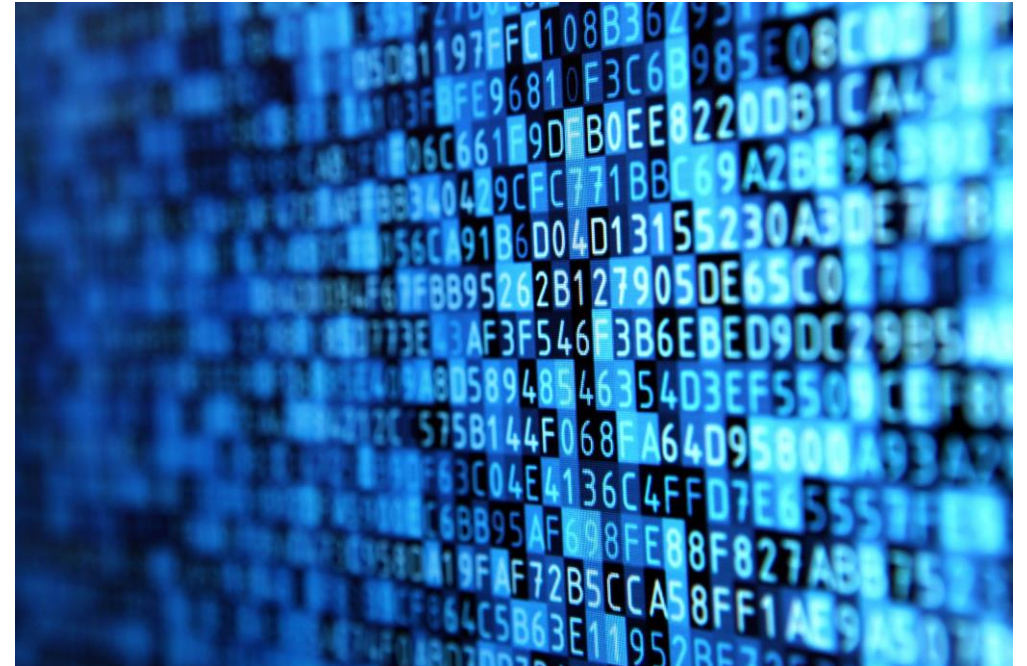
## 熟悉檔案的讀寫為何重要？

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。

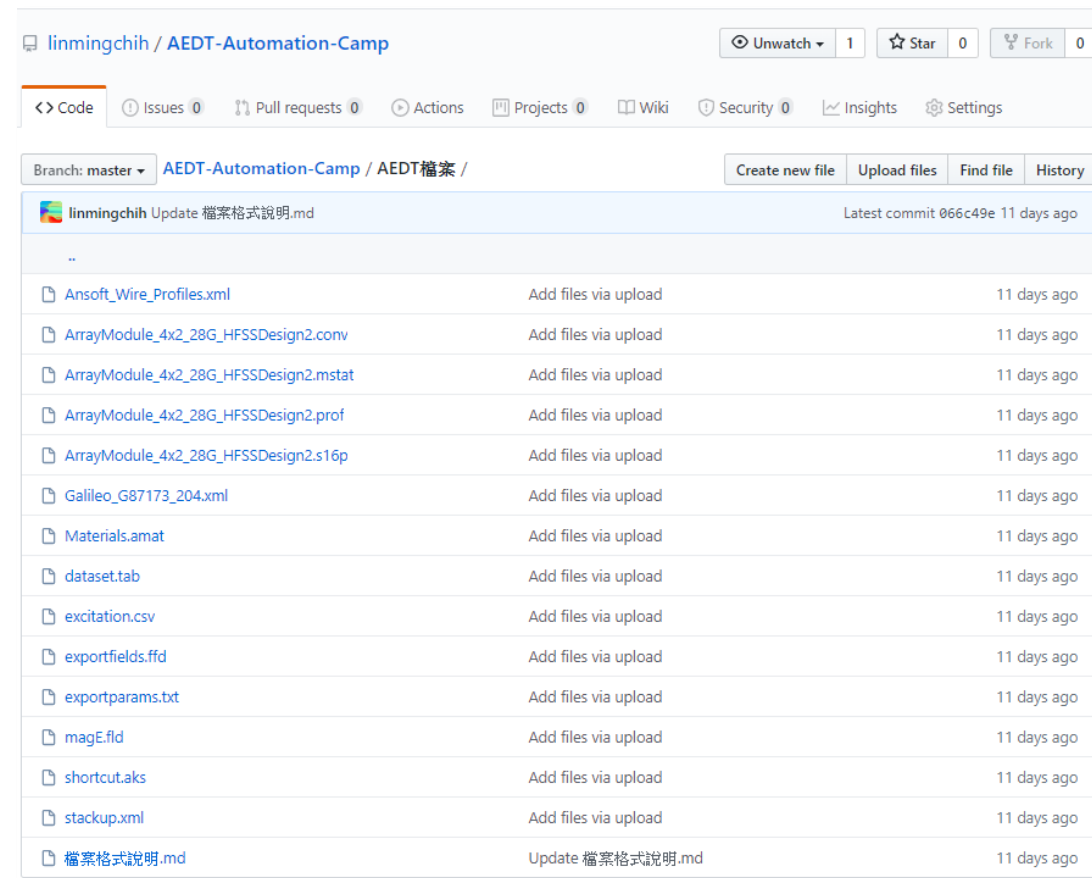
## 如何從AEDT當中匯出模擬資料

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。



# / AEDT可匯出之資料檔

- 檔案範例連結
- 格式
  - Touchstone1.0 .sNp
  - Touchstone 2.0 .ts
  - SPICE .cir, .sp
  - Result .csv
  - Far Field .ffd
  - Near Field .nfd
  - Excitation .csv
  - Profile .prof
  - Mesh .mstat
  - Dataset .tab
  - Stackup .xml
- Material .amat
- Option .xml
- Welement .sp
- Variables .autovar
- ...



# / 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>
```

# 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](#)



## / 找出特定目錄底下的特定檔案

- 找出單一目錄底下所有.txt 檔。

```
import os
for file in os.listdir("/mydir"):
    if file.endswith(".txt"):
        print(os.path.join("/mydir", file))
```

- 找出單一目錄底下所有.txt 檔，包含子目錄。

```
import os
for root, dirs, files in os.walk("/mydir"):
    for file in files:
        if file.endswith(".txt"):
            print(os.path.join(root, file))
```

## / 切換工作目錄

- 程式執行時有時需要配置檔或匯入自定義模組，而配置檔往往都是跟程式擺放在同一目錄底下。如不指定目錄，Python會嘗試到工作目錄尋找配置檔，
- 如果Python找不到配置檔，則會引發FileNotFoundError錯誤。為了解決這個問題，建議程式開頭便將工作目錄切換與程式本身所在的目錄。

```
import os

abspath = os.path.abspath(__file__)
dname = os.path.dirname(abspath)
os.chdir(dname)
```

# / import指令

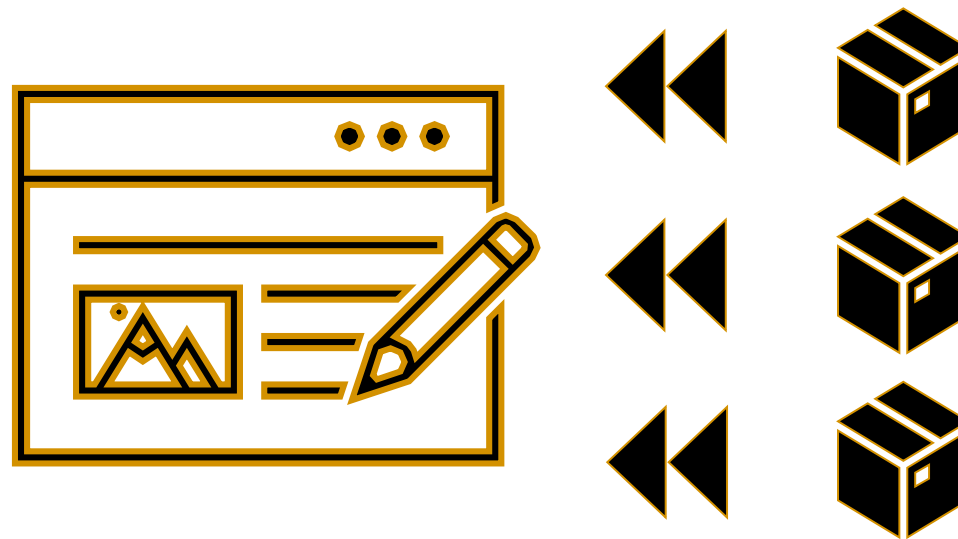
- import指令用來匯入模組
- 自動化常用模組
  - math, os, sys, re, tkinter, matplotlib
- 匯入指令

```
import math
```

```
from math import sin, cos
```

```
from math import sin as mysin
```

```
from math import *
```



# / math module & cmath module

- 基本數學函式庫：<https://docs.python.org/3/library/math.html>
- 複數函式庫：<https://docs.python.org/3/library/cmath.html>

```
IronPython Command Window

=====
ElectronicsDesktop 2020.1.0
IronPython 2.7.0.40 on .NET 4.0.30319.42000
=====
- With Tab completion
- dir()      - lists all available methods and objects
- dir(obj)   - lists all available attributes/methods on obj
- help(obj)  - provides available help on a method or object
- tutorial() - provides more help on using the console
=====
try executing "dir(oDesktop)" or dir_sig(oDesktop,"ver")
=====
>>> import math
>>> math.sin(math.pi/4)
0.70710678118654746
>>> math.cos(math.pi/4)
0.70710678118654757
>>> dir(math)
['_doc_', '__name__', '__package__', 'acos', 'acosh', 'asin', 'asinh', 'atan',
'atan2', 'atanh', 'ceil', 'copysign', 'cos', 'cosh', 'degrees', 'e', 'erf', 'erfc',
'exp', 'expm1', 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum', 'gamma',
'hypot', 'isinf', 'isnan', 'ldexp', 'lgamma', 'log', 'log10', 'loglp', 'modf', 'pi',
'pow', 'radians', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'trunc']
>>> |
```

- 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%'
>>>
```

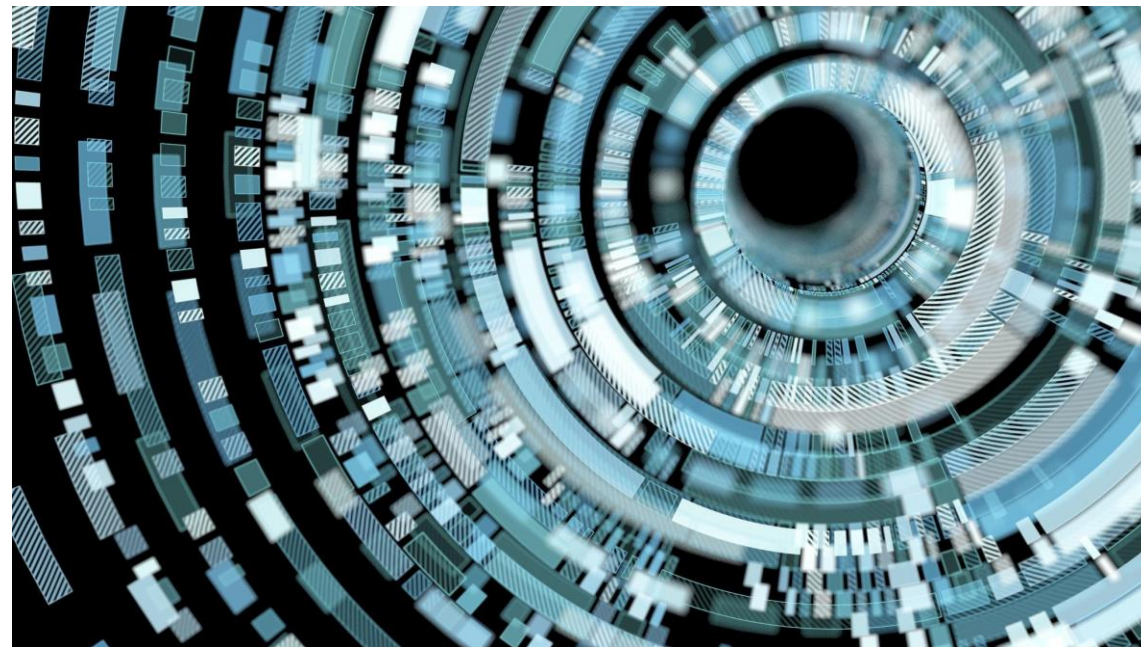
## / 讀/寫文字檔

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')
```



## 例外處理 v.s. 事先檢查

在程式開發階段，一般都是假設輸入參數會嚴格遵守規範，並基於這個假設開發演算法。如果格式出錯，就算這個錯誤無關緊要，程式也會終止運算並返回錯誤訊息，如果要對輸入的資料一一判斷是否合規，將導致程式複雜化並額外耗用運算資源。

一個解決思維是先做再說，發生了正常流程無法處理的狀況，也就是例外，再另行處理。這種設計思維就是所謂的例外處理。目前例外處理已經是主流程式語言所採取的設計方案。這種設計風格又稱之為EAFP( Easier to Ask for Forgiveness than Permission) ,有別於LBYL(Look Before You Leap)。



 **Ansys**

