一步一腳印
精通Python程式設計

檔案讀寫、字串及例外處理

課程講師: 陳建銘

2018/06/27

課程大綱

- 檔案讀寫
  - 檔案類型
  - TXT檔案讀寫
  - Excel檔案讀寫
  - JSON檔案讀寫
  - XML檔案讀寫
- 字串處理
  - 內建處理函式
  - 正規表示式
- 例外處理

檔案讀寫

檔案類型–

資料處理需要大量的檔案讀取與寫入，常見的檔案存取類型有文字檔(.txt)、Excel檔(.xlsx、.xls)、JSON檔(.json)及XML檔(.xml)。

檔案讀寫 cont.

JSON (JavaScript Object Notation) –
將結構化資料儲存為JavaScript物件標準格式，常應用於網路上的資料傳輸及呈現。

JSON內容類似Python裡的複合字典

```
[{"FirstName": "Peter",
  "LastName": "Lin",
  "Report": [{"Subject": "English",
              "Score": 82},
             {"Subject": "Science",
              "Score": 76}],
  "SID": 123456},
 {"FirstName": "Mary",
  "LastName": "Wang",
  "Report": [{"Subject": "English",
              "Score": 73},
             {"Subject": "Science",
              "Score": 94}],
  "SID": 456789},
 ..., {...}]
```

檔案讀寫 cont.

XML (Extensible Markup Language) –

可延伸標籤式語言是一種文件實體，文件由字元資料、標籤及屬性三者組合而成。XML主要的設計目的是用來傳送及攜帶資料，而常見的網頁標籤式語言HTML則是用於表現及展示資料。

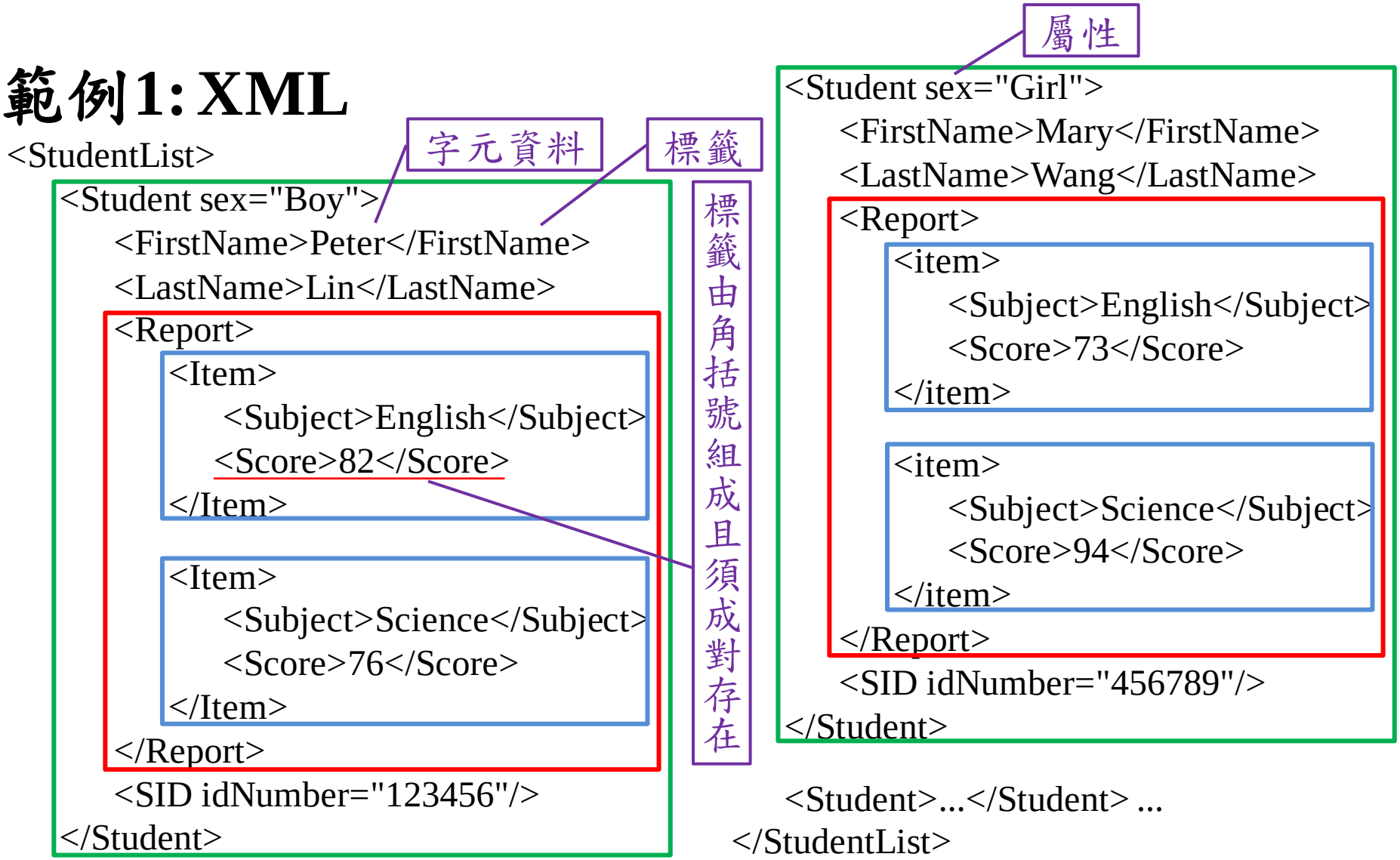
```
<html>
    <head>
        <title>Example</title>
    </head>
    <body>
        <center>
            <img src="show.png" style="border:1px solid #FF0000">
        </center>
    </body>
</html>
```

顯示網頁標題文字於瀏覽器

置中顯示圖檔於網頁內容

檔案讀寫 cont.

範例1: XML



檔案讀寫 cont.

TXT檔案讀寫 –

open('FileName.txt', 'mode', encoding='EncodingType'):

模式	描述
r	讀取模式，此為函式預設值。
w	會覆蓋舊檔內容的寫入模式。
a	附加至舊檔內容的寫入模式。
x	寫入模式。若檔案不存在，則開新檔案。否則發生錯誤。
t	文字模式。
b	二進位模式。
r+	更新模式，可讀可寫。檔案須已存在，從檔案開頭開始讀寫。
w+	更新模式，可讀可寫。開新檔案，覆蓋舊檔內容。
a+	更新模式，可讀可寫。開新檔案，附加內容至舊檔尾端。

檔案讀寫 cont.

範例2: 讀取並顯示TXT檔內容

```
def readShowText(fileName):
    readTxt = open(fileName, 'r')

    for line in readTxt:
        print(line.strip())

    readTxt.close()

if __name__ == '__main__':
    readShowText('TestFile.txt')
```

TestFile.txt

```
1   This is the first sentence.
2   Here is the second sentence.
3   The third sentence is here.
```

透過檔案物件逐列讀取檔案內容。

檔案讀取完畢須關閉檔案以釋放資源。

輸出如下:

```
This is the first sentence.
Here is the second sentence.
The third sentence is here.
```

檔案讀寫 cont.

範例3: 讀寫TXT檔內容

```
def readWriteText(inFile, outFile):
    index = 1

    inT = open(inFile, 'r', encoding='UTF-8')
    outT = open(outFile, 'w', encoding='UTF-8')

    for line in inT:
        outT.write(str(index) +'. '+ line)
        index += 1

    inT.close()
    outT.close()
```

```
if __name__ == '__main__':
    readWriteText('TestFile.txt', 'Result.txt')
```

TestFile.txt

```
1   This is the first sentence.
2   這是檔案內容第二句。
3   The third sentence is here.
```

讀寫含有中文內容的檔案，須加入UTF-8編碼。

Result.txt

```
1   1. This is the first sentence.
2   2. 這是檔案內容第二句。
3   3. The third sentence is here.
```

逐列讀取並逐列寫入。

檔案讀寫 cont.

Excel檔案讀寫 –

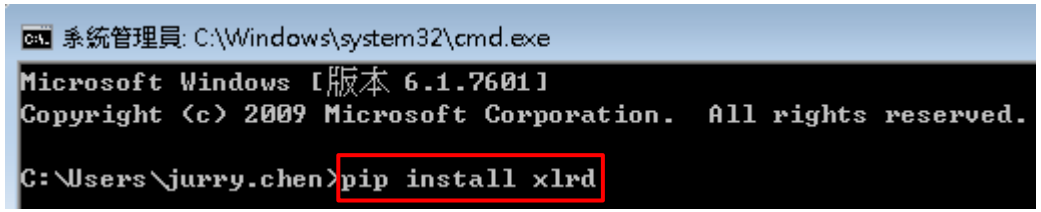
安裝可讀寫Excel的外部函式庫

步驟1: 打開終端機

步驟2:安裝xlrd、openpyxl

輸入 pip install xlrd

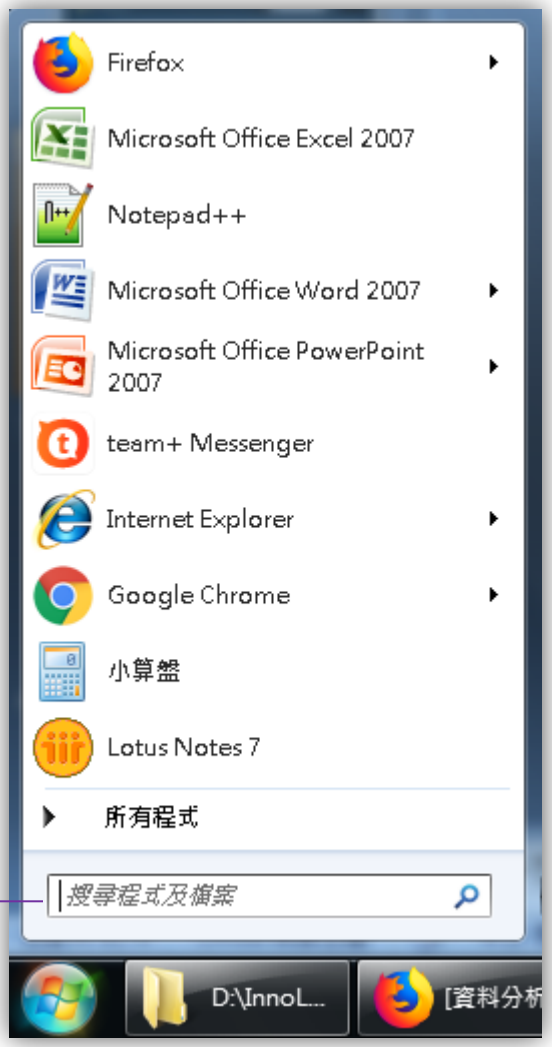
輸入 pip install openpyxl



確認外部函式庫是否安裝成功?!

ANS: 輸入pip freeze，查看是否有xlrd及openpyxl

輸入cmd開啟終端機



檔案讀寫 cont.

範例4: 讀寫Excel檔內容

```
if __name__ == '__main__':
    readWriteXlsx('TestExcel.xlsx', 'resultExcel.xlsx')
```

TestExcel.xlsx

	A	B	C
1	Field1	Field2	Field3
2	1	2	3
3	4	5	6
4	7	8	9

ResultExcel.xlsx

	A	B	C
1	Field1	Field2	Field3
2	1	2	3
3	4	5	6
4	7	8	9
5			
6	61	62	63
7	71	72	73
8	81	82	83
9	91	92	93

```
import xlrd
import openpyxl

def readWriteXlsx(inFile, outFile):
    wb = xlrd.open_workbook(inFile)
    getSheet = wb.sheet_by_name('FirstTable')

    wt = openpyxl.Workbook()
    writeSheet = wt.active
    writeSheet.title = 'SecondTable'

    for rowIdx in range(getSheet.nrows):
        rowInfoList = getSheet.row_values(rowIdx)

        for clnIdx in range(len(rowInfoList)):
            writeSheet.cell(row=rowIdx+1,
                            column=clnIdx+1,
                            value=rowInfoList[clnIdx])
            writeSheet.cell(row=rowIdx+6,
                            column=clnIdx+1,
                            value=int(str(rowIdx+6)+str(clnIdx+1)))

    wt.save(outFile)
```

初始化檔案讀取物件

初始化檔案寫入物件

逐列讀取檔案內容

逐筆讀取每列內容

檔案讀寫 cont.

JSON檔案讀寫 –

安裝可讀寫JSON的外部函式庫，pip install json。

以字串值為JSON內容的字串做為loads方法的引數。此方法可將引數內容解碼轉換成複合字典。

json.loads('{"key$_1$":"value$_1$", "key$_2$":"value$_2$", ...}')

load方法接受JSON檔案物件做為引數。可將引數內容解碼轉換成複合字典。

json.load(jsonFileObject)

json.dumps({"key$_1$":"value$_1$", "key$_2$":"value$_2$", ...})

以複合字典做為dumps方法的引數。此方法可將引數內容編碼轉換成字串值為JSON內容的字串。

檔案讀寫 cont.

範例5: 讀取並顯示JSON檔內容

```
import json

def jsonAccess(inputFile):
    with open(inputFile, 'r') as fileIn:
        jsonRawData = json.load(fileIn)
        #jsonRawData = json.loads(fileIn.read())

    for jsonObj in jsonRawData:
        print(jsonObj['FirstName'] +' '+ \
                jsonObj['LastName'] +', SID = '+ \
                str(jsonObj['SID']))
```

load方法接受JSON檔案物件做為引數。

loads方法接受字串值為JSON內容的字串做為引數。

透過內建字典的語法指令來存取JSON檔內容。

```
if __name__ == '__main__':
    jsonAccess('TestJSON.json')
```

輸出如下:

```
Peter Lin, SID = 123456
Mary Wang, SID: 456789
```

檔案讀寫 cont.

範例6: 讀寫JSON檔內容

```
import json

def jsonAccess(inputFile, outputFile):
    jsonDict = {}

    with open(inputFile, 'r') as fileIn:
        jsonRawData = json.load(fileIn)

    jsonDict['FirstName'] = 'Jacky'
    jsonDict['LastName'] = 'Huang'
    jsonDict['SID'] = '135246'
    jsonDict['Report'] = [{'Subject':'English', 'Score':77}]

    jsonRawData.append(jsonDict)

    with open(outputFile, 'w') as fileOut:
        json.dump(jsonRawData, fileOut)
```

```
if __name__ == '__main__':
    jsonAccess('TestJSON.json', 'ResultJSON.json')
```

TestJSON.json

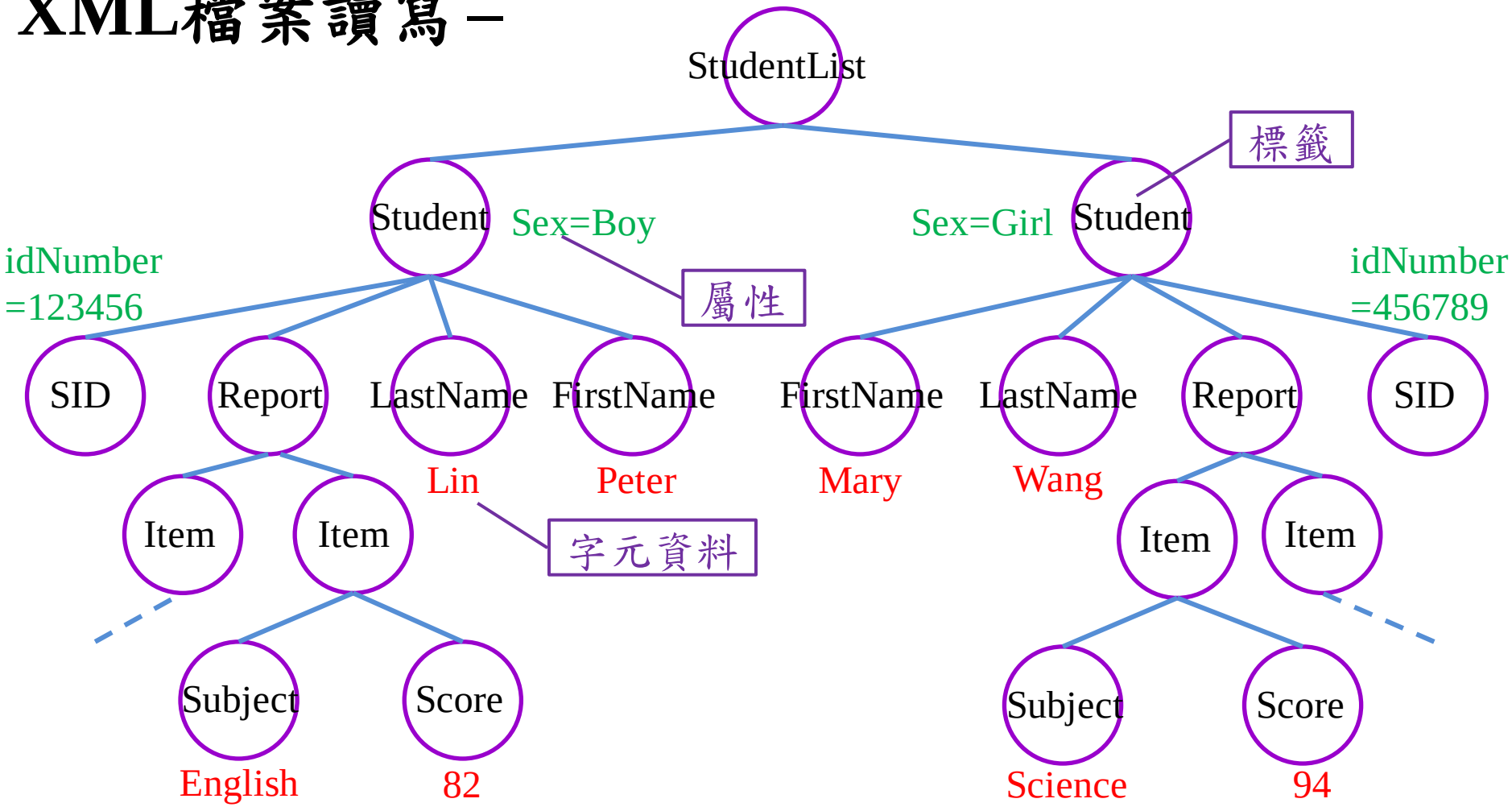
```
[{"FirstName": "Peter",
  "LastName": "Lin",
  "Report": [{"Subject": "English", "Score": 82},
             {"Subject": "Science", "Score": 76}],
  "SID": 123456},
 {"FirstName": "Mary",
  "LastName": "Wang",
  "Report": [{"Subject": "English", "Score": 73},
             {"Subject": "Science", "Score": 94}],
  "SID": 456789}
]
```

ResultJSON.json

```
[{"FirstName": "Peter",
  "LastName": "Lin",
  "Report": [{"Subject": "English", "Score": 82},
             {"Subject": "Science", "Score": 76}],
  "SID": 123456},
 {"FirstName": "Mary",
  "LastName": "Wang",
  "Report": [{"Subject": "English", "Score": 73},
             {"Subject": "Science", "Score": 94}],
  "SID": 456789},
 {"FirstName": "Jacky",
  "LastName": "Huang",
  "SID": "135246",
  "Report": [{"Subject": "English", "Score": 77}]}
]
```

檔案讀寫 cont.

XML檔案讀寫 –



檔案讀寫 cont.

範例6: 讀取XML檔內容

```
import xml.etree.ElementTree as ET

def xmlAccess(inFile):
    tree = ET.parse(inFile)

    for path in tree.iterfind('Student/SID'):
        print(path.tag +' = '+ path.get('idNumber'))

    for node in tree.findall('Student'):
        print(node.find('FirstName').text +' '+\
              node.find('LastName').text +', sex: '+\
              node.get('sex'))
```

```
if __name__ == '__main__':
    xmlAccess('TestXML.xml')
```

iterfind方法接受標籤路徑作為引數，回傳特定路徑下的所有節點。

findall方法接受單一標籤作為引數，可找出目前標簽下的所有特定名稱的子標籤。

輸出如下:

```
SID = 123456
SID = 456789
Peter Lin, sex: Boy
Mary Wang, sex: Girl
```

字串處理

內建處理函式I –

方法	描述
str.capitalize()	將字串第一個字元改為大寫，其餘字元皆改為小寫並回傳
str.center(width, fillchar)	回傳原字串置中，前後補width-len(str)個空白字元的新字串
str.count(sub, start, end)	回傳sub在起始索引start至結束索引end間的出現次數
str.encode(encoding="UTF-8")	回傳encoding版本的bytes物件
str.endswith(suffix, start, end)	判斷字串在起始索引start至結束索引end間是否以suffix結尾
str.find(sub, start, end)	回傳sub在起始索引start至結束索引end間第一次出現的索引
str.index(sub, start, end)	回傳sub在起始索引start至結束索引end間第一次出現的索引
str.isalnum()	判斷字串中的字元是否至少一個字母或數字
str.isalpha()	判斷字串中的字元是否至少一個字母
str.isdecimal()	判斷字串中的所有字元是否是十進位數字
str.isdigit()	判斷字串中的所有字元是否是數字

字串處理 Cont.

內建處理函式 II –

方法	描述
str.isidentifier()	判斷字串是否可作為合法的識別字
str.islower()	判斷字串中的所有字元是否是小寫字母
str.isnumeric()	判斷字串中的所有字元是否是數字
str.isprintable()	判斷字串中的所有字元是否都屬於可見字元
str.isspace()	判斷字串是否為空格字元
str.istitle()	判斷字串是否適合當作標題
str.isupper()	判斷字串中的所有字元是否是大寫字母
str.join(iterable)	連結iterable各元素的字串並回傳
str.ljust(width, fillchar)	將字串以寬度width向左對齊且fillchar向結尾填充並回傳
str.lower()	將字串中的英文字母全改成小寫
str.lstrip(char)	從字串的開頭移除所有char字元，預設為空白字元

字串處理 Cont.

內建處理函式 III –

方法	描述
str.partition(sep)	以sep分割字串為三個子字串並回傳包含這三個子字串的序對
str.replace(old, new)	將字串中的old子字串置換為new子字串
str.rfind(sub, start, end)	回傳sub在起始索引start至結束索引end間最後出現的索引
str.rindex(sub, start, end)	回傳sub在起始索引start至結束索引end間最後出現的索引
str.rjust(width, fillchar)	將字串以寬度width向右對齊且fillchar向開頭填充並回傳
str.rpartition(sep)	以sep從結尾分割字串為三個子字串並回傳包含這三個子字串的序對
str.rsplit(sep)	將字串從結尾以sep分割成子字串並回傳儲存子字串的串列
str.rstrip(char)	從字串的結尾移除所有char字元，預設為空白字元
str.split(sep)	將字串以sep分割成子字串並回傳包含這些子字串的串列
str.startswith(prefix, start, end)	判斷字串在起始索引start至結束索引end間是否以prefix開頭
str.strip(char)	從字串的開頭及結尾移除char字元，預設為空白字元

字串處理 Cont.

內建處理函式 IV –

方法	描述
str.swapcase()	將字串中的英文字母進行大小寫轉換
str.title()	將字串轉換成作為標題的字串
str.translate(map)	將字串中的字元以map中配對的字元做轉換
str.upper()	將字串中的英文字母全改成大寫
str.zfill(width)	回傳以0塞滿寬度為width的新字串

字串處理 Cont.

範例1: 基本字串處理 I

```
def stringOperation():
    myString = 'The first python string example.'

    print('The 11\'th character is ' + myString[10])
    print('The character with index "-8" is ' + myString[-8])
    print('The final token is ' + myString[-8:])
    print('The third token is ' + myString[10:16])
    print(myString[17:23]==myString[-15:-9])

    myTuple = myString[10:16], myString[17:23]
    print(myTuple)
```

```
if __name__ == '__main__':
    stringOperation()
```

輸出如下:

```
The 11'th character is p
The character with index "-8" is e
The final token is example.
The third token is python
True
('python', 'string')
```

負值索引表示由後往前計數，最後一個索引值為-1。

冒號前後分別代入起始索引值及結束索引值。

字串處理 Cont.

範例2: 基本字串處理II

```
def stringProcessing():
    myString = 'this is My first String.'
    myString = myString.capitalize()

    print(myString.center(30))
    print('The number of i is', myString.count('i'))
    print(myString.strip())
    print(myString.startswith('This'))
    print(myString.endswith('string'))
    print('The second i is in index', myString.find('i',3))
    print('The second t is in index', myString.index('t',1))
```

```
if __name__ == '__main__':
    stringProcessing()
```

原字串置中，前後補空白字元。

輸出如下:

```
 This is my first string.
The number of i is 4
This is my first sting.
True
False
The second i is in index 5
The second t is in index 15
```

字串處理 Cont.

範例3: 字串切割

```
def stringSegmentation():
    nameString = 'Peter,Tom,Mary,Bob, Jack'

    tokenTuple = nameString.partition(',')
    tokenList = nameString.split(',')

    print(tokenTuple)
    print(tokenList)
```

```
if __name__ == '__main__':
    stringSegmentation()
```

輸出如下:

```
('Peter', ',', 'Tom,Mary,Bob,Jack')
['Peter', 'Tom', 'Mary', 'Bob', 'Jack']
```

partition方法將字串切割成三序對(3-tuple)，分別是切割點左字串、切割點及切割點右字串。

split方法將字串依切割點切分成多個子字串，並以串列裝載及回傳。

字串處理 Cont.

範例4: 串列合併成字串

```
def listCombination():
    stringList= ['This', 'is', 'my', 'first', 'string.']
    string1 = ' '.join(stringList)

    numberList = [1, 4, 7, 2, 5, 8, 3, 6, 9]
    string2 = '|'.join(str(intValue) for intValue in numberList)

    print(string1)
    print(string2)
```

```
if __name__ == '__main__':
    listCombination()
```

輸出如下:

This is my first sting.

1|4|7|2|5|8|3|6|9

整數無法直接串接成字串，因此必須經過一層轉換。

正規表示式

字串擷取樣板 –

透過自訂的字串擷取規則，將符合某樣式的子字串做自動化的擷取、處理與應用。

ex: Given an example sentence: "This sentence contains multiple numbers inclusive of 2, 14, 3.5, 141 and 54.21.". Find all numbers within this sentence.

ex: Given another sentence: "Both YES and STAT1 were verified as direct miR-145 targets.". Find all small molecules within this sentence.

正規表示式 Cont.

特殊字元 (Metacharacter)

字元	描述
^	字串的開頭
$	字串的結尾
.	除了換行以外的萬用字元
\d	對應0到9的任一字元，等於[0-9]
\D	對應非數字的任一字元，等於[^0-9]
\w	對應任何文數字及_，等於[a-zA-Z0-9_]
\W	對應任何非文數字及_，等於[^a-zA-Z0-9_]
\s	對應空白字元，等於[\f\n\r\t\v]
\S	對應非空白字元，等於[^\f\n\r\t\v]
\n	對應換行字元
\t	對應 tab字元
\f	對應換頁字元
|	邏輯OR

數量定義詞 (Quantifier)

字元	描述
?	某字元出現0次或1次
*	某字元出現0次或多次
+	某字元出現1次以上
{M}	某字元出現M次
{M, }	某字元出現至少M次
{M, N}	某字元出現M次以上，N次以下

特殊字元 (Metacharacter)

字元	描述
()	將pattern分組並提供記憶功能
[]	可表示集合字元的任一字元

正規表示式 Cont.

RE示例說明 –

RE	描述	不成立之字串
'a'	含字母"a"的字串，ex: "ab", "bac", "cba"	"xyz"
'a.'	含字母"a"及其後任一字元的字串，ex: "ab", "bac"	"a", "ba"
'^xy'	以"xy"為開頭的字串，ex: "xyz", "xyab"	"axy", "bxy"
'xy$'	以"xy"為結尾的字串，ex: "axy", "abxy"	"xya", "xyb"
'[13579]'	含"1"、"3"、"5"、"7"或"9"的字串，ex: "a3b", "1xy"	"y2k"
'[0-9]'	含數字之字串	不含數字之字串
'[a-z0-9]'	含數字或小寫字母之字串	不含數字及小寫字母之字串
'[a-zA-Z0-9]'	含數字或字母之字串	不含數字及字母之字串
'^b[aeiou]t$'	"bat", "bet", "bit", "bot", "but"	"bxt", "bzt"
'[^0-9]'	不含數字之字串	含數字之字串
'[^aeiouAEIOU]'	不含母音之字串	含母音之字串
'[^\^]'	不含"^"之字串，ex: "xyz", "abc"	"xy^", "a^bc"

正規表示式 Cont.

範例1: 數字擷取 I

```
import re

def numberExtraction():
    sentence = 'This sentence contains multiple \
                numbers inclusive of 2, 14, 3.5, \
                141 and 54.21.'

    numMatch = re.finditer('(\d+(\.\d+)?)', sentence)

    for match in numMatch:
        print(match.group(1), match.start(),
              match.end(), match.group(2))

if __name__ == '__main__':
    numberExtraction()
```

使用finditer方法找出某字串中滿足樣板條件的多個子字串。

自訂字串擷取樣板。當某字串中的子字串滿足樣板條件，即被擷取。

被擷取的子字串依群組依序被存放於不同的group中。

輸出如下:

```
2 77 78 None
14 80 82 None
3.5 84 87 .5
141 113 116 None
54.21 121 126 .21
```

正規表示式 Cont.

範例2: 數字擷取 II

```
import re

def numberExtraction():
    extractNumbers = ''
    sentence = 'This sentence contains multiple \
                numbers inclusive of 2, 14, 3.5, \
                141 and 54.21.'

    numMatch = re.findall('(\d+(\.\d+)?)', sentence)

    for match in numMatch:
        print(match)
        extractNumbers += match[0] + '/'

    print(extractNumbers[0:-1])
```

```
if __name__ == '__main__':
    numberExtraction()
```

使用findall方法找出某字串中滿足樣板條件的多個子字串。

被擷取的子字串依群組依序被存放於Tuple容器中。

輸出如下:

```
('2', '')
('14', '')
('3.5', '.5')
('141', '')
('54.21', '.21')
2/14/3.5/141/54.21
```

正規表示式 Cont.

範例3: 網址擷取

```
import re

def urlExtraction():
    sentence = '<a href="https://www.google.com.tw">'

    findMatch = re.search('href="(.+)"', sentence)

    if findMatch!=None:
        print(findMatch.group(1))
```

使用search方法判斷某字串是否存在子字串滿足樣板條件。

將出現多次的萬用字元群組化。

```
if __name__ == '__main__':
    urlExtraction()
```

輸出如下:

https://www.google.com.tw

正規表示式 Cont.

範例4: 樣式取代

```
import re

def patternReplacement(sentence):
    return re.sub('(mir-\d+)',
                  '<microRNA>\\1</microRNA>',
                  sentence)
```

欲取代的字串樣式。

取得欲取代字串群組。

取代後的字串樣式。

```
if __name__ == '__main__':
    text = 'Both mir-17 and mir-32 are all miRNAs.'

    print(patternReplacement(text))
```

輸出如下:

Both <microRNA>mir-17</microRNA> and <microRNA>mir-32</microRNA> are all miRNAs.

正規表示式 Cont.

範例5: 比對方法比較

```
import re

def patternComparison():
    sentence = 'Compare between match and search.'

    patternMatch1 = re.match('between', sentence)
    patternSearch1 = re.search('between', sentence)
    patternMatch2 = re.match('Compare', sentence)
    patternSearch2 = re.search('Compare', sentence)

    print(type(patternMatch1))
    print(type(patternSearch1))
    print(type(patternMatch2))
    print(type(patternSearch2))
```

```
if __name__ == '__main__':
    patternComparison()
```

輸出如下:

```
<class 'NoneType'>
<class '_sre.SRE_Match'>
<class '_sre.SRE_Match'>
<class '_sre.SRE_Match'>
```

re.match只比對字串的開始，如果字串開始不符合正規表示式，則比對失敗，函式回傳None。

re.search比對整個字串，直到找到一個配對結果。

例外處理

錯誤防堵–

當程式錯誤發生時，會產生例外。針對例外發生時所採取的處置措施，即稱為例外處理。

```
try:
    嘗試執行的指令
except 例外名稱 as 變數名稱:
    例外發生時執行的指令
else:
    若try沒有例外則執行此區塊的指令
finally:
    不管有無例外皆會執行的指令
```

except區塊可以多個，但至少一個。as 變數名稱可有可無。若存在，則變數會儲存例外事件所對應的訊息。

else及finally可有可無

例外處理 Cont.

範例1: 捕捉除法運算錯誤

```
def errorHandling():
    try:
        result = 10/0
    except ZeroDivisionError as errMsg:
        print('Error occurred!')
        print('Error Message:', errMsg)
    else:
        print('No problems!')
    finally:
        print('Show the text after error handling!')
```

```
if __name__ == '__main__':
    errorHandling()
```

result = 10/0
輸出:
Error occurred!
Error Message: division by zero
Show the text after error handling!

result = 10/2
輸出:
No problems!
Show the text after error handling!

例外處理 Cont.

範例2: 捕捉多例外錯誤 I

```
def errorHandling():
    myList = [1, 2, 3, 4, 5, 6]

    try:
        getResult = 10/'a'
        getValue = myList[7]
    except ZeroDivisionError as errMsg:
        print('Error (' +str(errMsg)+ ') occurred!')
    except IndexError as errMsg:
        print('Error (' +str(errMsg)+ ') occurred!')
    except:
        print('Some error occurred!')
    finally:
        print('Show the text after error handling!')
```

```
if __name__ == '__main__':
    errorHandling()
```

getResult = 10/0
輸出:
Error (division by zero) occurred!
Show the text after error handling!

getValue = myList[7]
輸出:
Error (list index out of range) occurred!
Show the text after error handling!

getResult = 10/'a'
輸出:
Some error occurred!
Show the text after error handling!

例外處理 Cont.

範例3: 捕捉多例外錯誤 II

```
import sys

def errorHandling():
    myList = [1, 2, 3, 4, 5, 6]

    try:
        getResult = 10/'a'
        getValue = myList[7]
    except (ZeroDivisionError, IndexError) as errMsg:
        print('Error (' +str(errMsg)+ ') occurred!')
    except:
        print('Some error (' +str(sys.exc_info()[1])+ ') occurred!')
```

```
if __name__ == '__main__':
    errorHandling()
```

getResult = 10/0
輸出:
Error (division by zero) occurred!

getValue = myList[7]
輸出:
Error (list index out of range) occurred!

getResult = 10/'a'
輸出:
Error (unsupported operand type(s) for /: 'int' and 'str') occurred!

本週課程終於結束囉^^

檔案讀寫 cont.

範例4: 讀檔方式比較

```
def readShowText(fileName):
    readTxt = open(fileName, 'r')

    for line in readTxt:
        print(line.strip())

    readTxt.close()

if __name__ == '__main__':
    readShowText('TestFile.txt')
```

```
def readShowText(fileName):
    with open(fileName, 'r') as readTxt:
        for line in readTxt:
            print(line.strip())

if __name__ == '__main__':
    readShowText('TestFile.txt')
```