文本分析與程式設計

Week 1

本課程由 卓騰語言科技贊助

學習目標

- 知道文本分析的定義和應用
- 知道什麼是對電腦來說是「字」、「字符」、「詞」
- 知道ArticutAPI 這個斷詞系統怎麼使用
- 本課程使用的文本是從
- 1. https://news-taiwan.xyz/uncategorized/39053.html
- 2. https://www.ctwant.com/article/111388 (有經過編輯)

什麼是「文本分析」？

1. 什麼是語言？

- 我們需要先知道語言和文字的不同
- 語言 = 語音 + 意義 + 結構
- 一般而言，語言可以理解成我們所說出的話，所以有包括我們說話所發聲的聲音，這一串聲音是由句法結構的，說出的聲音也組建成不同的意思

2. 什麼是文本？

- 文字 = 語言的記號
- 文本是指所有以「文字」為記號來呈現「語言」的內容，例如我們用電腦打字打下來的內容。

2. 什麼是文本？

- 所以文本分析的意思就是指說我們使用程式語言來幫我們解構文本中的內容，方便我們做事後的「分析」。
- 分析的內容可以很廣泛，甚至說可以說只要是文字相關都可以算是某種文字分析，從簡單的頻率計算到如何讓電腦自動幫我們分辨文本類別。
- 例如讓電腦從多篇新聞中去學會分類某個文本是屬於運動新聞還是政治新聞，就是文本分類的其中一個例子。

3. 文本分析常見應用

- 文本分析是屬於自然語言處理(Natural Language Processing, NLP) 的一部分，指處理文字方面的相關工作。
- 而自然語言處理則是人工智慧底下的一個子類別。
- 黃色框框的是常見的文本分析任務

4. 文本分析常見應用解析

- 資訊檢索：從大量之中找到關鍵詞或是關鍵事實
- 文本探勘：處理大量資料來發現文本中的趨勢或是隱藏的訊息
- 語意分析：讓電腦整理出段落或是文章大意

4. 文本分析常見應用解析

- 意圖分析：讓電腦整理出作者可能背後意圖
- 機器翻譯 ：由機器翻譯不同語言的內容
- 文本分類：利用電腦將大量資料依照文本內容分類

4. 文本分析常見應用解析

- 文本生成：給機器一部分資料之後，可以生成更多文字，有點像是電腦自己寫文章
- 情緒分析：將不同的文章、語句或是字詞依照不同情緒分類
- 自動語音辨識：電腦讀取文字之後可以轉成語音，或是收到的語音轉成文字
- 自動語音合成：用人工方式合成語音

為什麼需要文本分析呢？

- 由上述的例子，我們可以發現文本分析的主要目標是希望電腦可以處理更多重複的事情，換言之，可以幫我們節省時間

文本分析第一步：斷詞

- 文本分析的第一步其實就是要先讓電腦可以先知道什麼是字。
- 唯有如此，這樣電腦才能先知道怎麼樣讀懂句子和文章。

什麼是字？

- 對電腦來說，怎麼樣是一個「字」呢？
- 如果電腦得到以下字串：「啊我朋友就認識一個家裡養了綿羊的小朋友 」，電腦會將這些字詞做以下分類

什麼是字？

- 字 (word): 句子裡的獨立意義段落，例如「朋友」
- 字符 (character): 字碼表裡的獨立符號，例如「朋」這個字
- 詞 (phrase): 字+詞綴(構詞/句法) ，例如「養了」

什麼是字？

- 符記 (token): 自定切分規格後的結
- 詞彙 (lexicon): 字典中列出的獨立項
- 詞條 (entry): 資料庫中列出的獨立項目

什麼是字？

- 如果想要了解更多關於NLP 或文字分析可以閱讀這篇文章：http://bit.ly/nlp-sinica

課間練習1

- 請問下面這段文字有幾個字？幾個字符？幾個詞？
- 「這個星期日本想往後山藥師佛寺去世人罕至處想一想自己的人生」

認識斷詞系統：

ArticutAPI 套件介紹

- Articut 是一套「純台灣製造」的中文 NLP 系統。它能同時處理中文斷詞、詞性標記以及命名實體標記的套件。相較於 Jieba 分詞、Stanford CoreNLP 以及中研院的 CKIP 或是其它基於簡體字的語料訓練出來的 HanLP、哈工大LTP...等等方案，Articut 具有功能完備、應用靈活及對新詞的接受度高且更新迅速的特點。

ArticutAPI 套件介紹

- Articut 需在 Python3.6 以上的環境中運作！
- 本週課程的完整程式碼可在以下github中取得
  https://github.com/Droidtown/NLP_Training/blob/main/Unit01/

ArticutAPI 套件介紹

- 如果想看jupyter notebook 的範例，請至以下 github
- https://github.com/howardsukuan/textual-data-analysis-python.git

安裝 ArticutAPI

- 在電腦下指令有兩種方法
  - 一種是可以開CMD (命令提示字元，windows) ，或是 linux 或是 mac 的 terminal內輸入
  - 如果是使用Anaconda 內的jupyter notebook，可以在 jupyter notebook 中直接執行 `!pip3 install ArticutAPI` 或是開啟anaconda prompt 中開啟使用

安裝 ArticutAPI

- ArticutAPI 則是可透過 Python 的套件系統 (pip) 進行安裝並操作 Articut的 API 介面。
- 只要在電腦裡下以下指令擇一使用，即可安裝完成

```
pip3 install ArticutAPI
```

```
python3 -m pip install ArticutAPI
```

註冊並取得教學用 API 金鑰

卓騰語言科技免費提供一個月無字數限制之 Articut 教學用金鑰給中華民國教育部承認之教學單位授課使用。請先至 https://api.droidtown.co 完成註冊，並來信 info@droidtown.co 載明：

1. 授課教師姓名
2. 授課教師 email (必需和前述註冊帳號一致)
3. 授課大綱

經查證後，即可取得「一個月」無字數限制之 Articut 教學用金鑰。 再登入 https://api.droidtown.co後，即能在以下畫面取得 Articut NLP 系統的操作金鑰。

金鑰可分給課堂學生使用。

點擊 [複製 API 金鑰] 鈕，即可取得金鑰

基本操作範例：輸入-語法講解

```
from ArticutAPI import Articut #呼叫ArticutAPI套件
from pprint import pprint      #載入 pprint相關套件


if __name__ == "__main__":

    username = "" #這裡您註冊之droidtown email
        apikey   = "" #這裡您註冊之droidtown 後所得到的 api key
    articut = Articut(username, apikey)

    inputSTR = "會被大家盯上，才證明你有實力" #輸入中文字串
    resultDICT = articut.parse(inputSTR) #將結果存在 resultDICT 的變數中
    pprint(resultDICT)
```

基本操作範例：輸出

執行結果


```
{ 'document': 'https://api.droidtown.co/document/',
  'exec_time': 0.09083318710327148,
  'level': 'lv2',
  'msg': 'Success!',
  'product': 'https://api.droidtown.co/product/',
  'result_obj': [[{'pos': 'MODAL', 'text': '會'},
                  {'pos': 'ACTION_lightVerb', 'text': '被'},
                  {'pos': 'ENTITY_nouny', 'text': '大家'},
                  {'pos': 'ACTION_verb', 'text': '盯'},
                  {'pos': 'RANGE_locality', 'text': '上'}],
                 [{'pos': 'PUNCTUATION', 'text': '．'}],
                 [{'pos': 'MODAL', 'text': '才'},
                  {'pos': 'ACTION_verb', 'text': '證明'},
                  {'pos': 'ENTITY_pronoun', 'text': '你'},
                  {'pos': 'ACTION_verb', 'text': '有'},
                  {'pos': 'ENTITY_noun', 'text': '實力'}]],
  'result_pos': ['<MODAL>會</MODAL><ACTION_lightVerb>被</ACTION_lightVerb><ENTITY_nouny>大家</ENTITY_nouny>
                  <ACTION_verb>盯</ACTION_verb><RANGE_locality>上</RANGE_locality>',
                 '．',
                  <MODAL>才</MODAL><ACTION_verb>證明</ACTION_verb><ENTITY_pronoun>你</ENTITY_pronoun><ACTION_verb>有
                  </ACTION_verb><ENTITY_noun>實力</ENTITY_noun>'],
  'result_segmentation': '會/被/大家/盯/上/．/才/證明/你/有/實力',
  'status': True,
  'version': 'v235',
  'word_count_balance': 263}
```


將剛剛輸入之中文斷詞需要之時間

使用哪一種程度的斷詞

斷詞後是屬於哪種詞性(pos) 以及對應的字詞(text)

將詞性及字詞放在同一句表示

僅顯示斷詞結果

- 紅色箭頭和綠色字為解釋

進階用法之一：lv1 和 lv2 輸入

- Articut 可以針對不同的需求，來自由選擇斷詞的細緻程度。
- Articut Level 意指斷詞的深度。數字愈小，切得愈細。例如lv1 會些比較細一些。
- 使用的方法就是在 articut.parse不只放入您想斷詞的中文字串，也要調整level 參數

進階用法之一：lv1 和 lv2 輸入

- 比如說「小紅帽」
- 在 lv1 的設定下，
  將會回傳為 [小/紅/帽]。
- 但在 lv2 的設定下，
  則會回傳 [小紅帽]。

```
from ArticutAPI import Articut

if __name__ == "__main__":

    username = ""
           apikey   = ""
    articut = Articut(username, apikey)

    inputSTR = "小紅帽"
    resultDICT = articut.parse(inputSTR, level='lv1')
    print("\n lv1 的設定下，處理結果：\n", resultDICT)

    resultDICT = articut.parse(inputSTR, level='lv2')
    print("\n lv1 的設定下，處理結果：\n", resultDICT)
```

進階用法之一：lv1 和 lv2 輸出

- 比較 lv1 和 lv2 的結果，可發現在 lv1 中切的極細的「小 / 紅 / 帽」，在 lv2 中被結合成「小紅帽」一個詞彙。

執行結果

lv1 的設定下，處理結果：


```
{'document': 'https://api.droidtown.co/document/',
 'exec_time': 0.03942060470581055,
 'level': 'lv1',
 'msg': 'Success!',
 'product': 'https://api.droidtown.co/product/',
 'result_obj': [[{'pos': 'MODIFIER', 'text': '小'},
                 {'pos': 'MODIFIER_color', 'text': '紅'},
                 {'pos': 'ENTITY_nounHead', 'text': '帽'}]],
 'result_pos': ['<MODIFIER>小</MODIFIER><MODIFIER_color>紅</MODIFIER_color><ENTITY_nounHead>帽</ENTITY_nounHead>'],
 'result_segmentation': '小/紅/帽',
 'status': True,
 'version': 'v236',
 'word_count_balance': 1933}
```


lv2 的設定下，處理結果：


```
{'document': 'https://api.droidtown.co/document/',
 'exec_time': 0.04227423667907715,
 'level': 'lv2',
 'msg': 'Success!',
 'product': 'https://api.droidtown.co/product/',
 'result_obj': [[{'pos': 'ENTITY_nouny', 'text': '小紅帽'}]],
 'result_pos': ['<ENTITY_nouny>小紅帽</ENTITY_nouny>'],
 'result_segmentation': '小紅帽',
 'status': True,
 'version': 'v236',
 'word_count_balance': 1930}
```

進階用法之一：lv1 和 lv2

- 同理，在 lv1 下會把動詞和時態標記分開，因此「創造了」會被切分成「創造/了」；但在 lv2 的設定下，則會把動詞和時態標記結合在一起，因此「創造了」將在 lv2 處理為「創造了」。

進階用法之一：lv1 和 lv2

- 以下為範例

```
from ArticutAPI import Articut

if __name__ == "__main__":

    username = “”
        apikey   = “”
    articut = Articut(username, apikey)

    inputSTR = "小紅帽"
    resultDICT = articut.parse(inputSTR, level='lv1')
    print("\n lv1 的設定下，處理結果：\n", resultDICT)

    resultDICT = articut.parse(inputSTR, level='lv2')
    print("\n lv1 的設定下，處理結果：\n", resultDICT)
```

進階用法之一：lv1 和 lv2

- 這是因為 Articut 將時態標記「了」視為像是英文裡的 -ed。因此，在 lv1 時，採取將之處理為 "create/-ed" 分開的兩個元素，但在 lv2 的設定下，則是以 "created" 這種「詞 + 時態標記」的形式輸出。

執行結果

lv1 的設定下，處理結果：

{'document': 'https://api.droidtown.co/document/',
 'exec_time': 0.03736257553100586,
 'level': 'lv1',
 'msg': 'Success!',
 'product': 'https://api.droidtown.co/product/',
 'result_obj': [[{'pos': 'ACTION_verb', 'text': '創造'},
                 {'pos': 'ASPECT', 'text': '了'}]],
 'result_pos': ['<ACTION_verb>創造</ACTION_verb><ASPECT>了</ASPECT>'],
 'result_segmentation': '創造/了',
 'status': True,
 'version': 'v236',
 'word_count_balance': 345}

lv2 的設定下，處理結果：

{'exec_time': 0.08351516723632812, 'result_pos': ['<MODAL>會</MODAL><ACTION_lightVerb>被</ACTION_lightVerb><ENTITY_noun>大家</ENTITY_noun><ACTION_verb>盯</ACTION_verb><RANGE_locality>上</RANGE_locality>', '，', '<MODAL>才</MODAL><ACTION_verb>證明</ACTION_verb><ENTITY_pronoun>你</ENTITY_pronoun><ACTION_verb>有</ACTION_verb><ENTITY_noun>實力</ENTITY_noun>'], 'result_segmentation': '會/被/大家/盯/上/，/才/證明/你/有/實力', 'result_obj': [[{'text': '會', 'pos': 'MODAL'}, {'text': '被', 'pos': 'ACTION_lightVerb'}, {'text': '大家', 'pos': 'ENTITY_noun'}, {'text': '盯', 'pos': 'ACTION_verb'}, {'text': '上', 'pos': 'RANGE_locality'}], [{'text': '，', 'pos': 'PUNCTUATION'}], [{'text': '才', 'pos': 'MODAL'}, {'text': '證明', 'pos': 'ACTION_verb'}, {'text': '你', 'pos': 'ENTITY_pronoun'}, {'text': '有', 'pos': 'ACTION_verb'}, {'text': '實力', 'pos': 'ENTITY_noun'}]], 'level': 'lv2', 'version': 'v236', 'status': True, 'msg': 'Success!', 'word_count_balance': 354, 'product': 'https://api.droidtown.co/product/', 'document': 'https://api.droidtown.co/document/'}
{'document': 'https://api.droidtown.co/document/',
 'exec_time': 0.04187440872192383,
 'level': 'lv2',
 'msg': 'Success!',
 'product': 'https://api.droidtown.co/product/',
 'result_obj': [[{'pos': 'VerbP', 'text': '創造了'}]],
 'result_pos': ['<VerbP>創造了</VerbP>'],
 'result_segmentation': '創造了',
 'status': True,
 'version': 'v236',
 'word_count_balance': 342}

課間練習2

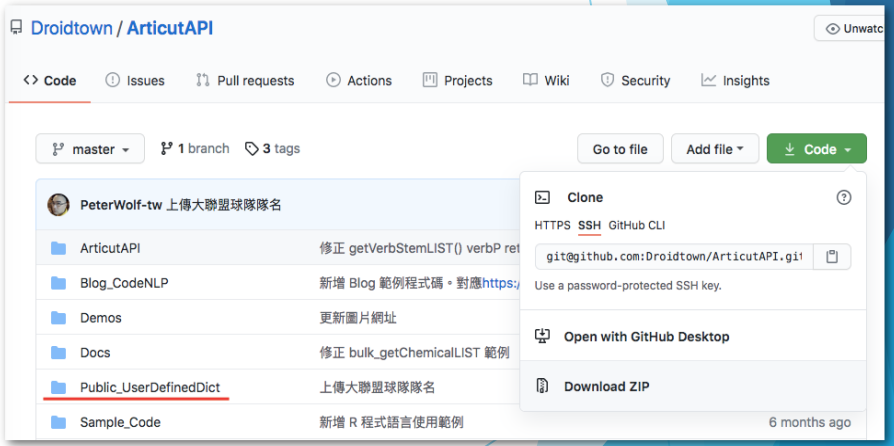
- 仿照前例，用 Python3 設計一段程式，分別用 "lv1" 和 "lv2" 輸入「閱讀創造了奇蹟」，觀察 lv1 和 lv2 的設定下，回傳的結果有何差別。

進階用法之二：載入自訂字典

- 處理不同領域的文本時，我們可以載入相應的領域字典以便增加處理結果的正確率。以下示範載入 MLB 和 NBA 兩種字典來處理棒球和籃球的語料結果。

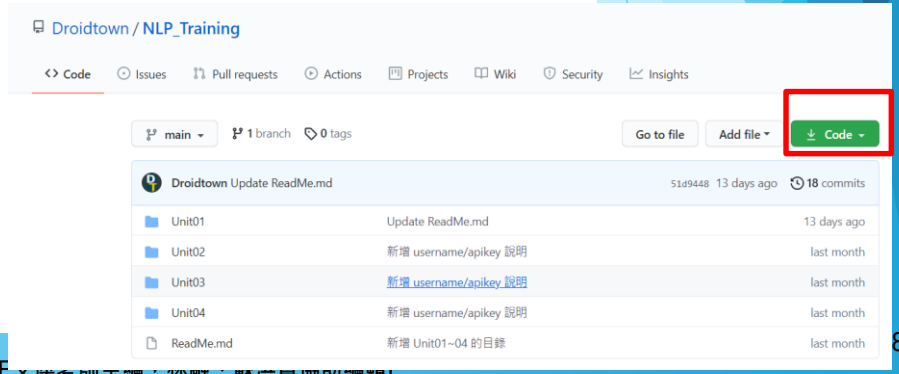
進階用法之二：載入自訂字典

- 字典檔的存放位置在 ArticutAPI 的 Github.com 專案內
- 網址是：https://github.com/Droidtown/ArticutAPI



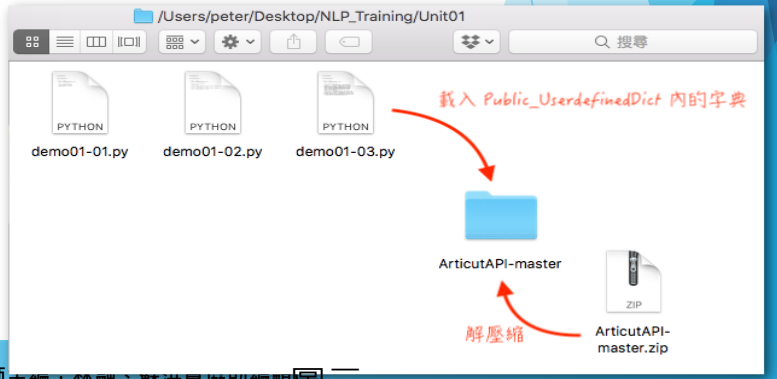
進階用法之一：lv1 和 lv2

- 點擊圖中畫面右方的 [Code] 找到 [Download ZIP] 的選項，下載壓縮檔，此次下載會將所有 NLP_Training 內容下載下來，然後將它解壓縮。



進階用法之一：lv1 和 lv2

- 最後請把ArticutAPI-master 放在您使用的範例檔案旁邊，這樣後面示範才讀的到檔案 (如圖，如果demo01-03.py 為範例檔，那ArticutAPI-master 要放在它旁邊)



圖二

進階用法之一：lv1 和 lv2

- 由於在之後的課程裡，我們將以「體育運動」類文章作範例，所以我們會使用**ArticutAPI-master** 的 **Public_UserdefinedDict** 內幾個球隊名稱的領域字典。
- 我們將這幾個領域字典讀取出來，並結合成單一一個字典檔，和要處理的文本一起傳給 Articut 以便增加結果的正確性。

進階用法之一：lv1 和 lv2

- 以下為範例語料 (可以複製然後使用)

棒球語料：

本週三在紐約的比賽中，馬林魚此戰使用投手車輪戰，4名投手輪番上陣壓制大都會打線，前8局僅被敲出4支安打失1分，讓球隊能帶著2-1的領先優勢進入到9局下半。不過馬林魚推出巴斯登板關門，他面對首名打者麥尼爾，就被打出一發陽春砲，讓大都會追平比數，接下來又分別被敲出2支安打、投出保送，形成滿壘局面，此時輪到康福托上場打擊。在2好1壞的局面下，巴斯投了一顆內角滑球，康福托眼看這顆球越來越靠近自己的身體，似乎有下意識地將手伸進好球帶內，結果這球就直接碰觸到他的身肘，隨後主審庫爾帕判定這是一記觸身球，讓大都會兵不血刃拿下再見分，最終贏得比賽勝利。

籃球語料：

昨晚的紐約西區霸王之戰中，錯失勝利的太陽沒有就此束手就擒，延長賽一開始就打出7比2攻勢，米契爾和康利雖然力圖追分，但太陽總能馬上回應。康利讀秒階段上籃得手，布克兩罰一中，再次留給爵士追平機會。米契爾造成犯規，可惜兩罰一中，保羅隨後用兩罰鎖定勝利。米契爾狂轟41分8籃板3助攻，本季單場得分次高；戈貝爾16分18籃板3抄截，波格丹諾維奇20分。康利拿到11分4助攻，克拉克森11分，兩人合計28投僅9中。爵士的三分攻勢難以有效施展，全場44投僅11中。

課間練習3

仿照後一頁的例子，結合兩個字典檔成為一個 mixedDICT.json 檔以後，分別將 basebalSTR 和 basketballSTR 輸入 Articut 進行處理並取得回傳結果。

課間練習3

```
#!/usr/bin/env python3
# -*- coding:utf-8 -*-
from ArticutAPI import Articut
import json
if __name__ == "__main__":

        username = ""
        apikey   = ""
    articut = Articut(username, apikey)

     #以下語料可以參考
     baseballSTR = "^" "^".replace(" ", "") #參考第40頁語料
     basketballSTR = "^" """.replace(" ", "") #參考第40頁語料
```

課間練習3

```
#將 KNOWLEDGE_NBA_Teams.json 和 KNOWLEDGE_MLB_Teams.json 兩個體育欸字典讀取出來，合併成 mixDICT 以後，寫入 mixedXICT.json 檔
    with open("ArticutAPI-master/Public_UserDefinedDict/KNOWLEDGE_NBA_Teams.json", encoding="utf-8") as f:
        nbaDICT = json.loads(f.read())
    with open("ArticutAPI-master/Public_UserDefinedDICT/KNOWLEDGE_MLB_Teams.json", encoding="utf-8") as f:
        mlbDICT = json.loads(f.read())

    mixedDICT = {**nbaDICT, **mlbDICT}
    with open("mixedDICT.json", mode = "w", encoding = "utf-8") as f:
        json.dump(mixedDICT, f, ensure_ascii=False)
```

課間練習3

```
# 將baseballStR 和 basketballSTR 兩篇文本各自送入articut.parse() 裡，同時指定
userDefinedDictFILE 為剛才產生mixedDICT.json
    baseballResultDICT = articut.parse(baseballSTR,
userDefinedDictFILE="./mixedDICT.json")
    basketballResultDICT = articut.parse(baseballSTR,
userDefinedDictFILE="./mixedDICT.json")

    print("\n棒球斷詞結果：\n", baseballResultDICT)
    print("\n籃球斷詞結果：\n", basketballResultDICT)
```

作業:從字頻來分析文本

- 作業連結

任務1:

請用ArticurtAPI將以下兩個文本 medicalSTR 以及 weatherSTR 用lv2進行斷詞，並將斷詞的結果依據頻率進行排序，取前20名。

medicalSTR

指揮中心今天公布新增135例武漢肺炎確定病例，其中132 例為本土個案，另有3例境外移入；確診個案中新增8例死亡。指揮官陳時中說，病例及死亡數減少是好現象。

中央流行疫情指揮中心指揮官陳時中下午在記者會中說明，新增132例武漢肺炎（2019冠狀病毒疾病，COVID-19）本土病例，為62例男性、70例女性，年齡介於未滿5歲至80多歲，發病日介於6月1日至6月14日。

陳時中說，個案分布以新北市65例最多，其次為台北市26例，苗栗縣18例，桃園市12例，基隆市3例，台南市、台中市及花蓮縣各2例，嘉義縣及彰化縣各1例。其中雙北地區以外縣市41例中，33例為已知感染源、6例關聯不明、2例調查中；相關疫情調查持續進行中。...

weatherSTR

今天是一年一度的端午佳節，台語有句俗話說：「未食端午粽，破裘不甘放」，形容過了端午之後，天氣才會穩定炎熱，終於可以把冬衣給收起來了，實際上今天(14日)開始台灣附近西南風逐漸增強，未來這一週的確都是暖熱且潮濕的西南風影響，溫度普遍都偏高，尤其是位在西南風背風面的北台灣，受到西南風過山後沉降增溫作用的加持，接下來幾天都很可能出現超過36度以上的高溫，其他像是中部地區、花東地區高溫也都有33-35度，只有直接面迎西南風的南部地區因為雲量較多又有機會下雨，高溫相對較低只有30-32度，這樣的狀況預期會持續一週左右，提醒大家要留意高溫，外出要多喝水預防中暑，同時做好防曬。...

任務1所需能力

- 解讀ArticutAPI斷詞後的字典檔:

因為他回傳的檔案是字典檔，我們可以運用以下方式取得斷詞結果(回傳的會是一個字串)

```
resultDICT['result_segmentation']
```

任務1所需能力

- 將字串進行分割:

因為回傳結果是整個字串

例如: \n/指揮中心/今天/公布/新增/135例/武漢/肺炎/確定/病例/

所以我們必須將字串依據”/”，的符號進行分割

任務1所需能力

- 將字串進行分割:

我們可以使用.split()這個功能，如下

```
inputSTR = "\n/指揮中心/今天/公布/新增/135例"
inputSTR.split("/") #請電腦依據 "/"切開字串
```

Output: ['\n', '指揮中心', '今天', '公布', '新增', '135例']

任務1所需能力

- 根據切割好的LIST去統計每個字出現頻率，並轉成dataframe

```
import pandas as pd
wordDF = pd.DataFrame({"WORD":["apple","apple","guava"]})
#將我們做好的list存成dictionary然後轉成dataframe
wordDF = pd.value_counts(wordDF.WORD).to_frame().reset_index()
# value_counts()會自己去數設定欄位中字詞現的頻率
wordDF.columns = ['WORD','FREQ']
#幫新做好的dataframe命名
wordDF
```

:

	WORD	FREQ
0	apple	2
1	guava	1

作業:從字頻來分析文本

- 任務2:

經過斷詞還有頻率的分析，我們大概可以知道每篇文章的代表詞彙，但是在這些文章中的前20名裡頭，我們發現有一些標點符號(、，。)以及換行符號(\n)也被包含到分析內，這些標點符號對於目前的文本貢獻性不大，因此如果可能的話我們會盡量將其移除。

任務2所需能力

- 將字串特定內容進行取代: 使用re.sub()

```
import re
testSTR = "我~~喜歡文本分析~~~~!!!!!!!"
re.sub("[~!]","",testSTR)
#re.sub(要取代的東西,取代成什麼樣子,要被處理的字串)
```

Output: "我喜歡文本分析"

作業:從字頻來分析文本

- 任務3: 反思

請觀察斷詞後並濾掉標點符號後的表格，這樣的結果其實並不盡人意，有一些意義不大的功能詞，例如"的"，請想想有什麼可能的方法可以讓分析出來的結果更能表達文本的差異以及內容呢?