- 學習source:
  - 使用Google Colab - Python 教學| STEAM 教育學習網Google Colab（Colaboratory）是一個在雲端運行的編輯環境，由 Google 提供一個雲端虛擬主機，支援 Python 程式及機器學習 TensorFlow 演算法，Colab 目的在提供教育訓練以及教學研究，不用下載或安裝，就可直接編輯 Python，並使用 Python 的資源庫，大幅降低初學者的入門門檻，不用耗費太多時間在環境的安裝與設定。在 Colab 裡編輯的程式碼，預設直接儲存在開發者的 Google Drive 雲端硬碟中，執行時由虛擬主機提供強大的運算能力，並不會用到本機的資源。但要如果程式閒置一段時間，會被停止並回收運算資源。
    - https://colab.research.google.com/
    - cliff [交代] 有網路online 編輯器能試錯網上的code

- 這個網頁python 都有而且講很細，我很喜歡
  - (Youtuber: Programiz) https://www.programiz.com/python-programming/datetime/strptime

- **"regular expression正規表示式"（防止user亂input:）**

正規表示式	說明
/a?/	零或一個 a（若要比對? 字元，請使用 \? ）
/a+/	一或多個 a（若要比對+ 字元，請使用 \+ ）
/a*/	零或多個 a（若要比對* 字元，請使用 * ）
/a{4}/	四個 a
/a{5,10}/	五至十個 a
/a{5,}/	至少五個 a
/a{,3}/	至多三個 a
/a.{5}b/	a 和 b中間夾五個（非換行）字元

正規表示式的特定字元	說明	等效的正規表示式
\d	數字	[0-9]
\D	非數字	[^0-9]
\w	數字、字母、底線	[a-zA-Z0-9_]
\W	非 \w	[^a-zA-Z0-9_]
\s	空白字元	[\r\t\n\f]
\S	非空白字元	[^ \r\t\n\f]

正規表示式	說明及範例	比對不成立之字串
/a/	含字母 “a” 的字串，例如 “ab”, “bac”, “cba”	“xyz”
/a./	含字母 “a” 以及其後任一個字元的字串，例如 “ab”, “bac”（若要比對 . ，請使用 \. ）	“a”, “ba”
/^xy/	以 “xy” 開始的字串，例如 “xyz”, “xyab”（若要比對 ^ ，請使用 \^ ）	“axy”, “bxy”
/xy$/	以 “xy” 結尾的字串，例如 “axy”, “abxy”以 “xy” 結尾的字串，例如 “axy”, “abxy”（若要比對 $ ，請使用 \$ ）	“xya”, “xyb”
[13579]	包含 “1” 或 “3” 或 “5” 或 “7” 或 “9” 的字串，例如：”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”

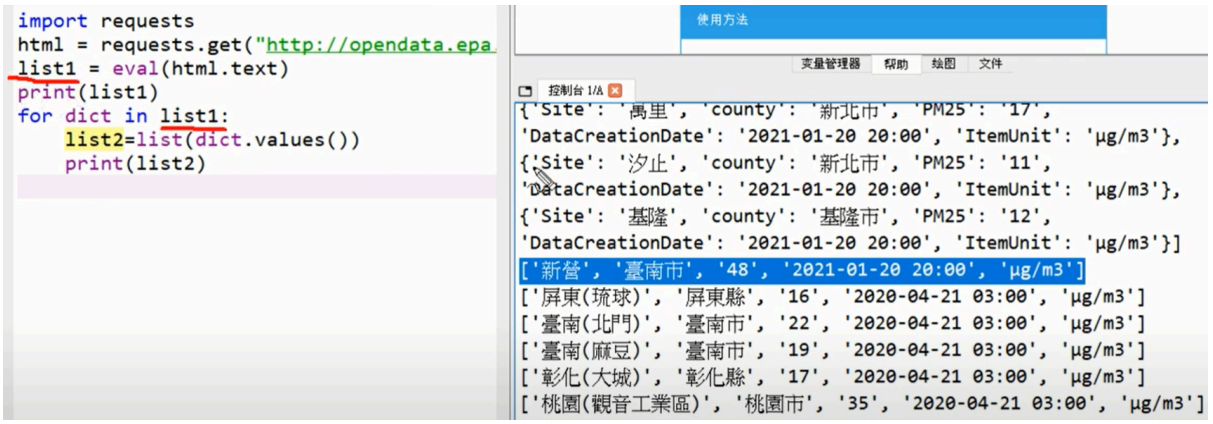
- 字元：就是單純照字面上的意義。例如 dog 可以用來匹配 dog 這字串；hello 101 則用來匹配 hello 101 這字串。
- 或 (or)：`|` 管線符號，用來將所有可能的選擇條件分隔開。例如 gray|grey 可以用來匹配 gray 或是 grey 字串。
- 群組 (grouping)：`()` 小括號，用來表示作用範圍或優先順序。例如 gray|grey 和 gr(a|e)y 都同樣可以用來匹配 gray 或是 grey 字串。
- 量詞 (quantifier)：quantifier 用來接在字符串或群組後面，表示某個條件應該出現「幾次」。常見的量詞有：
  - `?`：表示連續出現 0 次或 1 次。例如 colou?r 可以用來匹配 color 或 colour。
  - `*`：表示連續出現 0 次或多次。例如 ab*c 可以用來匹配 ac, abc, abbc, abbbc 或 abbbbbbc。
  - `+`：表示連續出現 1 次或多次。例如 ab+c 可以用來匹配 abc, abbc, abbbc, abbbbbbc，但 ac 不符合。
  - `{min,max}`：表示至少連續出現 min 次，但最多連續出現 max 次。

- 了解 Json (資料格式的一種，全球通用，之後 MongoDB 也需要)

  -
    - “python JSON add Value”

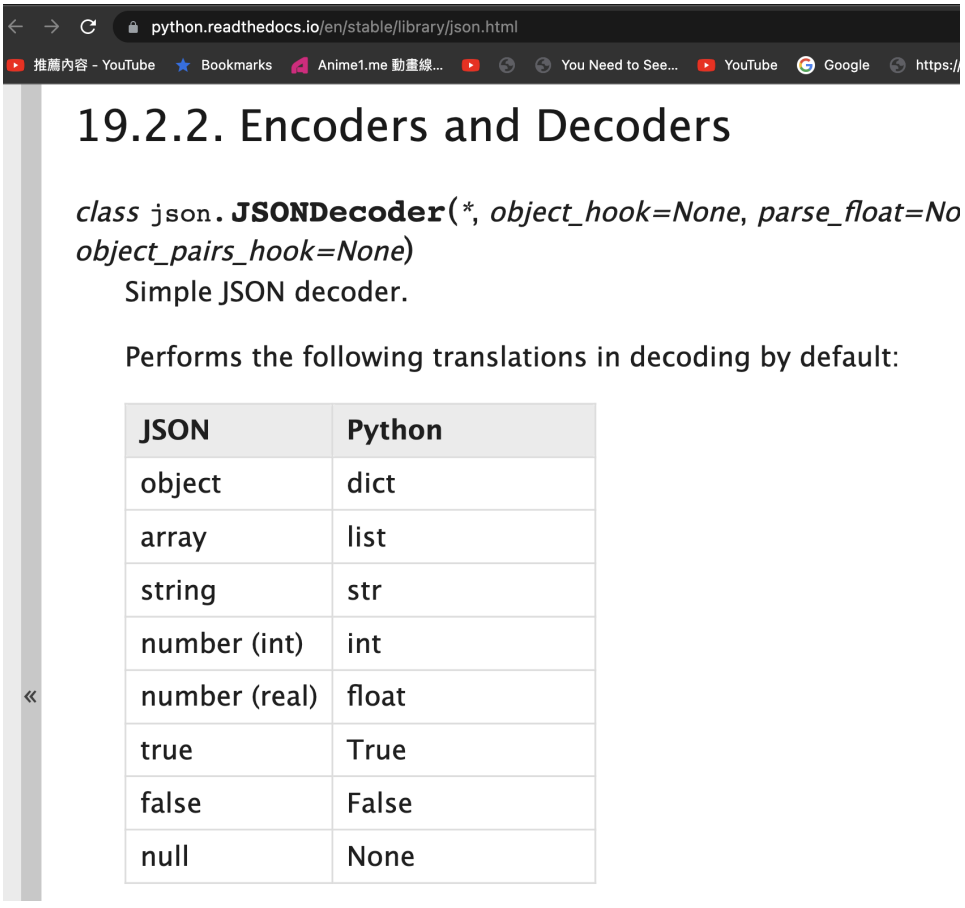
      05 抓取JSON格式與抓取資料

      https://www.youtube.com/watch?v=Z6sAjSHZ7qs&ab_channel=%E5%90%B3%E8%80%81%E5%B8%AB%E6%95%99%E5%AD%B8%E9%83%A8%E8%90%BD%E6%A0%BC



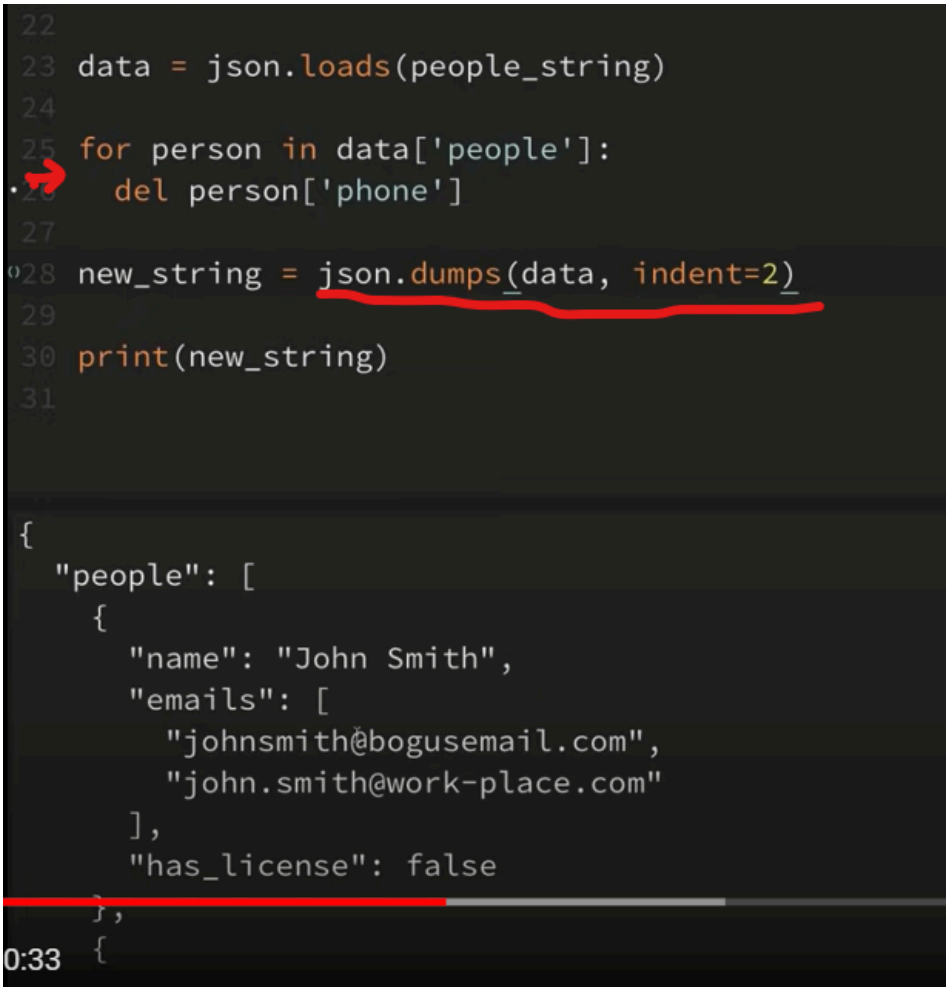
19.2.2. Encoders and Decoders¶ 編碼器和解碼器¶

https://python.readthedocs.io/en/stable/library/json.html

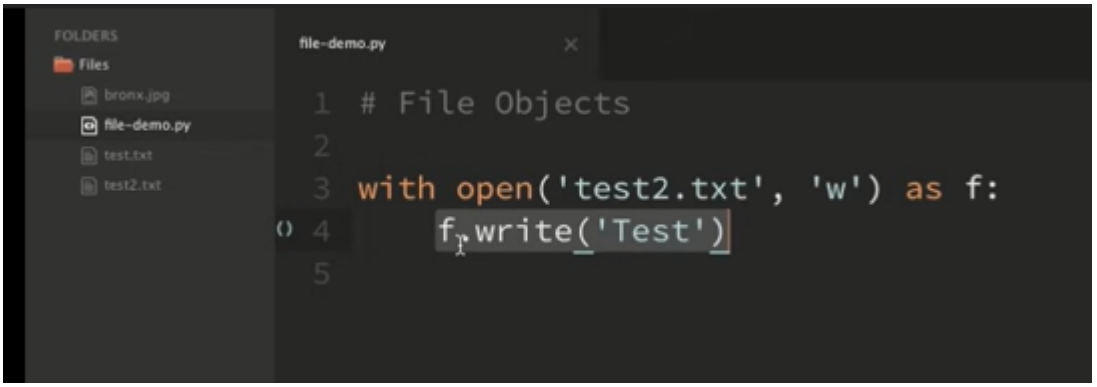


Python Tutorial: Working with JSON Data using the json Module

"when del data , and indent to new format"
https://youtu.be/9N6a-VLBa2I?t=381



- Python Tutorial: File Objects - Reading and Writing to Files
  Python 教程：文件對象 - 讀取和寫入文件
  https://youtu.be/Uh2ebFW8OYM?t=993



Python JSON
https://www.runoob.com/python/python-json.html

Python JSON

本章节我们将为大家介绍如何使用 Python 语言来编码和解码 JSON 对象。

JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式，易于人阅读和编写。

JSON 函数

使用 JSON 函数需要导入 json 库：**import json**。

函数	描述
json.dumps	将 Python 对象编码成 JSON 字符串
json.loads	将已编码的 JSON 字符串解码为 Python 对象

json.dumps

json.dumps 用于将 Python 对象编码成 JSON 字符串。

- 公司物料單 用 regular expression
  - 標題 02
-

==========

- [Json] [Python] import data function能用

  - 注意 json.loads 跟 json.dumps

Python JSON

本章节我们将为大家介绍如何使用 Python 语言来编码和解码 JSON 对象。

JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式，易于人阅读和编写。

JSON 函数

使用 JSON 函数需要导入 json 库：**import json**。

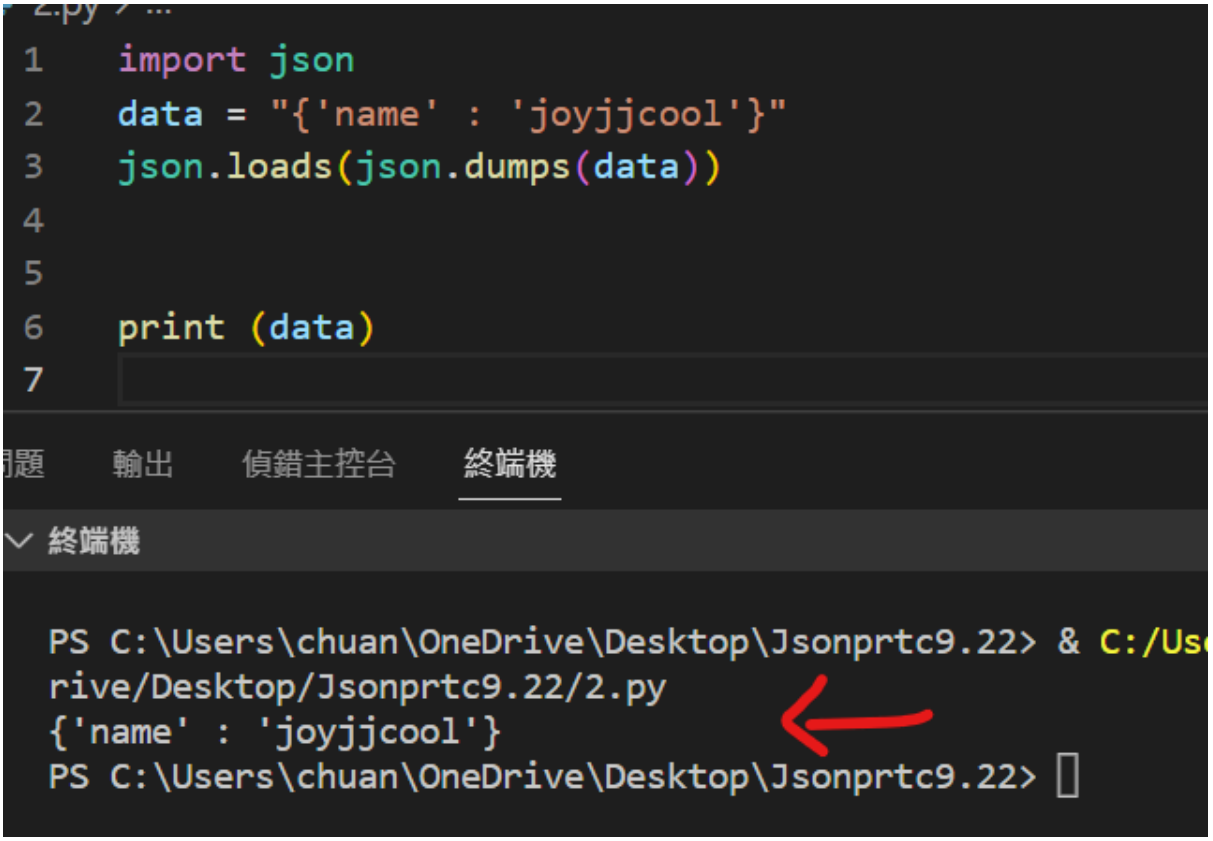
函数	描述
json.dumps	将 Python 对象编码成 JSON 字符串
json.loads	将已编码的 JSON 字符串解码为 Python 对象

json.dumps

json.dumps 用于将 Python 对象编码成 JSON 字符串。

```
import json
data = "{'name' : 'joyjjcool'}"
json.loads(json.dumps(data))


print (data)
```

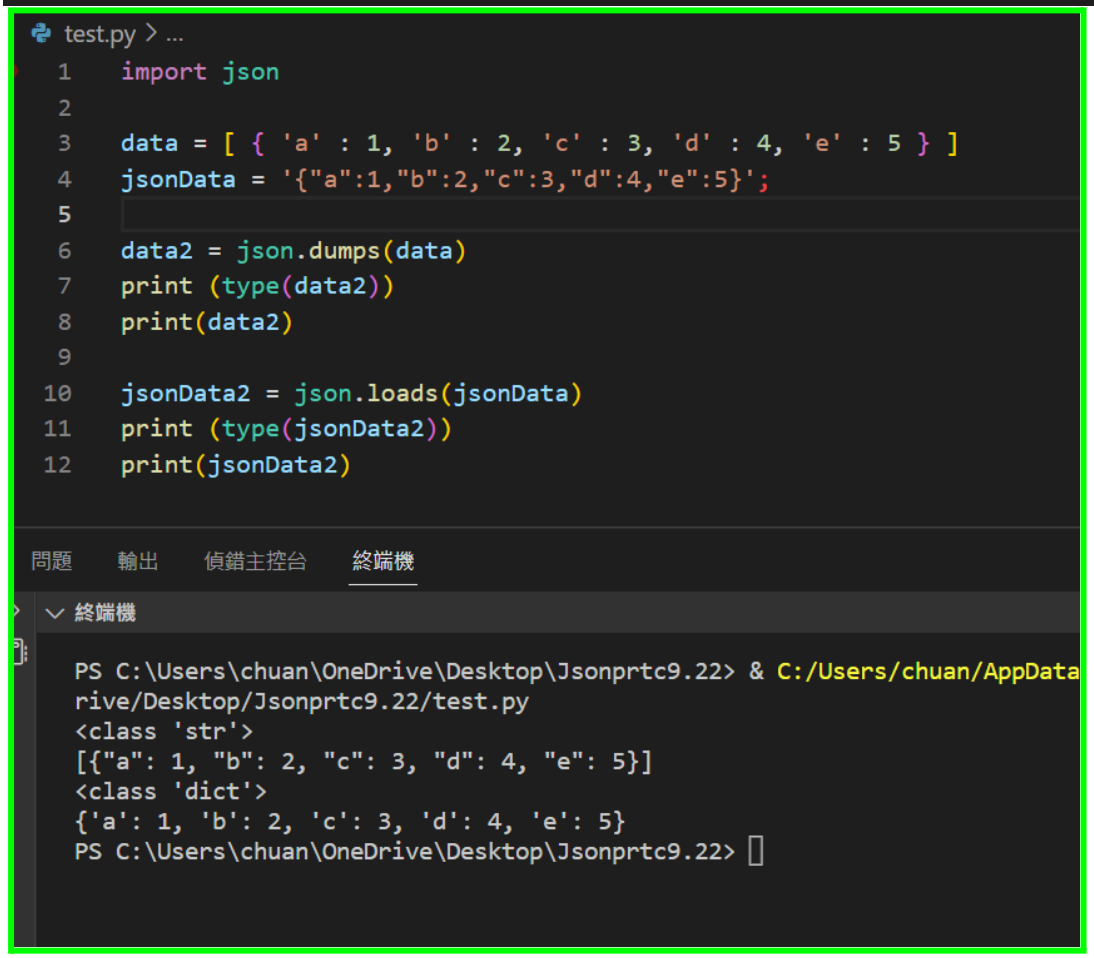


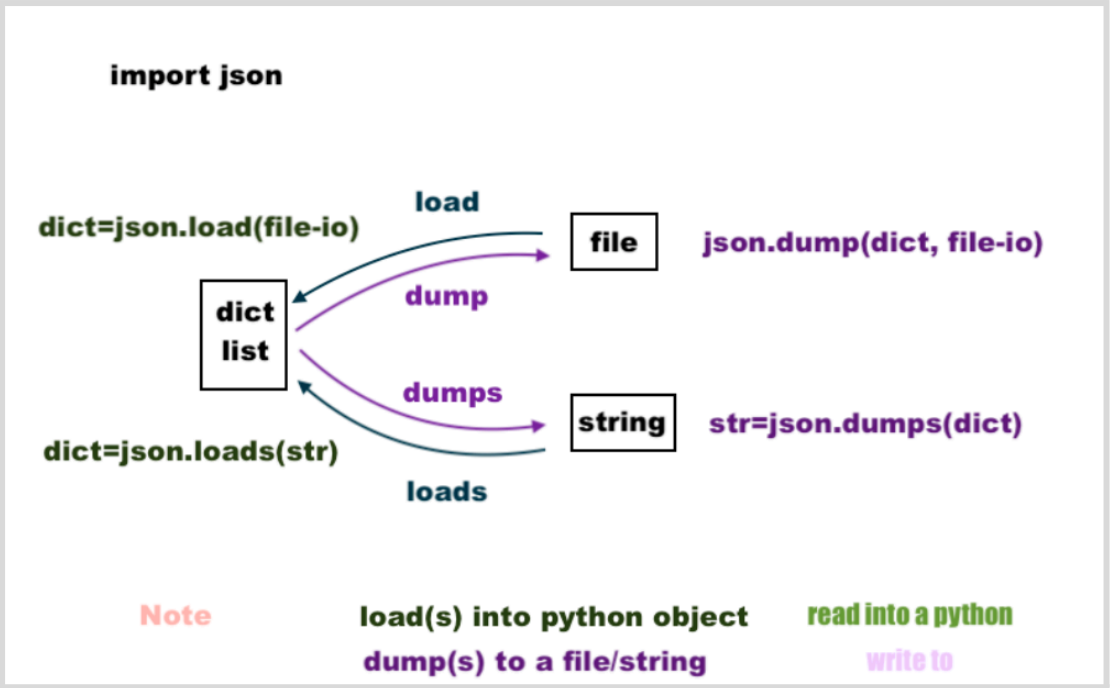
```python
import json

data = [ { 'a' : 1, 'b' : 2, 'c' : 3, 'd' : 4, 'e' : 5 } ]
jsonData = '{"a":1,"b":2,"c":3,"d":4,"e":5}';

data2 = json.dumps(data)
print (type(data2))
print(data2)

jsonData2 = json.loads(jsonData)
print (type(jsonData2))
print(jsonData2)
```





==========

- 關於資料庫的commit以及rollback(轉載)

  https://www.796t.com/content/1549101624.html

rollback　回滾的意思。就是資料庫裡做修改後（update　,insert　, delete）未commit之前　使用rollback　可以恢復資料到修改之前。

從功能上劃分，SQL 語言可以分為DDL,DML和DCL三大類。

1. DDL（Data Definition Language）

資料定義語言，用於定義和管理 SQL 資料庫中的所有物件的語言；

CREATE---建立表

ALTER---修改表

DROP---刪除表

2. DML（Data Manipulation Language）

資料操縱語言，SQL中處理資料等操作統稱為資料操縱語言；

INSERT---資料的插入

DELETE---資料的刪除

UPDATE---資料的修改

SELECT---資料的查詢

3. DCL（Data Control Language）

資料控制語言，用來授予或回收訪問資料庫的某種特權，並控制 資料庫操縱事務發生的時間及效果，對資料庫實行監視等；

GRANT--- 授權。

ROLLBACK---回滾。

COMMIT--- 提交。

4. 提交資料有三種類型：顯式提交、隱式提交及自動提交。

下面分 別說明這三種類型。

(1) 顯式提交

用 COMMIT 命令直接完成的提交為顯式提交。

(2) 隱式提交

用 SQL 命令間接完成的提交為隱式提交。這些命令是：

ALTER， AUDIT， COMMENT， CONNECT， CREATE， DISCONNECT， DROP， EXIT ， GRANT， NOAUDIT， QUIT， REVOKE， RENAME 。

(3) 自動提交

若把 AUTOCOMMIT 設定為 ON，則在插入、修改、刪除語句執行後，

系統將自動進行提交，這就是自動提交。其格式為：SQL>SET AUTOCOMMIT ON ；

COMMIT / ROLLBACK這兩個命令用的時候要小心。COMMIT / ROLLBACK 都是用在執行DML語句（INSERT / DELETE / UPDATE / SELECT ）之後的。DML 語句，執行完之後，處理的資料，都會放在回滾段中（除了 SELECT 語句），等待使用者進行提交（COMMIT）或者回滾（ROLLBACK），當用戶執行 COMMIT / ROLLBACK後，放在回滾段中的資料就會被刪除。

（SELECT 語句執行後，資料都存在共享池。提供給其他人查詢相同的資料時，直接在共享池中提取，不用再去資料庫中提取，提高了資料查詢的速度。）

所有的 DML 語句都是要顯式提交的，也就是說要在執行完DML語句之後，執行 COMMIT 。而其他的諸如 DDL 語句的，都是隱式提交的。也就是說，在執行那些非 DML 語句後，資料庫已經進行了隱式提交，例如 CREATE TABLE，在執行指令碼後，表已經建好了，並不在需要你再進行顯式提交。

- Q9 [Json] [Python] 資料: 能互換，演算datatype; sort 大小排序; change element; 動態插入
    - json內不同data type: date, boolean, int(年齡), float(體重)
    - sort: 依照年齡排序json內容
    - 變更element: 例如把某人的age從11歲變成20歲
    - 動態插入:例如年齡15歲以下的element，全部再加入一個"class":"中學"的element
      (新增備註)

==========

Q11: [Json] [Python] [Excel] 資料的讀，寫

ex: READ WRITE，1. 從URL 抓日期在Excel 裡呈現
2. 在Excel 讀資料

Convert JSON to Excel using json2excel Library in Python
https://www.youtube.com/watch?v=gop5KuQiE_o&ab_channel=CodingDiksha

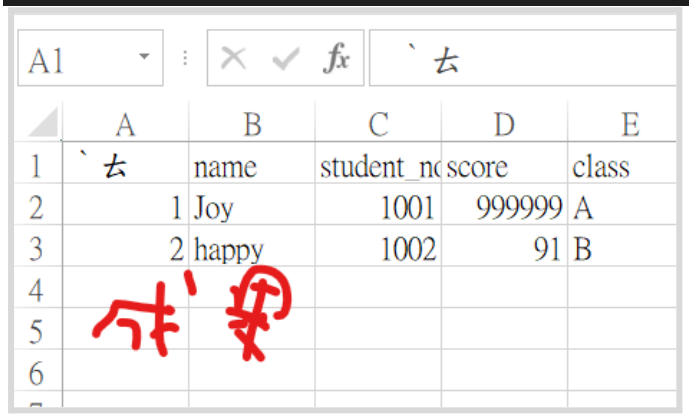
```
from json2excel import Json2Excel

if __name__ == '__main__':
    json2excel = Json2Excel(head_name_cols=["rank", "name"])
    # print(json2excel.run('./test.json'))
```

```
jsons = [
    {
        "rank": 1,
        "student_no": 1001,
        "name": "Joy",
        "score": 999999,
        "class": "A",

    },
    {

        "name": "happy",
        "student_no": 1002,
        "score": 91,
        "class": "B",
        "rank": 2
    },
]
print(json2excel.run(jsons))
```



	A	B	C	D	E
1	` ㄊ	name	student_nc	score	class
2	1	Joy	1001	999999	A
3	2	happy	1002	91	B
4					
5					
6					

==========

- Python 中獲取日期是星期幾

https://www.delftstack.com/zh-tw/howto/python/python-datetime-day-of-week/

1. 在 Python 中使用 `weekday()` 方法獲取日期是星期幾
2. 在 Python 中使用 `isoweekday()` 方法獲取一天的名稱
3. Python 中使用 `calendar` 模組獲取日期是星期幾
4. Python 中使用 Pandas `Timestamp` 方法獲取當天的名稱

在 Python 中使用 `weekday()` 方法獲取日期是星期幾

在 Python 中，`weekday()` 可以用來檢索一週的哪一天。`datetime.today()` 方法返回當前日期，而 `weekday()` 方法則以整數形式返回一週中的某一天，其中週一的索引為 0，週日為 6。

下面給出了該方法的示例程式碼。

```
from datetime import datetime

print(datetime.today().weekday())
```

輸出：

```
1
```

==========

- strftime与strptime之间用法与区别
- https://blog.csdn.net/weixin_42139375/article/details/81105479

相信有部分小伙伴在调用时间对象处理数据表格，会经常遇到这两个函数吧？

对于老程序员来说，自然不在话下，但是对于小萌新来说，多少就有点晕菜了。而且随时把两者混淆，导致程序报错或者出现小bug。结果查询了几个小时，到最后才发现竟然一个字母的差别.....顿时泪崩！！！

strftime

strftime是一种计算机函数，根据区域设置格式化本地时间/日期，函数的功能：将时间格式化！！！，或者说格式化一个时间字符串。

这是出自百度百科的解释。说白了就是按照一定格式把时间显示成所需要的效果。

下面是一段示例代码：辅助理解

```
# 添加当天开始时间字符串到当前日期中

current_date = []  #代表一个容器，即装时间对象的一个小盒子#
current_date.append(begin_date.strftime("%Y-%m-%d"))
```

就是把自己所需要的时间按照年/月/日或者时/分/秒等你想要的格式。他在很多编程语言当中都用到。

strptime

strptime()，功能：按照特定时间格式将字符串转换（解析）为时间类型。

下面还是一段示例代码来辅助理解：

```
#将日期字符串转成日期对象
start_date = datetime.strptime(start_str,'%Y-%m-%d')
```

简而言之，就是将一段给定的日期或时间字符串转换成一个对象输出。这个函数多见于一些python函数库里面底层封装了一个函数。方便直接调用。

如下示例代码，就是datetime内部封装好的一段可以供直接调用的方法：

```
@classmethod
def strptime(cls, date_string, format):
    'string, format -> new datetime parsed from a string (like time.strptime()).'
    import _strptime
    return _strptime._strptime_datetime(cls, date_string, format)
```

现在小伙伴们清楚了没有？strftime是转换为特定格式输出，而strptime是将一个（时间）字符串解析为时间的一个类型对象。一个是按照想要的格式，去转换。重点是格式！另外一个不管什么格式，我只要把特定的时间字符串转成时间类型即可！虽然只是一个字母之差，但是意义和侧重点完全不一样啊！

==========

- bubble_sort

https://www.adamsmith.haus/python/answers/how-to-bubble-sort-in-python

```
def bubble_sort(data):
    for _ in range(len(data)):
        for i in range(len(data) - 1):

            if data[i] > data[i + 1]:


                data[i], data[i + 1] = data[i + 1], data[i]
    return data

a_list = [2, 1, 5, 4, 3]

sorted_list = bubble_sort(a_list)

print(sorted_list)
```

OUTPUT

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

==========

- 負責偵錯的"Python 的 logging"，把logging 插在每個程式裡就能知道哪個程式沒跑

  - Python 的 logging

https://editor.leonh.space/2022/python-log/

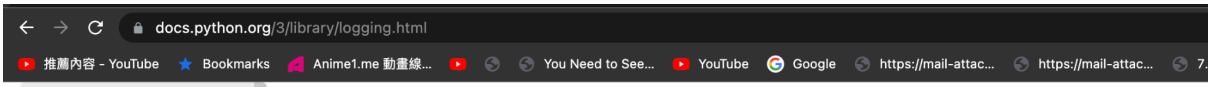
- Python Logging 软件运行日志.【老王编程】Python Series 系列讲座 14
  https://www.youtube.com/watch?v=mQWsByMAGUo&list=WL&index=1&ab_channel=JohnQWang

- Python Tutorial: Logging Basics - Logging to Files, Setting Levels, and Formatting
  https://www.youtube.com/watch?v=-ARI4Cz-awo&list=WL&index=2&ab_channel=CoreySchafer

- Python logging 日誌用法與範例
  https://shengyu7697.github.io/python-logging/

LogRecord attributes¶
https://docs.python.org/3/library/logging.html

Python Tutorial: Logging Basics - Logging to Files, Setting Levels, and Formatting
https://www.youtube.com/watch?v=-ARI4Cz-awo&ab_channel=CoreySchafer



LogRecord attributes

The LogRecord has a number of attributes, most of which are derived from the parameters to the constructor. (Note that the names do not always correspond exactly between the LogRecord constructor parameters and the LogRecord attributes.) These attributes can be used to merge data from the record into the format string. The following table lists (in alphabetical order) the attribute names, their meanings and the corresponding placeholder in a %-style format string.

If you are using {}-formatting (`str.format()`), you can use `{attrname}` as the placeholder in the format string. If you are using $-formatting (`string.Template`), use the form `${attrname}`. In both cases, of course, replace `attrname` with the actual attribute name you want to use.

In the case of {}-formatting, you can specify formatting flags by placing them after the attribute name, separated from it with a colon. For example: a placeholder of `{msecs:03d}` would format a millisecond value of `4` as `004`. Refer to the `str.format()` documentation for full details on the options available to you.

Attribute name	Format	Description
args	You shouldn't need to format this yourself.	The tuple of arguments merged into `msg` to produce `message`, or a dict whose values are used for the merge (when there is only one argument, and it is a dictionary).
asctime	`%(asctime)s`	Human-readable time when the `LogRecord` was created. By default this is of the form '2003-07-08 16:49:45,896' (the numbers after the comma are millisecond portion of the time).
created	`%(created)f`	Time when the `LogRecord` was created (as returned by `time.time()`).
exc_info	You shouldn't need to format this yourself.	Exception tuple (à la `sys.exc_info`) or, if no exception has occurred, `None`.

Attribute name	Format	Description
args	You shouldn't need to format this yourself.	The tuple of arguments merged into `msg` to produce `message`, or a dict whose values are used for the merge (when there is only one argument, and it is a dictionary).
asctime	`%(asctime)s`	Human-readable time when the `LogRecord` was created. By default this is of the form '2003-07-08 16:49:45,896' (the numbers after the comma are millisecond portion of the time).
created	`%(created)f`	Time when the `LogRecord` was created (as returned by `time.time()`).
exc_info	You shouldn't need to format this yourself.	Exception tuple (à la `sys.exc_info`) or, if no exception has occurred, `None`.
filename	`%(filename)s`	Filename portion of `pathname`.
funcName	`%(funcName)s`	Name of function containing the logging call.
levelname	`%(levelname)s`	Text logging level for the message (`'DEBUG'`, `'INFO'`, `'WARNING'`, `'ERROR'`, `'CRITICAL'`).
levelno	`%(levelno)s`	Numeric logging level for the message (`DEBUG`, `INFO`, `WARNING`, `ERROR`, `CRITICAL`).
lineno	`%(lineno)d`	Source line number where the logging call was issued (if available).
message	`%(message)s`	The logged message, computed as `msg % args`. This is set when `Formatter.format()` is invoked.
module	`%(module)s`	Module (name portion of `filename`).
msecs	`%(msecs)d`	Millisecond portion of the time when the `LogRecord` was created.
msg	You shouldn't need to format this yourself.	The format string passed in the original logging call. Merged with `args` to produce `message`, or an arbitrary object (see Using arbitrary objects as messages).
name	`%(name)s`	Name of the logger used to log the call.
pathname	`%(pathname)s`	Full pathname of the source file where the logging call was issued (if available).

==========

- 虛擬碼 (Pseudocode), 解構, 寫程式之前理解達到目的需要甚麼步驟及流程, 方便簡化原問題
  - 你知道什麼是虛擬碼(pseudo code)嗎？

- https://www.ro-boy.com/blog/what-is-pesudo-code

- [week 2] 先別急著寫 leetcode - 虛擬碼、Debugger、解題技巧
  - https://hackmd.io/@Heidi-Liu/note-alg101

近年來已有許多人，以保留C語言流程控制語法架構，再將填充於架構內的程式碼以文字說明或數學方程式取代，織組成描述演算法的說明文件，這種表達方式，被稱為是**虛擬碼 (pseudo code)**。

前述的客服電話處理流程圖範例，如果使用虛擬碼，可以寫成以下範例：

接聽電話 Answer phone

switch(請問有什麼事可幫忙How can we help)

case產品資訊product info:

請教並記錄公司寶號take name of company

轉接銷售部門 transfer to sales (ext 2203)

break

case客訴問題problem ask:

switch (請問是那種客訴what is problem)

case: 產品客訴 problem with product

請教並記錄公司寶號take name of company

轉接維修櫃台 transfer to help desk (ext 2217)

break

```
        case送貨客訴: shipping

            :

        case 帳單客訴: billing

            :

    endswitch(客訴種類)

        case 其他:

            :

        endswitch(何事)
```

由於虛擬碼與程式碼具有很強的對應性，而且佔用空間也遠小於流程圖，本課程未來介紹演算法，多半會利用**虛擬碼**作為工具。

[week 2] 先別急著寫 leetcode - 虛擬碼、Debugger、解題技巧

> 本篇為 [ALG101] 先別急著寫 leetcode 這門課程的學習筆記。如有錯誤歡迎指正。 程式解題新手入門注意事項 - Huli

寫程式≠寫程式碼

重點是在寫程式之前，先想好要怎麼做，整理自己的想法，並轉換成程式碼表達。

- 初學者寫程式 1.邊寫程式碼邊想怎麼解 2.試著套用自己以為學過的語法
- 會寫程式的人寫程式 1.先想解法，在腦中構思（太快看不出來） 2.把解法轉換成程式碼

初期如何學習寫程式？

- 步驟

1. 大概想一下解法，不寫程式碼
2. 把解法寫成 pseudo code（虛擬碼）
3. 把 pseudo code 翻譯成程式碼

- 利用條列式寫法

1. 將大問題分割成小問題
2. 一行只做一件事
3. 善用敘述、條件判斷
4. 善用跳轉（jump）來實現重複執行

範例：印出 1~100 之間的偶數

- 步驟一：如何印出 1~100？

1. 令 i 為 1　//　設置初始條件
2. 如果 i>100，結束　//　判斷是否該結束程式（終止條件）
3. 印出 i
4. i = i+1
5. 跳回第 2 行　//　重複動作

- 步驟二：如何判斷某個數是偶數？

1. 令 i = 1
2. 如果 i>100，結束
3. 如果 i % 2 ===0，印出 i　//　利用是否能整除2來判斷奇偶
4. i = i+1
5. 跳回第 2 行

==========

- **標題**
    - 01
        - 02

        格式與抓取資料

        https://www.youtube.co

==========

- **標題**
    - 01
        - 02

        格式與抓取資料

        https://www.youtube.co

==========

- 標題

    - 01
        - 02

          格式與抓取資料

          https://www.youtube.co

==========

- 標題

    - 01
        - 02

          格式與抓取資料

          https://www.youtube.co

==========

- 標題

    - 01
        - 02

          格式與抓取資料

          https://www.youtube.co

==========

- 標題
    - 01
        - 02

          格式與抓取資料

          https://www.youtube.co

==========

- 標題
    - 01
        - 02

          格式與抓取資料

          https://www.youtube.co

==========

- 標題
    - 01
        - 02

          格式與抓取資料

          https://www.youtube.co

==========

- 標題
    - 01

- 02
  格式與抓取資料
  https://www.youtube.co

==========

- 標題
  - 01
    - 02
      格式與抓取資料
      https://www.youtube.co

==========

- 標題
  - 01
    - 02
      格式與抓取資料
      https://www.youtube.co

==========

- 標題
  - 01
    - 02
      格式與抓取資料
      https://www.youtube.co

==========

- 標題

    - 01
        - 02

          格式與抓取資料

          https://www.youtube.co

==========

- 標題

    - 01
        - 02

          格式與抓取資料

          https://www.youtube.co

==========

- 標題

    - 01
        - 02

          格式與抓取資料

          https://www.youtube.co

==========

- 標題

    - 01
        - 02

        格式與抓取資料

        https://www.youtube.co

==========

- 標題

    - 01
        - 02

        格式與抓取資料

        https://www.youtube.co

==========

- 標題

    - 01
        - 02

        格式與抓取資料

        https://www.youtube.co