科學計算程式設計實習

Python 資料永續化

王昱景 Brian Wang
brian.wang.frontline@gmail.com

永續化機制

- 除了利用基本 I/O 來保存運算結果之外，Python 中還提供了一些方式，可以直接保存物件狀態，在下次重新執行程式時讀取以恢復運算時必要的資料，在這邊要介紹幾個方式，像是

- 物件序列化（Serialization）— 如透過 marshal、pickle、cPickle 模組

- DBM（Database Manager）— 簡單的“資料庫" 介面。DBM 物件行為上像個字典（Dictionary）物件，不過鍵（Key）值（Value）型態都必須是字串
- shelve 模組 — 一個 shelve 物件是個像字典的永續物件，不過值可以是 pickle 模組可處理的 Python 物件

- DB-API 2.0（PEP 249）— 存取資料庫的標準介面
- 除此之外還可以透過第三方程式庫，進行物件關聯對應（Object-Relational Mapping），像是 SQLAlchemy、SQLObject

marshal、pickle 與 cPickle

- 在物件序列化方面，marshal 是個很基礎的模組，其存在主要是為了支援 Python 的 .pyc 檔案
- 一般來說，如果要序列化 Python 物件，使用 pickle 模組會是比較好的方式

- pickle 會記錄已經序列化的物件，如果後續有物件參考到相同物件，才不會再度被序列化
- pickle 可以序列化使用者自定義的類別及實例，在格式方面，pickle 格證向後相容於新的 Python 版本
- cPickle 模組則是用 C 實作的模組，介面上與 pickle 相同，速度在理想上可達 pickle 的 1000 倍

pickle

```
class DVD:
    def __init__(self, title, year=None,
        duration=None, director_id=None):
        self.title = title
        self.year = year
        self.duration = duration
        self.director_id = director_id
        self.filename = self.title.replace(' ', '_') + '.pkl'

    def check_filename(self, filename):
        if filename is not None:
            self.filename = filename
```

- 這個 DVD 物件有 title、year、duration、director_id 四個狀態
- 每個 DVD 物件會以 title 作主檔名，加上 .pkl 副檔名進行儲存
- 接下來列出儲存物件的 save 方法

```
def save(self, filename=None):
    self.check_filename(filename)
    fh = None
    try:
        data = (self.title, self.year,
                self.duration, self.director_id)
        fh = open(self.filename, 'wb')
        pickle.dump(data, fh)
    except (EnvironmentError, pickle.PicklingError) as err:
        raise SaveError(str(err))
    finally:
        if fh is not None:
            fh.close()
```

- 最主要地，要以 'wb' 模式開啟檔案，然後使用 pickle.dump 進行物件序列化
- 接下來列出載入檔案 load 方法定義：

```
def load(self, filename=None):
    self.check_filename(filename)
    fh = None
    try:
        fh = open(self.filename, 'rb')
        data = pickle.load(fh)
    (self.title, self.year,
     self.duration, self.director_id) = data
except (EnvironmentError, pickle.PicklingError) as err:
    raise LoadError(str(err))
finally:
    ...
```

- 這次是讀取，因此你要用 'rb' 模式開啟檔案，然後使用 pickle.load 載入檔案。這個 DVD 物件可以這麼使用

```
filename = 'PyConTutorial2013.pkl'
dvd1 = DVD('PyCon Tutorial', 2013, 1, 'Justin Lin')
dvd1.save()
```

```
dvd2 = DVD('PyCon Tutorial')
dvd2.load()
print dvd2
```

DBM

- dbm 為柏克萊大學發展的檔案型資料庫
- Python 的 dbm 模組提供了對 Unix 程式庫的介面
- dbm 物件就像個字典，在不需要關聯式資料庫，只需要快速存取鍵值的場合可以使用，dbm 物件的鍵值都必須是字串

- Python 提供 DBM 的多數實現，如果不確定要用哪一種，可以使用 anydbm 模組，它會檢查並選擇系統上可用的 DBM 實作

```
import anydbm

# Open database, creating it if necessary.
db = anydbm.open('cache', 'c')

# Record some values
db['www.python.org'] = 'Python Website'
db['www.cnn.com'] = 'Cable News Network'

# Loop through contents. Other dictionary methods
# such as .keys(), .values() also work.
for k, v in db.iteritems():
    print k, '\t', v

# Storing a non-string key or value will raise an exception (most
# likely a TypeError).
db['www.yahoo.com'] = 4

# Close when done.
db.close()
```

shelve 模組

- shelve 物件也是個行為上像是字典的物件
- 與 DBM 差別在於值的部份可以是 pickle 模組可處理的 Python 物件
- 以下來看個實例，搭配 DAO 模式 來使用 shelve 模組的功能：

```
class DvdDao:
    def __init__(self, shelve_name):
        self.shelve_name = shelve_name

    def save(self, dvd):
        shelve_db = None
        try:
            shelve_db = shelve.open(self.shelve_name)
            shelve_db[dvd.title] = (dvd.year,
                dvd.duration, dvd.director_id)
            shelve_db.sync()
        finally:
            if shelve_db is not None:
                shelve_db.close()
```

- save 方法中，主要是使用 shelve.open 來開啟永續化時的字典檔案
- 在指定鍵值之後，使用 sync 方法將資料從快取中寫回檔案
- 接下來列出的 DAO 方法實作也是類似的操作：

```
def all(self):
    shelve_db = None
    try:
        shelve_db = shelve.open(self.shelve_name)
        return [DVD(title, *shelve_db[title])
                for title in sorted(shelve_db, key=str.lower)]
    finally:
        if shelve_db is not None:
            shelve_db.close()
    return []

def load(self, title):
    shelve_db = None
    try:
        shelve_db = shelve.open(self.shelve_name)
        if title in shelve_db:
            return DVD(title, *shelve_db[title])
    finally:
        if shelve_db is not None:
            shelve_db.close()
    return None

def remove(self, title):
    shelve_db = None
    try:
        shelve_db = shelve.open(self.shelve_name)
        del shelve_db[title]
        shelve_db.sync()
    finally:
        if shelve_db is not None:
            shelve_db.close()
```

- 以下是個使用 DvdDao 的例子：

```
filename = 'dvd_library.slv'
dao = DvdDao(filename)
dvd1 = DVD('PyCon Tutorial 2012', 2012, 1, 'Justin Lin')
dvd2 = DVD('PyCon Tutorial 2013', 2013, 1, 'Justin Lin')
dao.save(dvd1)
dao.save(dvd2)
print dao.all()
print dao.load('PyCon Tutorial 2012')
dao.remove('PyCon Tutorial 2013')
print dao.all()
```

資料庫

- 資料庫（Database, DB），簡單來說可視為電子化的檔案櫃
- 儲存電子檔案的處所，使用者可以對檔案中的資料執行新增、擷取、更新、刪除等操作
- 資料庫指的是以一定方式儲存在一起、能為多個用戶共享、具有儘可能小的冗餘度、與應用程式彼此獨立的資料集合

關聯式資料庫

- 關聯式資料庫（Relational database），是建立在關聯模型基礎上的資料庫，藉助於集合代數等數學概念和方法來處理資料庫中的資料
- 現實世界中的各種實體以及實體之間的各種聯繫均用關聯模型來表示

- 標準資料查詢語言 SQL 就是一種基於關聯式資料庫的語言，這種語言執行對關聯式資料庫中資料的檢索和操作

SQL

- 結構化查詢語言（Structured Query Language），是一種特殊目的之程式語言
- 用於資料庫中的標準資料查詢語言
- IBM 公司最早使用在其開發的資料庫系統中

- SQL 包含四個部分：
  - 資料定義語言
  - 資料操縱語言
  - 資料控制語言
  - 交易控制語言

資料定義語言

- 資料定義語言（Data Definition Language, DDL）是 SQL 語言集中，負責資料結構定義與資料庫物件定義的語言
- 由 CREATE、ALTER 與 DROP 三個語法所組成

- CREATE 是負責資料庫物件的建立，舉凡資料庫、資料表、資料庫索引、預存程式、使用者函式、觸發程式或是使用者自定型別等物件，都可以使用 CREATE 指令來建立

- ALTER 是負責資料庫物件修改的指令

- DROP 則是刪除資料庫物件的指令，並且只需要指定要刪除的資料庫物件名稱即可

資料操縱語言

- 資料操縱語言（Data Manipulation Language, DML）是SQL語言中，負責對資料庫物件執行資料存取工作的指令集
- 以 INSERT、UPDATE、DELETE 三種指令為核心，分別代表插入、更新與刪除

- 是開發以資料為中心的應用程式必定會使用到的指令
- SQL 的 SELECT 語句的四大指令以「CRUD」來稱呼
- DML 的主要功能即是存取資料，因此其語法都是以讀取與寫入資料庫為主
- 除了 INSERT 以外，其他指令都可能需搭配 WHERE 指令來過濾資料範圍，或是不加 WHERE 指令來存取全部的資料

- INSERT 是將資料插入到資料庫物件中的指令，可以插入資料的資料庫物件有資料表以及可更新檢視表兩種
- UPDATE 指令是依給定條件，將符合條件的資料表中的資料更新為新的數值
- DELETE 指令為自資料庫物件中刪除資料的指令

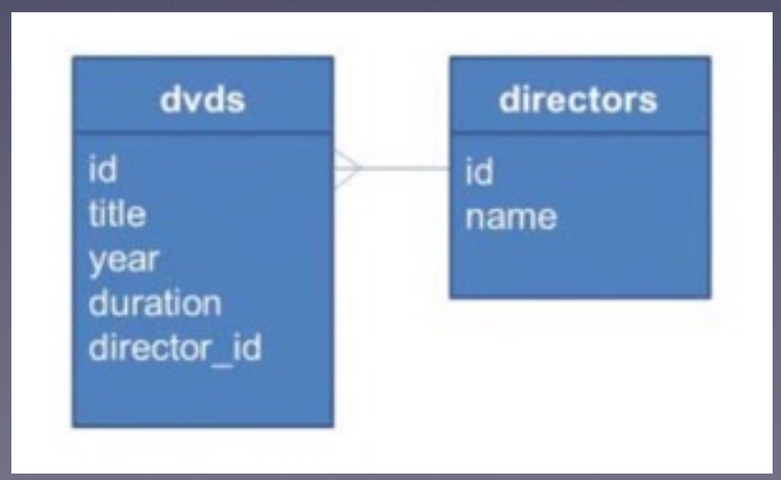
SQLite

- SQLite 是遵守 ACID 的關聯式資料庫管理系統，它包含在一個相對小的 C 程式庫中
- 不像常見的客戶端/伺服器結構範例，SQLite 引擎不是個程式與之通訊的獨立行程，而是連結到程式中成為它的一個主要部分

- 所以主要的通訊協定是在程式語言內的直接 API 呼叫
- 整個資料庫（定義、表、索引和資料本身）都儲存在主機端上單一個檔案中
- 此種簡潔的設計是透過寫入時鎖定整個資料檔案而完成的
- 提供了一個叫做 sqlite3 的獨立程式用來查詢和管理 SQLite 資料庫檔案

DB-API 2.0（PEP 249）

- 為 Python 中存取資料庫的標準介面
- Python 中的 sqlite3 模組，提供了 DB-API 2.0 的實作，可用以存取 SQLite 資料庫
- 接下來的範例，會存取的資料庫表格如下：



```
def connect(name):
    create = not os.path.exists(name)
    conn = sqlite3.connect(name)
    if create:
        cursor = conn.cursor()
        cursor.execute("CREATE TABLE directors ("
            "id INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE NOT NULL, "
            "name TEXT UNIQUE NOT NULL)")
        cursor.execute("CREATE TABLE dvds ("
            "id INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE NOT NULL, "
            "title TEXT NOT NULL, "
            "year INTEGER NOT NULL, "
            "duration INTEGER NOT NULL, "
            "director_id INTEGER NOT NULL, "
            "FOREIGN KEY (director_id) REFERENCES directors)")
        conn.commit()

    return conn

def add_dvd(conn, title, year, duration, director):
    director_id = get_and_set_director(conn, director)
    cursor = conn.cursor()
    cursor.execute("INSERT INTO dvds "
                   "(title, year, duration, director_id) "
                   "VALUES (?, ?, ?, ?)",
                   (title, year, duration, director_id))
    conn.commit()

def get_and_set_director(conn, director):
    director_id = get_director_id(conn, director)
    if director_id is not None:
        return director_id
    cursor = conn.cursor()
    cursor.execute("INSERT INTO directors (name) VALUES (?)",
                   (director,))
```

```
    conn.commit()
    return get_director_id(conn, director)

def get_director_id(conn, director):
    cursor = conn.cursor()
    cursor.execute("SELECT id FROM directors WHERE name=?",
                   (director,))
    fields = cursor.fetchone()
    return fields[0] if fields is not None else None

def all_dvds(conn):
    cursor = conn.cursor()
    sql = ("SELECT dvds.title, dvds.year, dvds.duration, "
           "directors.name FROM dvds, directors "
           "WHERE dvds.director_id = directors.id"
           " ORDER BY dvds.title")
    cursor.execute(sql)
    return [(str(fields[0]), fields[1], fields[2], str(fields[3]))
            for fields in cursor]

def all_directors(conn):
    cursor = conn.cursor()
    cursor.execute("SELECT name FROM directors ORDER BY name")
    return [str(fields[0]) for fields in cursor]
```

- 以下是個存取資料庫的例子：

```
db_name = 'dvd_library.sqlite3'
conn = connect(db_name)
add_dvd(conn, 'Python Tutorial 2013', 2013, 1, 'Justin')
print all_directors(conn)
print all_dvds(conn)
```