

$$(X, O) = e^{-\frac{x^2}{2\sigma^2}}$$

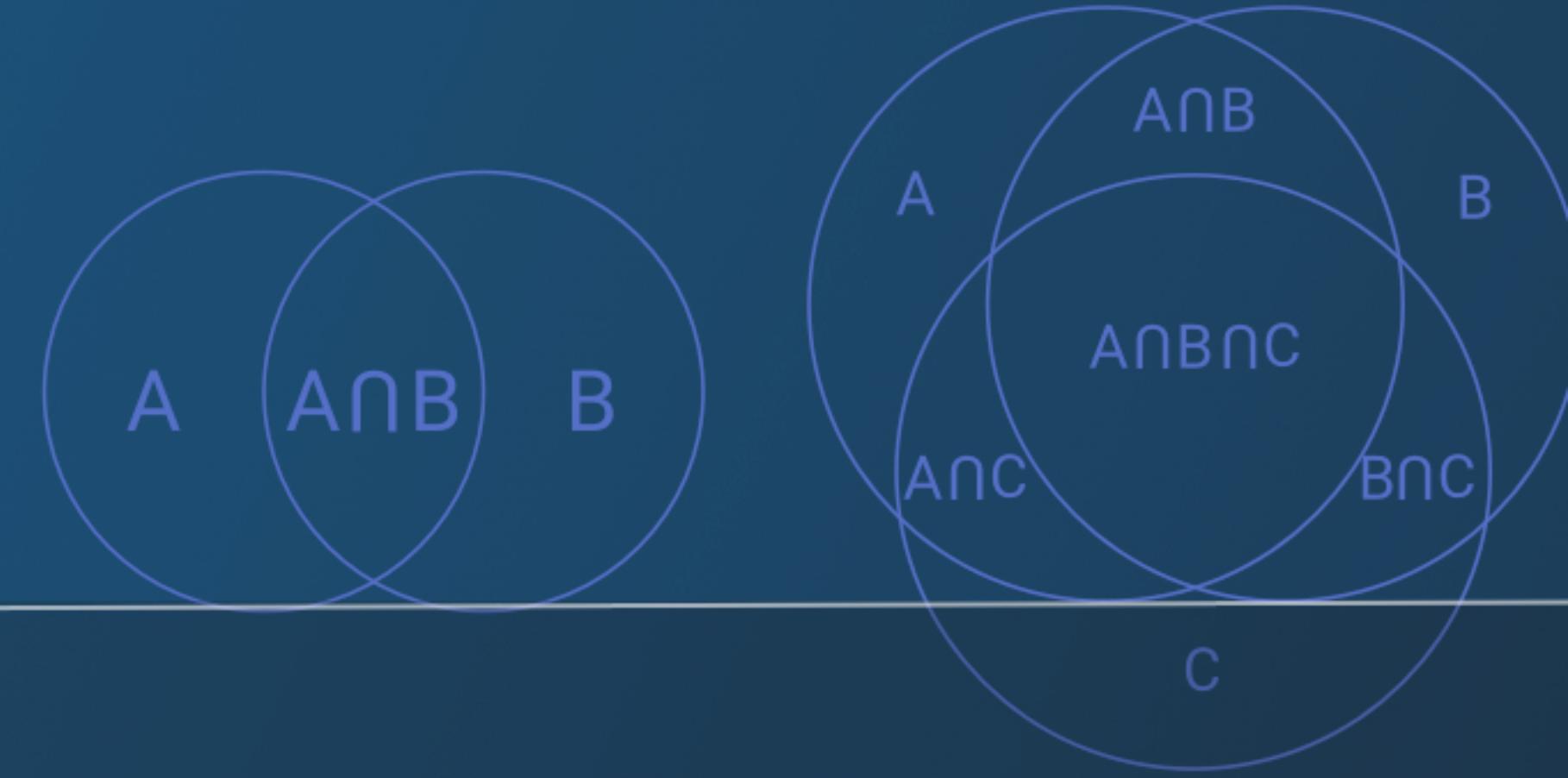
$$x(X, O) = -\frac{x}{\sigma^2} G(X, O) = -\frac{x}{\sigma^2} e^{-\frac{x^2}{2\sigma^2}}$$

$$xx(X, O) = \frac{x^2 - \sigma^2}{\sigma^4} G(X, O) = \frac{x^2 - \sigma^2}{\sigma^4} e^{-\frac{x^2}{2\sigma^2}}$$

$$xxx(X, O) = -\frac{x^3 - x\sigma^2}{\sigma^6} G(X, O) = -\frac{x^3 - x\sigma^2}{\sigma^6} e^{-\frac{x^2}{2\sigma^2}}$$

# Julia 程式語言學習馬拉松

## Day 14



$$\ln(x + \sqrt{1+x^2}) + x - \frac{1}{x + \sqrt{1+x^2}} \left( 1 + \frac{x}{\sqrt{1+x^2}} \right)$$



cupay 陪跑專家 : James Huang

# 檔案處理 與 資料庫連線





# 重要知識點



- 流 (Stream)、輸入輸出 (IO)、序列化 (serialization) 是處理資料的基礎，例如檔案、資料庫、網路流... 等。
- 在處理文字檔案的部分，今天將介紹最常用的讀取及寫入純文字檔、CSV、JSON 檔案。CSV 及 JSON 的處理將介紹 CSV.jl 與 JSON.jl 套件的基本使用。
- 資料庫的連接可以透過 ODBC 介面連接，也可以使用原生的資料庫套件，接下來將以 SQLite 資料庫示範。



# 流 (Stream) 與 輸入輸出 (IO)



- 各種 Julia 不同 stream 的上層型別均為 IO，也都繼承了 read() 和 write() 函式。
- 最常見的標準流為輸入 (stdin)、輸出 (stdout)、錯誤 (stderr)。
- 標準流可以透過下列函式查詢其物件的屬性：

函式名稱	功能	stdin	stdout	stderr
<code>isopen()</code>	是否開啟	TRUE	TRUE	TRUE
<code>isreadable()</code>	是否可讀	FALSE	FALSE	FALSE
<code>isreadonly()</code>	是否為唯讀	TRUE	FALSE	FALSE
<code>iswritable()</code>	是否可寫入	FALSE	TRUE	TRUE



# 文件檔案操作

- 要操作文件檔案時，需要的時候可以透過函式 `open()` 及 `close()` 進行開啟和關閉。
- 開啟檔案時，預設的模式是唯讀。若需要進行寫入或添加 (`append`) 的話，需要指定相對應的模式。

模式 (Mode)	操作說明
r	開啟並為唯讀。
r+	開啟可讀且可寫入。
w	檔案不存在時建立檔案、存在的話清空內容，可寫入。
w+	可讀且可寫入，檔案不存在時建立檔案、存在的話清空內容。
a	檔案不存在時建立檔案、存在的話添加內容，可寫入。
a+	可讀且可寫入，檔案不存在時建立檔案、存在的話添加內容。



# 緩衝區 (Buffer)



- 有時候因為傳輸及處理的需求，我們需要先將 stream 裡面的資料暫存，再進行後續的處理，這時候就可以使用 IOBuffer。
- IOBuffer 內容除了資料流本身之外，也包含了豐富的屬性來標示及做為操作 buffer 之用。使用 dump() 函式可以列出資料及所有屬性及其值。
- 透過 take!() 函式，可以取出 buffer 內容並重置 buffer。



# 序列化 (Serialization)



- 序列化可以將資料在不同的類型與 stream 之間互相轉換。例如，在資料傳輸之前先將字串 `serialize()`，在收到資料後進行反序列化 `deserialize()` 還原資料。
- 序列化時會自動先加入 8 byte 的 header 資訊。
- 序列化和反序列化時，可以針對 stream，也可以針對檔案，呼叫 `serialize()` 和 `deserialize()` 函式時可傳入 stream 或是檔案名稱。



# CSV 檔案的讀取與寫入



- CSV.jl 套件提供讀取與寫入 csv 檔案（符號分隔文字檔案）的功能。
- 讀取 `read()` 函式
  - 如果沒有 `header` 的話，可以設定 `header` 屬性為 `false`，或是指定 `header` 名稱。
  - `delim` 屬性可以指定分隔符號，若未指定的話，讀取時會自動判斷。
  - 讀取回傳預設類型為 `DataFrame`。
  - 配合 `ZipFile.jl` 或其他套件可直接從壓縮檔案中讀取 csv 檔案。
- 寫入 `write()` 函式
  - 在範例中我們會以 `DataFrame` 存入 csv 檔做為示範。



# JSON 檔案的讀取與寫入



- JSON.jl 套件提供讀取與寫入 JSON 檔案的功能。
- 呼叫 `parsefile()` 函式，將 JSON 檔案讀入，回傳類型為 Array。
- 儲存 JSON 檔案時，先透過 `json()` 函式將物件轉換為 JSON 字串，再存入文字檔案。





# 資料庫存取



- 連接資料庫，將介紹 2 種方式，大家可以依自己的需求自行選擇：
  - 使用 ODBC 介面。ODBC 提供一致的應用程式介面，透過 ODBC 與資料庫驅動程式的橋接程式，可以有一致的資料庫存取方式。範例以 ODBC.jl 在 Windows 10 作業系統做為示範，資料庫為 SQLite。
  - 使用原生的資料庫連線套件，直接連接資料庫。範例以 SQLite.jl 做為示範。
- 在今天的範例中，使用的資料庫為 SQLite，資料庫檔案與範例程式存放在同一個路徑下即可。
- ODBC 的設定方式，以及橋接程式的下載安裝，請參照範例程式。

# 知識點 回顧

- 流 (Stream)、輸入輸出 (IO)、序列化 (serialization) 是處理資料的基礎，在今天的內容中並搭配了不同的套件，協助我們進行不同格式檔案及資料庫的連接與資料處理。
- 透過 ODBC 統一介面，大家可以嘗試安裝不同的橋接及驅動程式，依自己的需求連接不同的資料庫。或是，也可以使用 Julia 原生的資料庫套件。
- Julia 官方也有 DelimitedFiles 模組，可依自己的喜好決定要使用 CSV.jl 或是 DelimitedFiles 模組。



## 推薦閱讀

- [devart ODBC Driver for SQLite](#) (免費試用下載)
- [devart ODBC Driver for SQLite](#) (設定說明)
- 不同的原生資料庫驅動程式，可以參考：

### [JuliaDatabases](#)

- [CSV.jl 官方文件](#)
- [JSON.jl 官方網站](#)



解題時間

請跳出 PDF 至官網 Sample Code  
& 作業開始解題