

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

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

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

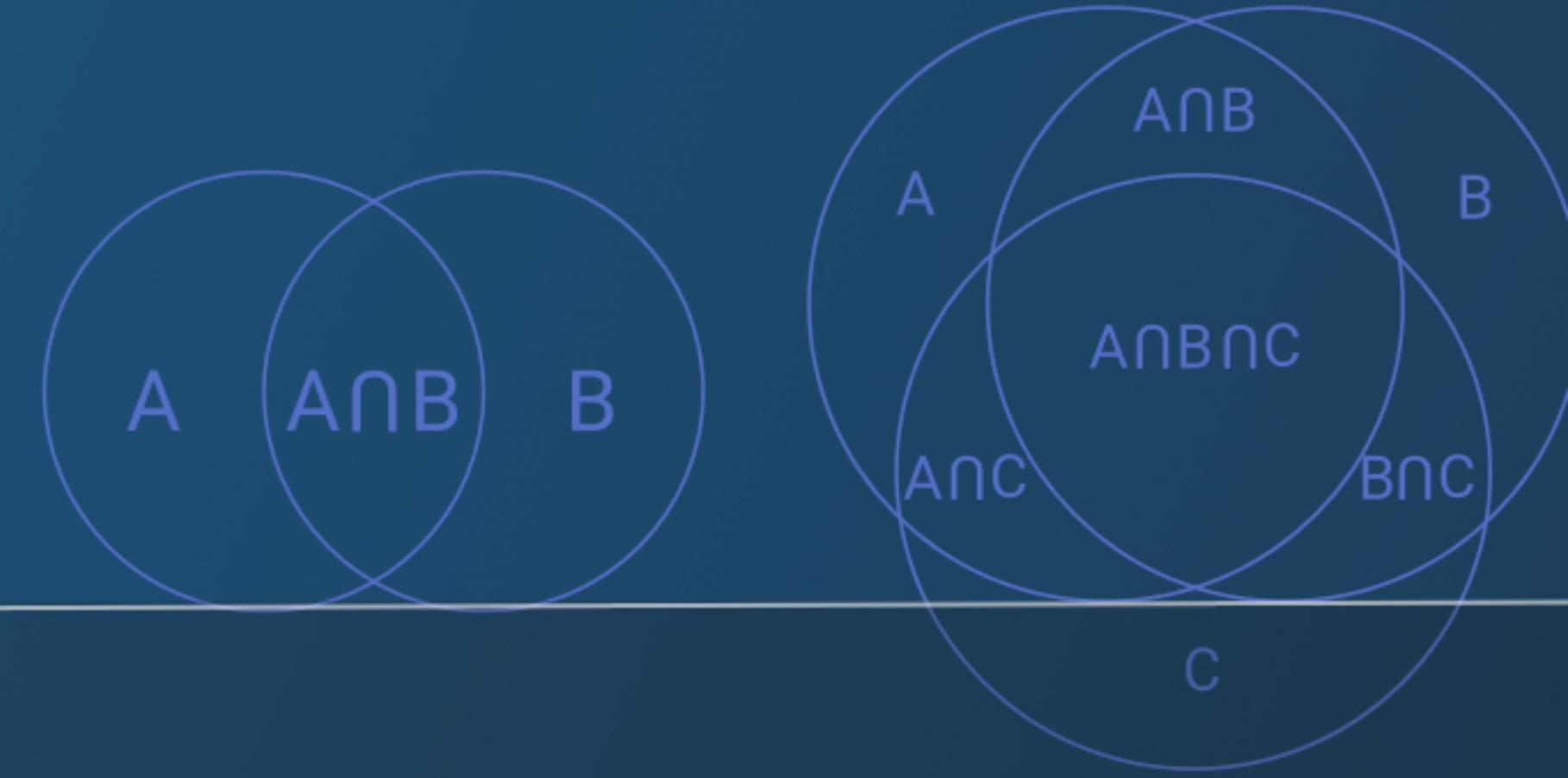
$$xxx(X, O) = -\frac{x^3 - x\sigma^2}{\sigma^6} G(X, O) = -\frac{x^3 - x\sigma^2}{\sigma^6} e^{-\frac{x^2}{2\sigma^2}}$$

Julia 程式語言學習馬拉松

Day 8



Cupay 陪跑專家 : James Huang



$$\begin{aligned}y'' &= \left[\ln(x + \sqrt{1 + x^2}) \right]' + \left(\frac{x}{\sqrt{1 + x^2}} \right)' \\&= \frac{1}{x + \sqrt{1 + x^2}} \left(\frac{\sqrt{1 + x^2} + x}{\sqrt{1 + x^2}} \right)' + \frac{1}{\sqrt{1 + x^2}} - \frac{x^2}{\sqrt{(1 + x^2)^3}} \\&= \frac{1}{\sqrt{1 + x^2}} + \frac{1}{\sqrt{1 + x^2}} - \frac{x^2}{\sqrt{(1 + x^2)^3}} \\&= \frac{2}{\sqrt{1 + x^2}} - \frac{x^2}{\sqrt{(1 + x^2)^3}} = \frac{2(1 + x^2) - x^2}{\sqrt{(1 + x^2)^3}} = \frac{2 + x^2}{\sqrt{(1 + x^2)^3}}\end{aligned}$$

字元與字串





重要知識點



- 字元和字串是應用程式開發和資料科學領域常需要處理的資料型態。Julia 對於字元和字串的支援，相較於較早的程式語言，由於原生就支援了 Unicode 編碼，所以可以很直接方便地進行處理。
- 在今天的內容中，將介紹字元和字串資料型別，以及其相關的操作。



字元



- 字元在 Julia 中是 32 位元長的原始型別，以單引號包含單一字元的方式表示。
- 對於超出 ASCII 碼之外的 Unicode 字元，也可以作為字元物件。
 - 不過，Unicode 字元的長度不一，在字串中使用 Unicode 字元時，要留意字元的處理，後續在字串的部分會以範例說明。



字元與字元變數及其型別

In [1]:

```
1 # 字元
2 'x'
```

Out[1]: 'x': ASCII/Unicode U+0078 (category Ll: Letter, lowercase)

In [2]:

```
1 # answer 為字元變數，使用 typeof 函數可以看到其型號為字元
2 answer = 'y'
3 typeof(answer)
```

Out[2]: Char

In [3]:

```
1 # 使用 \u 或 \U 加上 Unicode 編碼，可以顯示相對應的 Unicode 字元
2 '\u2460'
```

Out[3]: '①': Unicode U+2460 (category No: Number, other)

In [4]:

```
1 # 同樣的，我們可以看到 2460 這個 Unicode 字元型別為 Char
2 typeof('①')
```

Out[4]: Char



字元與數值的轉型



- 將數值轉型為字元：Char(<字元Unicode編碼>)
 - 使用 Char 轉型並不會判斷字元編碼是否為合法值，這時候可以用 `isValid()` 來判斷。合法的話，會傳回 `true` 值；如果編碼值為非法的話，則會傳回 `false` 值。
 - 字元編碼採用二/八/十/十六進位皆可。
- 將字元轉型為數值：`convert(<數值型別>, <字元>)`



字元的運算與比較



- 計算
 - 字元跟字元之間可用減法，計算兩個字元間編碼的距離。
 - 加、乘、除法均不合法。
 - 字元本身加減整數值，可以得到該字元往前或往後對應的字元。
- 比較：字元之間可以使用比較運算子進行比較



字串

- 字串是以成對雙引號或是成對的 3 個雙引號。
- 若是字串中要包含引號的話，可以用下列兩種方式：
 - 成對引號中，使用 \"，例如："Hello \"Julia\" world"
 - 成對三引號中，使用引號，例如：“””Hello ”Julia” world”””



字符串的索引

- 字串可以透過索引值，取得對應位置的字元或子字串，例如：
 - `str[i]`: 取得字串中第*i*個位置的字元。
 - `str[1:end-2]`: 取得字串第 1 個到倒數第 2 個字元的子字串。
- 字串的索引起始值是從 1 開始，在 Julia 語言中預設皆是如此，與其他大多程式語言有所不同。
- 使用索引值時，若遇到 UTF-8 編碼字元時，在計算上索引位置式需特別留意，在今天的範例中會有程式及詳細說明。



字符串的插值 (Interpolation)



- 在應用上，常會遇到需要在字符串中插入變數值，與字符串結合或輸出，這時候我們可以在字符串使用 \$，\$ 後接續變數名稱或是表達式 (expression)，就可以達到將字符串中的變數值或是表達式整合在一起。

```
In [1]: greet = "Welcome"  
whom = "John"  
  
println("$greet, $whom")
```

```
Welcome, John
```

```
In [2]: "1 + 2 = $(1 + 2)"
```

```
Out[2]: "1 + 2 = 3"
```



字符串常用操作 – 組合



- 字串與字串或字元之間的組合，可以透過下列幾種方式進行操作：
 - `string()` 函式：`string("abc", "123")`
 - * 運算子：`"abc" * "123"`
 - `Broadcast` 函式：`broadcast(*, "abc", "123")`
- 若要結合字串/字元元組 (tuple) 或是陣列 (array)，可以使用 `join()` 函式，
例如：`join(["abc" "123"])`
- 若要結合字元陣列，除了上述的 `join()` 函式之外，也可以使用 `String()` 建構子，
例如：`String(['a', 'b', 'c'])`



字串常用操作 – 比較



- 字串與字串之間可以用比較運算子進行比較，例如：“abc” < “xyz”
- 但是字串和字元之間不能互相比較大小，只能比較是否相等。例如：
 - “abc” < ‘a’，會產生 exception。
 - “abc” == ‘a’，則會回傳 False 值，表示兩者不相同。



字符串常用操作 – 搜尋



- 要搜尋 “world” 字串是否在 “hello world” 字串內，可使用 `occursin()` 函式，
例如：`occursin("world", "hello world")`
- 要搜尋字元是否在字串內，可使用 `in` 或是 `∈`，例如：`'w' in "hello world"` 或是 `'w' ∈ "hello world"`
- 除了搜尋字串或字元是否存在，使用 `findFirst()`, `findlast()`, `findprev()`, `findnext()` 函式，可以取得搜尋字串的位置，但是特別要留意的是若是要搜尋字元時，也要使用雙引號的 `AbstractString`，而不能使用字元型別，在範例中會有更詳細的示範。



字符串常用操作 – 取代



- 取代字符串中的內容，可以使用 `replace()` 函式，例如：`replace("hello world", "o" =>"p")` 會將字符串中的 o 取代成 p。
- 搭配使用 `count`，可以指定要取代的數目，例如：`replace("hello world", "o"=>"p"; count=1)` 僅會取代第一個 o。
- 由於字符串本身是 `immutable`，所以使用 `replace()` 函式不會變更原字符串，變更後的結果需要建立另一個字符串來存放。



字符串常用操作 – 分割



- 分割字串，例如要分割一個逗號分隔的字串時，可以使用 `split()` 函式：
 - `split("hello, world, John, Wick", ',')`
- 分割後的字串，型別為 `SubString`，可以利用 `String()` 建構子再轉型型別為 `String`。



字符串常用操作 – 與數值之間的轉型



- 要將數值轉型為字串，可以使用 `string()` 函式。例如：`string(10)`
 - 轉型時，加上 `pad` 參數可以補足位數，例如要對齊位數時：`string(110, base=10, pad=10)` 會在 110 之前補 7 個 0，產生總共 10 位數長的字串。
- 要將字串轉型為數值，可使用 `parse()` 函式，例如：`parse(Float64, "12.3")`

知識點回顧

- 今天的內容介紹字元和字串，以及字元及字串的常用操作，Julia 提供了許多內建函式，可以讓開發者方便地進行字元和字串的操作。
- 在字串的組合部分，與其他許多程式語言使用 ‘+’ 運算子來組合字串不同的是，Julia 使用 ‘*’，在官方文件裡認為 ‘+’ 運算子通常是符合交換律的 (commutative)，例如： $A + B == B + A$ ，但是字串的結合是 noncommutative，也就是說 $"hello" "world" != "world" "hello"$ ，因而 Julia 採用的是 ‘*’ 而非 ‘+’。



推薦閱讀

- [Strings 官方文件](#)
- [Unicode 字元編碼表](#)





解題時間

請跳出 PDF 至官網 Sample Code
& 作業開始解題