

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

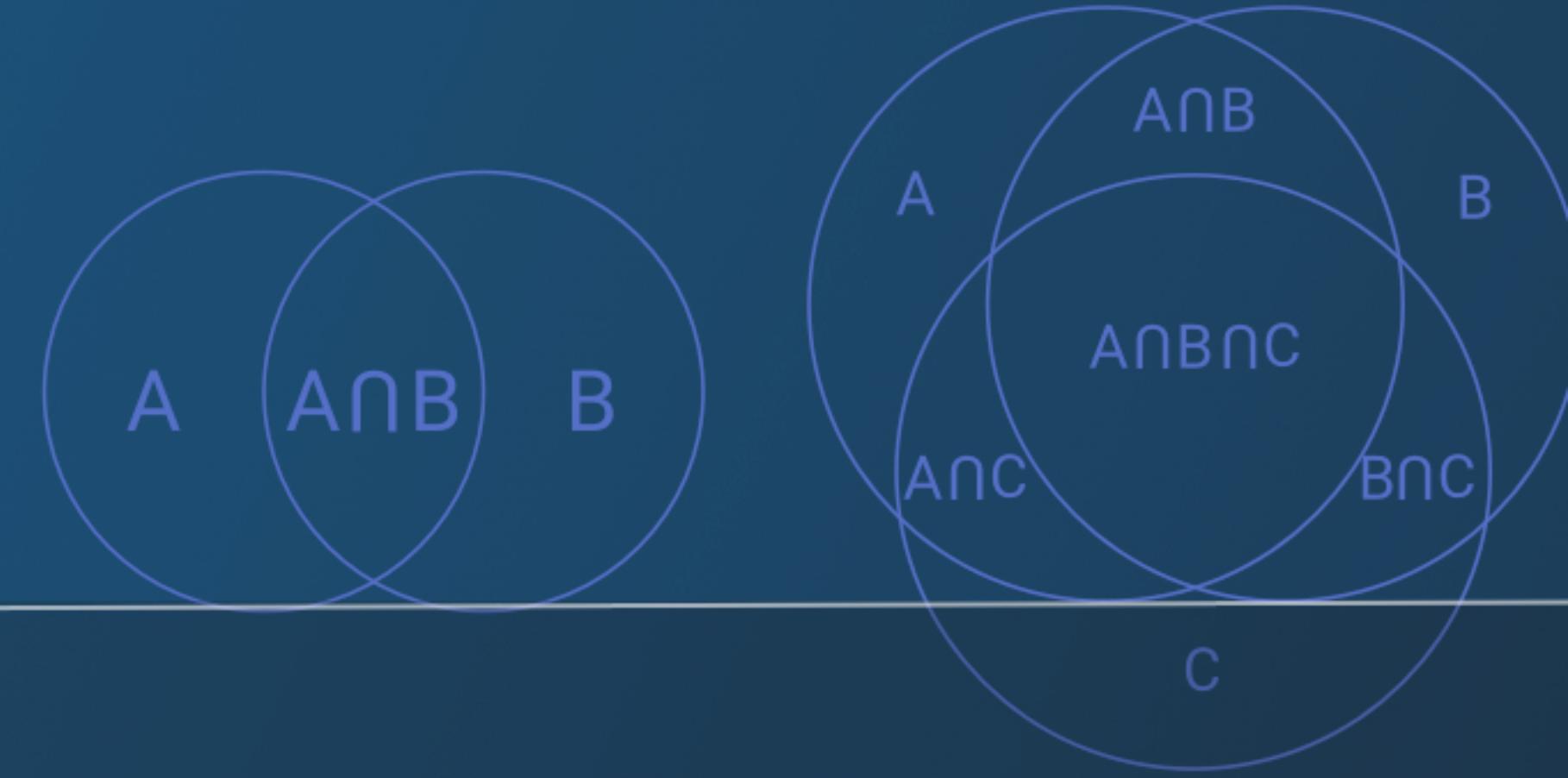
$$x(X, O) = -\frac{x}{\sigma^2} G(X, O) = -\frac{x}{\sigma^2} e^{-\frac{x^2}{2\sigma^2}}$$

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

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

# Julia 程式語言學習馬拉松

## Day 04



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



cupay 陪跑專家 : James Huang

# 數值系統





# 重要知識點



- 今天的內容中將介紹 Julia 的數值系統，並會在後續的內容中介紹運算，完整地讓大家了解如何進行數值的操作與運算。
- 在數值系統的介紹中，將介紹不同的數值型別，包含：整數 (Integer)、布林值 (Bool)、浮點數 (Floating Point)、有理數 (Rational)、複數 (Complex)、無理數 (Irrational)，以及如何隨機產生數值的常用函式。



# 整數 (Integer) - 宣告與使用



- 整數是數值系統中最基本的型別，分為有正負號(signed)與無正負號(unsigned)整數型別，其最大和最小值也不同。

signed / unsigned	型別	最大值	最小值
signed	Int128, Int64, Int32, Int16, Int8	$2^{\text{位元數}} - 1$	$-2^{\text{位元數}} - 1$
unsigned	UInt128, UInt64, UInt32, UInt16, UInt8	$2^{\text{位元數}} - 1$	0

- 另外，BigInt 型別是任意大小，BigInt 型別必須明確 (explicitly) 宣告，無法自動被向上轉型 (promote)。



# 整數 (Integer) - 宣告與使用 (續)



- 宣告時，Julia 會自動判斷適用的整數值型別，十進位的值會自動指定為有正負號的整數，預設的位數是以作業系統位元數為準，例如 64 位元的作業系統就會自動採用 Int64 型別。(註: 在不影響位元數的情況下，也就是未超過型別的最大值，且未指定型別。)
- 若是十六進位、八進位、二進位的整數值，則可以將 0x, 0o, 0b 放在數值之前，代表不同的進位制。須留意十六進位、八進位、二進位之英文字母均需小寫。
- 十進位的整數會自動判斷為有 signed 整數，但是十六/八/二進位則會自動判斷為 unsigned。



# 整數 (Integer) - 指定型別與轉換



- 型別的指定，只要在宣告時加上型別名稱即可。如果沒有指定的話，在不影響位元數的情況下，也就是未超過型別的最大值，型別會是 Int64 (64 位元作業系統)。
- 有無正負號的轉換，可以透過 `signed()` 或 `unsigned()` 函式來進行。要留意的是，`signed()` 轉換時，若造成轉換後溢位 (overflow)，系統並不會顯示錯誤。
- 相除取整數 (integer divide) 時，若是將整數  $\div$  零，會產生 `DivideError`。



# 布林值 (Bool) - 宣告、使用、轉換



- 布林值是用 `true` 或 `false` 保留字來代表邏輯判斷的“真”或“假”。
- 在型別系統上，`Bool` 是``Integer`` 的子型別。如果用 `bitstring()` 來查看的話，會發現 `true` / `false` 內部是以 `Int8` 型別表達。
- 跟有些程式語言不同的是，在 Julia 中邏輯判斷不能用 1 或 0 整數值來取代布林值 `true` / `false`，會造成 `TypeError` 錯誤。
- `Bool` 可以與整數型別進行互相轉換，但若嘗試將非 1 或 0 的整數值轉為 `Bool`，則會產生 `InexactError` 錯誤。



# 浮點數 (Floating Point) - 宣告、使用、轉換



- 浮點數沒有 signed / unsigned 的區分，而是分為不同的精度。
- 浮點數是實數的子型別，浮點數型別有 Float64、Float32、Float16，其邊界值為無窮大或負無窮大。無窮值 ( $\text{Inf}$  及  $-\text{Inf}$ ) 在 Julia 都是正常的浮點數。
- 科學記號的表示方式也可以用來宣告浮點數。
- 同樣的，預設的位數是以作業系統位元數為準，例如 64 位元的作業系統就會自動採用 Float64 型別。若加上 `f0` 字串，則是宣告為 Float32。
- NaN (not a number) 也是屬於特殊浮點數，有 NaN, NaN32, NaN16 型別。



# 浮點數(Floating Point) - 宣告、使用、轉換 (續)



- 浮點數零比較特別，正零與負零均為零。但是實際上正零與負零的位元內容不相同。請參考範例程式示範。
- 若要指定型別，只要在宣告時加上型別名稱即可。例如：Float32(1.0)。
- 計算 epsilon 可使用內建的 `eps()` 函式。
- 須特別留意在 Rounding 的方法部分，Julia 是採用 IEEE 754 規範，也就是 `RoundNearest`，取最近的偶數(四捨五取最近偶數)，而非我們一般講的四捨五入。要採用不同的 `rounding` 方法，可在呼叫 `round()` 函式時加入不同的常數參數，



# 有理數 (Rational) - 宣告、使用、轉換



- 有理數是用 "分子//分母" 的格式宣告，有規則如下：
  - 分子、分母均需為整數
  - 分子、分母的整數型別不同時，會自動進行必要的轉換
  - 分子、分母可為正或負數，但是有理數會被約分為分母不為負數的形式
  - 分子及分母不可為浮點數
  - 分母或分子可為零，但是不可分子和分母同時為零
- 有理數與浮點數之間可以進行轉換。
- 呼叫 `numerator()` 與 `denominator()` 函式分別可以取得有理數的分子或分母值。



# 複數 (Complex) - 宣告、使用、轉換



- 複數是由實部和虛部組成，格式為 "實部+虛部 $\text{i}m$ "。實部和虛部可以是任何實數 (Real) 型別，包含整數、浮點數、有理數、無理數。若未提供實部或虛部值，則預設為 0。
- 複數可以使用建構子 `Complex()` 或函式 `complex()` 來宣告。
- 若是使用變數來組成複數的實部和虛部，則在虛部的部分需要加上 \*。
- 實部及虛部為浮點數時，而且要指定型別時，可以使用 `Complex{Float64}`、`Complex{Float32}`、`Complex{Float16}`宣告。



# 複數 (Complex) - 常用函式



- 複數常用函式的使用，請參照範例程式。

函式	說明
<code>real()</code>	取得實部
<code>imag()</code>	取得虛部
<code>reim()</code>	同時取得實部與虛部
<code>conj()</code>	共軛
<code>abs()</code>	絕對值；與0的距離
<code>abs2()</code>	絕對值平方
<code>angle()</code>	相角



# 亂數

- 隨機產生數字是常會需要使用到的功能，亂數的型別預設是 Float64。
- 常用的亂數產生函式如下表，在範例程式中可以參考使用方法。其中 `rand()` 與 `randn()` 是屬於 `Base` 模組，而 `randexp()` 是 `Random` 模組，在呼叫前需先 `using` 或 `import` `Random`。

函式	說明
<code>rand()</code>	產生元素為隨機介於 $[0, 1)$ 區間的均勻分布數字
<code>randn()</code>	產生元素為隨機常態分布的數字
<code>randexp()</code>	產生元素為隨機指數分布的數字

# 知識點黑點 回顧

- 今天介紹了 Number 型別下所有的數值型別，及其使用與轉換應用。另外，在官方文件裡較少提到的無理數 (Irrational)，大家可以參考範例程式。
- 比較特別是 Rounding，預設的方法不是我們傳統認知的四捨五入，這點與 Python 相同，在應用時須留意。
- 了解數值系統之後，接下來我們介紹運算子及運算，與數值搭配可以讓我們進行各種運算應用。



## 推薦閱讀

- [Julia 官方文件 : Integers and Floating-Point Numbers](#)





解題時間

請跳出 PDF 至官網 Sample Code  
& 作業開始解題