

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

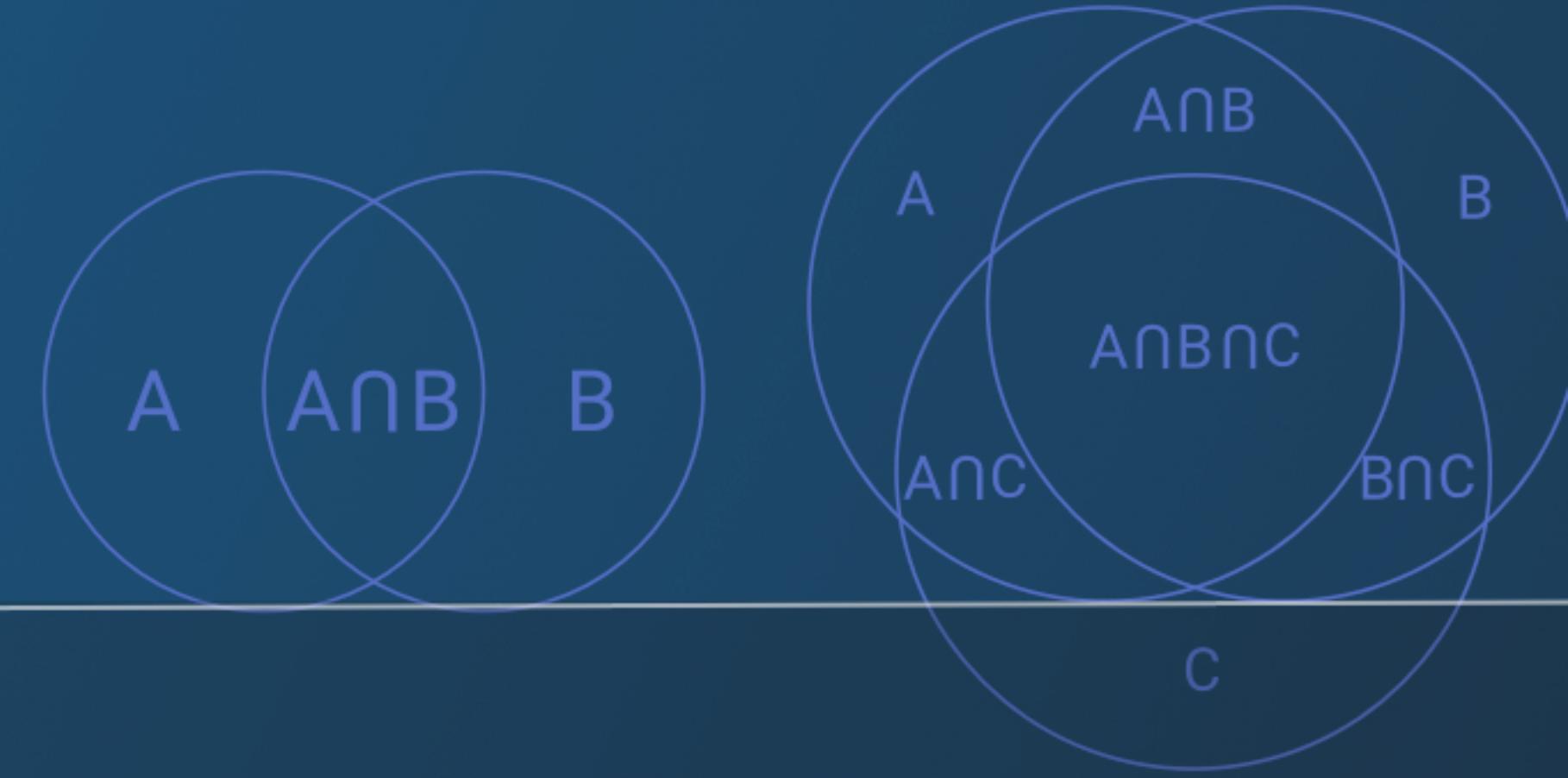
$$x(X, O) = -\frac{x}{\sigma^2} G(X, O) = -\frac{x}{\sigma^2} e^{-\frac{x^2}{2\sigma^2}}$$

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

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

# Julia 程式語言學習馬拉松

## Day 16



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



cupay 陪跑專家 : James Huang

# Multiple Dispatch





# 重要知識點



- Multiple Dispatch (多重分派) 在執行期間 (runtime) 根據傳入函式的引數型別，去判斷其”行為”，也就是應該調用哪一個 Method。
- Julia 內建的函式均已使用多重分派實作。



# Multiple Dispatch (多重分派)



- 多重分派的觸發時機是在執行期間 (runtime)，呼叫函式時根據傳入引數 (argument) 的數目及類型，判斷應該調用的方法 (method)。
- 之所以稱之為 "multiple" dispatch 的原因是，Julia 判斷時是根據所有引數，而非像在某些程式語言中是根據第一個引數做判斷。
- 與多重分派相似的程式語言概念是 Function Overloading，最大的不同點在於，Function Overloading 的觸發時機點在編譯 (compile time) 期間。



# Methods (方法)

- Method 是 Function (函式) 的行為 (behavior) 定義，一個函式可以被定義多個行為，也就是多個 Method，在 Method 中再去進行不同的實作。
- 使用 methods()函式列出“+”函式的實作定義。

```
1 methods (+)

161 methods for generic function +:

• +(x::Bool, z::Complex{Bool}) in Base at complex.jl:278
• +(x::Bool, y::Bool) in Base at bool.jl:96
• +(x::Bool) in Base at bool.jl:93
• +(x::Bool, y::T) where T in Base at bool.jl:104
• +(x::Bool, z::Complex) in Base at complex.jl:285
• +(a::Float16, b::Float16) in Base at float.jl:392
• +(x::Float32, y::Float32) in Base at float.jl:394
• +(x::Float64, y::Float64) in Base at float.jl:395
```



# 多重分派的運作



- 例如定義函式 `f` 的不同方法，可以接受不同型別的引數。若未定義引數型別的話則為 `Any` 型別。

```
1 f(x::Number, y::Number) = 2x - 3y
2 f(x::Float64, y::Float64) = 2x - 3y
3 f(x, y) = println("Dispatched to Any")
```

- 當呼叫 `f` 函式時，根據不同的引數型別，分派給不同的 Method。

```
1 # 符合 Float64
2 f(5.0, 3.0)
```

1.0

```
1 # 最符合 Number
2 f(5, 3)
```



# 多重分派的運作

- 使用 @which 巨集，可以查看會被分派到哪一個 Method。

```
1 @which f(5.0, 3.0)
```

f(x::Float64, y::Float64) in Main at In[2]:2

- 多重分派中，若 Method 有含糊 (ambiguous) 定義的話，可能會造成 MethodError。例

如：

```
1 f2(x::Float64, y) = 2x - 3y
2 f2(x, y::Float64) = 2x - 3y
3
4 f2(2.0, 3.0)
```

```
MethodError: f2(::Float64, ::Float64) is ambiguous. Candidates:
f2(x, y::Float64) in Main at In[8]:2
f2(x::Float64, y) in Main at In[8]:1
Possible fix, define
f2(::Float64, ::Float64)

Stacktrace:
[1] top-level scope at In[8]:3
```



# 型別參數化 (Type Parameter) 的 Method (Parametric Methods)



- Method 可以使用 type 參數做為引數的類型，如下例示範，傳入引數要屬於 Number 或是其子類型，且 2 個引數類型相同。

```
1 f3(x::T, y::T) where{T<:Number} = 2x - 3y
```

```
f3 (generic function with 1 method)
```

```
1 f3(5, 3)
```

```
1
```

- 以上例來說，若是傳入的 2 個引數型別不同，或是型別不符合 Number 或其子型別的話，則會產生 MethodError。

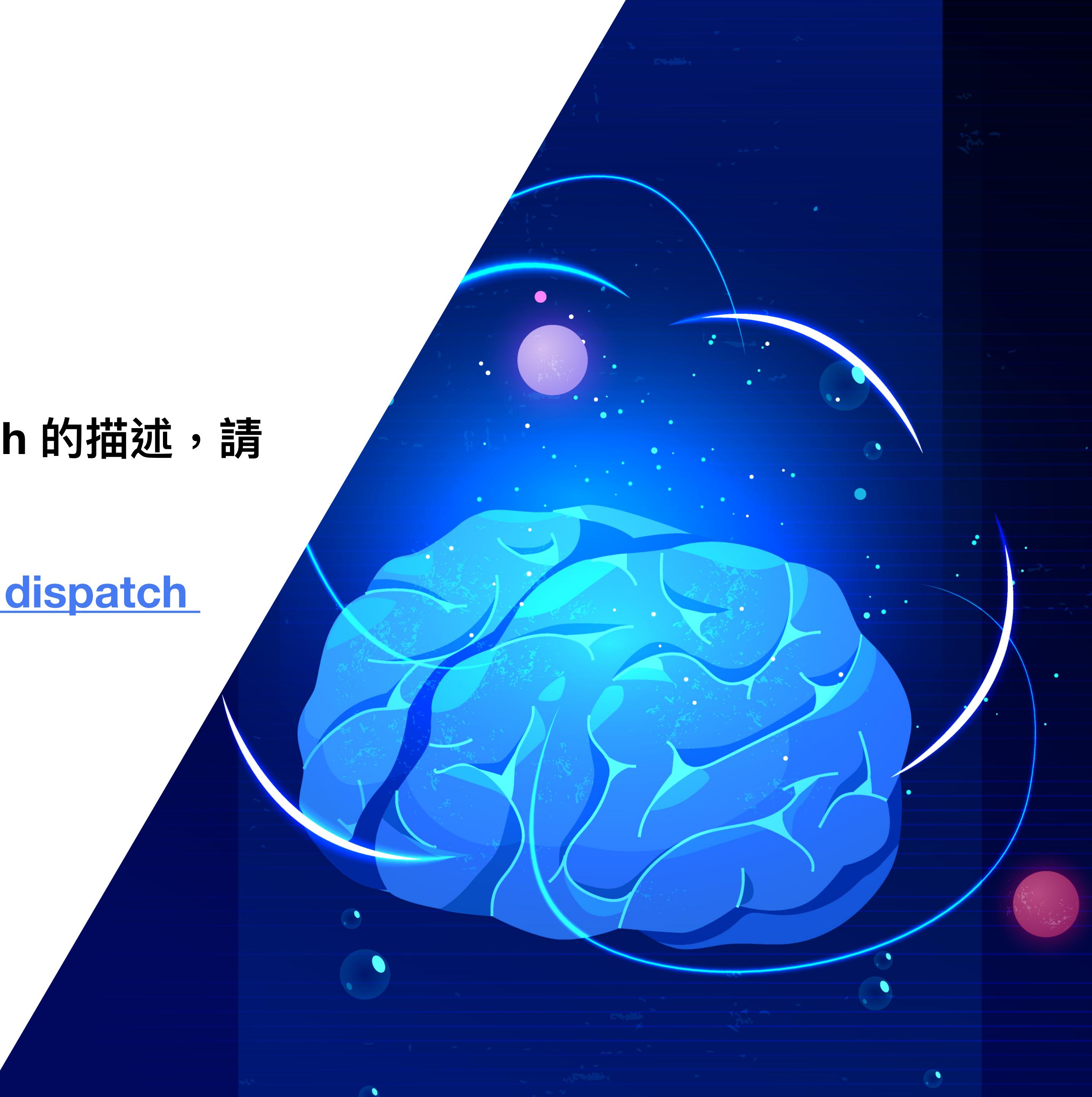
# 知識點 回顧

- Multiple Dispatch (多重分派) 被認為是 Julia 語言中最強大的單一特性 (在官方文件中提到: "... multiple dispatch on the types of values is perhaps the single most powerful and central feature of the Julia language." )。
- 多重分派根據傳入函式的引數型別，去判斷應該調用哪一個 Method。這讓我們在開發過程中能彈性提供函式不同的行為。



## 推薦閱讀

- Julia 官方文件中對於 Multiple Dispatch 的描述，請參考 [Methods 章節](#)
- Julia語言—從入門到專案系列 [Multiple dispatch](#)
- [Wikipedia - Multiple Dispatch](#)





解題時間

請跳出 PDF 至官網 Sample Code  
& 作業開始解題