CH. 10

函式與方法 (FUNCTION & METHOD)

HORAZON

C#程式設計

為什麼需要函式？

在撰寫程式時，我們經常會遇到**重複的邏輯**：

```
// 計算圓 A 面積
double areaA = 3.14 * 5 * 5;
Console.WriteLine("Area A: " + areaA);

// 計算圓 B 面積
double areaB = 3.14 * 10 * 10;
Console.WriteLine("Area B: " + areaB);
```

這樣的寫法有兩個缺點：

1. **重複程式碼**：難以維護，如果要改公式 (例如 3.14159)，要改很多地方。
2. **可讀性差**：無法一眼看出這段程式在做什麼 (雖然這個例子很簡單)。

函式 (FUNCTION) / 方法 (METHOD)

將一段**特定功能**的程式碼包裝起來，並給它一個**名字**。

- **重用性 (Reusability)**：寫一次，用很多次。
- **抽象化 (Abstraction)**：隱藏實作細節，只關心功能。
- **模組化 (Modularity)**：將大程式拆解成小零件。

> 在 C# (物件導向語言) 中，定義在類別內的函式通常稱為 **方法 (Method)**。

語法結構

```
回傳型別 方法名稱（參數列）
{
    // 方法本體（要做的事情）

    return 回傳值;
}
```

- **回傳型別** (Return Type)：執行完後會給出什麼資料？(如 `int`, `string`, `bool`)。若不回傳任何東西，則用 `void`。
- **方法名稱** (Method Name)：通常使用動詞開頭，PascalCase (如 `CalculateArea`)。
- **參數列** (Parameters)：輸入給方法的資料。

範例：定義一個加法方法

```
// 定義方法
int Add(int a, int b)
{
    int result = a + b;
    return result; // 回傳結果
}
```

```
// 呼叫方法 (Call)
int sum = Add(5, 3);
Console.WriteLine(sum); // 8
```

範例：沒有回傳值 (VOID)

如果方法只是要「做一件事」而不需要產出結果，型別使用 `void`。

```
void SayHello(string name)
{
    Console.WriteLine($"Hello, {name}!");
    // 不需要 return，除非想提早結束
}
```

```
SayHello("Horazon"); // 輸出: Hello, Horazon!
```

參數 (PARAMETERS) 與 引數 (ARGUMENTS)

- **參數** (Parameters)：定義方法時的變數 (如 `int a`, `int b`)。
- **引數** (Arguments)：呼叫方法時實際傳入的值 (如 `5`, `3`)。

```
void PrintInfo(string name, int age)
{
    Console.WriteLine($"{name} is {age} years old.");
}

// 呼叫時順序要對應
PrintInfo("Alice", 20);
```

變數的作用域 (SCOPE)

在方法內宣告的變數，稱為**區域變數** (Local Variable)。
它們**只在該方法內有效**，方法結束後就會消失。

```
void Test()
{
    int x = 10; // x 只有在 Test 裡面活著
}

void Main()
{
    // Console.WriteLine(x); // 錯誤！這裡找不到 x
}
```

不同的方法可以有同名的區域變數，它們互不影響。

實戰練習 1

請撰寫一個方法 `CalculateCircleArea`，輸入半徑，回傳面積。

```
double CalculateCircleArea(double radius)
{
    return 3.14159 * radius * radius;
}

// 主程式
double area = CalculateCircleArea(5.0);
Console.WriteLine($"圓面積: {area}");
```

這樣一來，計算公式就統一管理了！

實戰練習 2：BMI 計算機

試著寫一個計算 BMI 的方法：

BMI = 體重(kg) / 身高平方(m^2)

```
double CalculateBMI(double weight, double heightCm)
{
    double heightM = heightCm / 100.0;
    return weight / (heightM * heightM);
}

double myBMI = CalculateBMI(70, 175);
Console.WriteLine($"BMI: {myBMI}");
```

RETURN 的特性

`return` 關鍵字有兩個作用：

1. **回傳數值**給呼叫者。
2. **立即結束**該方法的執行。

```
string CheckScore(int score)
{
    if (score < 0 || score > 100)
    {
        return "錯誤的分數"; // 遇到 return 直接結束，下面不會執行
    }

    if (score >= 60) return "及格";
    else return "不及格";
}
```

總結

- **方法** (Method) 用來封裝重複的邏輯，提高程式可讀性與維護性。
- **回傳型別**決定方法產出的資料類型，`void` 代表不回傳。
- **參數**是方法的輸入，讓方法更具彈性。
- return 用來回傳值並結束方法。
- 變數有其**作用域**，方法內的變數外面看不到。

下章預告：類別與物件 (CLASS & OBJECT)

我們即將進入物件導向程式設計 (OOP) 的核心！

- 什麼是類別 (Class)？
- 什麼是物件 (Object)？
- 屬性 (Field/Property) 與 建構子 (Constructor)