*Why?防禦性程式發生在程式設計師不相信系統的輸入(input)資料時。一般來說我們不需要測試輸入的資料來確保功能正常。系統中大部分程式碼應該可以假設輸入的資料是正確的，只有一小部分的程式需要真的檢查資料，這通常發生在資料第一次輸入的時候。只要資料在輸入進系統時被檢查過，那麼就應該可以假設它是正確的。*

健壯性：*軟體本身的周密程度。即撰寫程式時考慮到各種不同的使用情況，並事先加以定義處理，避免使用時產生錯誤。*

*=>*可以讓系統出現未預期錯誤的機會降到最低。

原則，絕對不要相信使用者送進來的參數值

缺點在於這常是多餘的資源浪費，不但浪費效能，也增加程式碼維護的成本。

**進function 前：**

**Type Hint 類別檢查：**在 function 的參數前宣告類別

進function：

**function 一開始：**

### 內部檢查

碰到 string 或 int 這種不能再參數區宣告的

### 自動轉換

自動轉換通常用在公開類別或方法，例如框架或函式庫，因為可能的使用情境太多變了，會盡量讓介面可以吃下各種內容做轉換，

### Fallback 與預設值

Runtime Error

### 黑洞

NullObject 模式或者任何無行為的物件與回傳值。有時候我們的回傳值可能是空，此時要考慮該回傳什麼數值來給不知名的使用者，

大多數的寫法強調如何避免錯誤，讓系統自我還原

防禦型寫法大多用在不確定環境的公開介面上，如果是私有函數，或者團隊內有約定的設計模式時，則應該要盡量讓錯誤的參數提早報錯，在第一時間就發現並修正。然後讓錯誤的回傳值盡量多一兩層修復機制，不要讓使用者察覺。