第 1 週，第 3 課：錯誤訊息和除錯

目標

— 學習閱讀錯誤訊息

— 學習逐行除錯方法

學完本課，你將能夠

* 在 Julia 的錯誤訊息中找到有用的資訊
* 在單行 Julia 代碼中查找錯誤並修復它們
* 逐行執行 Julia 代碼，在發現錯誤時修復它們
* 偶爾回去修復你錯過的錯誤

除錯一些幾乎是第 1 課的代碼

假設我們錯誤地嘗試使用以下代碼運行第 1 課的練習[1].

```
mystiring1 = "Hello, world
prinln(mystring1)
```

你會發現它沒有辦法順利運行（並且，依照你嘗試運行它的方式，你可能會看到一個錯誤訊息）。

透過逐行檢查代碼，你可以找出原因。在這裡，當我們嘗試從 REPL 運行第一行時，它停住了，沒有任何錯誤消息。你現在的工作是找出原因，並從仔細查看代碼開始。它是有效的代碼嗎？不是！沒有結束雙引號字符來指出字串值的結尾。正如 Julia 所見，你仍然在字串中輸入字符[2]。

你可透過在結尾添加缺少的雙引號字符來修復該行。

在此有一個很好的技巧：在結束雙引號之後添加一個分號。在 REPL 中，這會抑制該行的輸出，這意味著你需要閱讀的內容更少，干擾更少。當然，在某些情況下，你會希望查看評估一行代碼的輸出，這並不總是正確的做法。

[1] NB: 如果你從新的 REPL 開始，這效果最好。它包含拼寫錯誤，你應該小心地按原樣輸入下面的示例 (拼寫錯誤及全部)。

[2] 如果將這兩行放在.jl 檔案中並嘗試使用 `include()` 運行它，則會看到錯誤訊息 "LoadError: syntax: incomplete: invalid string syntax"，這會更有幫助，但不是很多。事實上，發生語法錯誤的方式有很多種，Julia 並沒有很努力地告訴你錯誤到底是什麼。

好的，現在代碼變成了

```
mystiring1 = "Hello, world";
prinln(mystring1)
```

當我們在 REPL 運行第一行時，正如預期的那樣看不到輸出。所以我們運行第二行。哦親愛的。螢幕上突然出現很多行訊息，其中一些是紅色的！

以 ERROR 開頭的行是重要的[3]：這是錯誤訊息。在其中，Julia 為你提供了可能存在的問題的一些指示。在此，訊息是 `UndefVarError: prinln not defined`。當然這也是很容易修復的，所以現在代碼變成了

```
mystiring1 = "Hello, world";
println(mystring1)
```

現在我們看到一個錯誤：訊息是 `UndefVarError: mystring1 not defined`。多麼出乎意料！這似乎是錯誤的：該行是有效的 Jullia 代碼！當我們檢查 `println()` 行時，變數名稱似乎沒有問題，它怎麼可能是未定義的 ?? 這在除錯中經常發生：一些錯誤掩蓋了其他錯誤。你修復了你找到的第一個錯誤，最終它們會揭示在你的代碼中更早地錯誤。當然，這裡在它上面的行中有一個拼寫錯誤。如果我們沒有立即發現，錯誤訊息說 `mystring1 not defined` 的事實應該會有所幫助。這意味著這是 Julia 第一次看到名稱 `mystring1`。我們認為稍早已創建它是錯誤的：要嘛根本沒有創建它的行，要嘛這行是錯誤的。找到應該創建它的行，並修復錯誤。最終，我們最後一次正確的更改為

```
mystring1 = "Hello, world";
println(mystring1)
```

關於除錯的一些要點

以這種方式逐行檢查絕不是發現錯誤的唯一方法，但它是開始學習如何除錯代碼的好地方。你應該經常練習並能熟練地解讀錯誤訊息。

事實上，除錯對於程式設計師來說絕對是一項至關重要的技能。就像編寫代碼一樣，這裡有很多很多風格的除錯方式[4]。但它們的核心是能夠以非常挑剔的眼光閱讀代碼。那個能力是你應該培養的。

如果你繼續進行大量編程，你可能希望了解可以幫助你完成任務的軟體。那裡有很多選擇[5]，但在本課程中我們不會討論到。

[3] 非常重要，以至於在大多數終端中它都顯示為鮮紅色。

[4] 我們將在下一課中看到，編寫代碼從根本上是一種創造性表達的行為。除錯也允許你按照自己的方式進行。

[5] 對於 Julia，主要支持的是 `Debugger.jl` 套件包，它非常有用但需要一段時間來學習，以及 `Juno` 開發介面，它雖不是專門用於除錯，但有很多功能可以有所幫助—這些功能可以通過使用 `Debugger.jl` 進行擴展。

回顧與總結

* 逐行除錯是一項非常有用的技能學習。
* Julia 中的錯誤訊息包含有用的提示及許多其他細節。
* Julia[6] 中的錯誤訊息不能保證它們指向實際錯誤。
* 在 REPL 和運行具有相同代碼的.jl 檔案的逐行輸出並不完全相同。
* 對於逐行輸出，在行尾使用分號來抑制輸出是很有幫助的。

[6]或任何其他語言，就此而言。