**Q1**

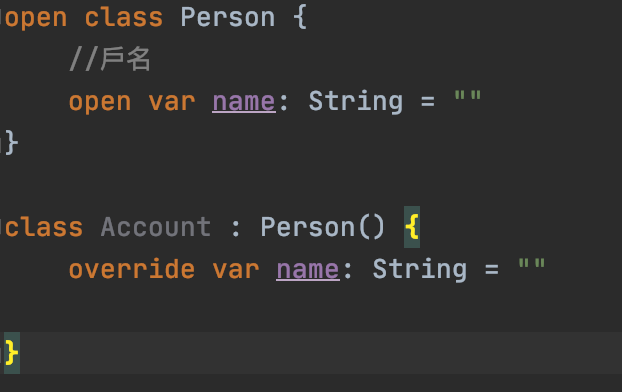
**Inheritance (繼承)**

**Subclassing**

如果一個類別 B 「繼承自」另一個類別 A ，就把這個 B 稱為「 A 的子類別」，而把 A 稱為「 B 的父類別」也可以稱「 A 是 B 的超類」。繼承可以使得子類別具有父類別的各種屬性和方法，而不需要再次編寫相同的代碼。在令子類別繼承父類別別的同時，可以重新定義某些屬性，並重寫某些方法，即覆蓋父類別的原有屬性和方法，使其獲得與父類別不同的功能。另外，為子類別追加新的屬性和方法也是常見的做法。

當子類別要繼承父類別時，會在父類別的前面加 「 open 」，換句話說，當在一個類別被加上 open 時，別人就可以來繼承這個類別裡面的屬性或功能。





**Overriding**

雖然父類別的方法能夠被繼承，但若子類別想要有自己的實作方式，還是可以覆寫的。作法與繼承類別相似，在父類別方法上加上「 open 」的關鍵字才能讓子類別覆寫。

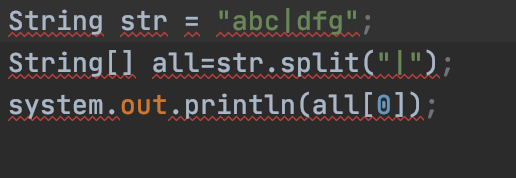
在子類別覆寫的方式與 Java 很像，一樣需要加上「 override 」。但 Java 是使用標記，而 Kotlin 是關鍵字。

**Q2**

分割字串需要加上中括號split（“[$]”）

****

特殊用法–當split()分割字串遇上特殊符號



在正規表示式中是個已經被使用的特殊符號（”.”、”|”、”^”等字元）  
所以想要使用 | ，必須用 \ 來進行轉義，而在 java 字串中，\ 也是個已經被使用的特殊符號，也需要使用 \ 來轉義。

所以應為： String[] all = str.split((“\\|”)；

**Q3**

**Key-value pair collection**

鍵-值資料庫，或鍵-值儲存，是設計用來儲存、檢索和管理關聯陣列的資料儲存範式，關聯陣列是現今更常稱為「字典」或雜湊表的一種資料結構。字典包含對物件或記錄的一個收集，依次、記錄內有多個不同的「域」或稱欄位，再次、每個欄位都包含資料。這些記錄使用唯一標識這個記錄的「鍵」來儲存和檢索，鍵也用來在資料庫中快速的找到資料。

這提供了可觀的靈活性並更加緊密的遵循了現代概念如物件導向程式設計。不同於多數的關聯式資料庫，由於不使用預留位置或輸入參數來表示可選值，鍵-值資料庫經常比同等的關聯式資料庫使用更少的記憶體。

1.使用 SQLite 保存數據

定義架構和契約

SQL 數據庫的主要原則之一是架構：關於如何組織數據庫的正式聲明。該模式反映在用於創建數據庫的 SQL 語句中。創建一個伴隨類（稱為契約類）會很有幫助，該類以系統的自記錄方式顯式指定架構的佈局。

契約類是用於定義 URI ，表和列名稱的常量的容器。契約類允許在同一package 中的所有其他類中使用相同的常量。這樣一來，可以在一處更改列名，並將其傳播到整個代碼中。

組織合同類的一種好方法是將整個數據庫的全局定義放在類的根級別。然後為每個表創建一個內部類。每個內部類枚舉相應表的列。

2. 儘管這些 API 功能強大，但它們的層級相當低，需要大量時間和功夫才能使用:沒有原始 SQL 查詢的編譯時驗證。隨著數據圖的更改，您需要手動更新受影響的 SQL 查詢。此過程可能很耗時且容易出錯。

您需要使用大量樣板代碼在 SQL 查詢和數據對象之間進行轉換。

由於這些原因，建議使用 Room Persistence Library 作為訪問應用程序 SQLite 數據庫中信息的抽象層。

3. 使用 Room 將數據保存在本地數據庫中

Room 在 SQLite 上提供了一個抽象層，以允許流利的數據庫訪問，同時利用SQLite 的全部功能。

處理大量數據的應用程序可以從本地持久存儲這些數據。最常見的是緩存相關數據。這樣當無法連網絡時，用戶仍可以瀏覽該內容。重新連網時，所有用戶啟動的內容更改都將同步。



**Q4**

**Arraylist vs. Linkedlist**

在集合( Collection )中，我們常使用 List 這個介面( Interface )。要實作 List ，通常會使用 ArrayList 或 LinkedList 這兩個 Class。

ArrayList 採用的資料結構是 Array (陣列)； LinkedList 採用的資料結構是 LinkedList (翻譯為鏈結串列或連結串列)。

鍊節串列的每一個節點包含的資訊有：

1. 本身儲存的資料

2. 指到另一個節點所在的位址。

鍊節串列可分為單向鍊節串列，雙向鍊節串列，迴圈鍊節串列等等。

當我們需要頻繁的取得 List 內的資料時，使用 ArrayList 的效率較好，因為它的資料結構是 array ， array 採用隨機存取( random access )的方式，也就是直接透過索引( Index ) ，取得值。 而 LinkedList要取得值必須從開始的節點一路搜尋到要需要的值的位址，需要花費大量的時間。

當我們需要頻繁的新增資料與刪除資料時，使用 LinkedList 較好。因為使用LinkedList 新增/刪除資料時，只要改動被新增/刪除的那筆資料前面一個節點所指向的位址及可。但若使用，要新增或刪除 array 的資料，新增/刪除結束後，需要重新搬動剛才被新增/刪除之後的所有節點，並且重新給定之後所有節點的索引值，因此效率會較差。

****

（ArrayList）

**Q5**

**Value Type variables**

**實值類型變數**

使用此變數來保存單一資料點，並在多個位置使用它。

使用實值類型變數的時機：此「預留位置」值可代表文字或數字資料。

建立變數之後，您可以透過在數個工作台命令中插入變數的方式來加以使用。修改變數的值時，這個值會反映在任何執行於 TaskBot / MetaBot 邏輯的後續命令中。

可以將實值類型變數標示為：

**記錄到分析 -** 將資料上傳到 Bot Insight 控制面板進行資料分析。預設會啟用此選項。

（如果您的自動化作業已從舊版移轉到 Enterprise Enterprise 用戶端 11，您可以編輯變數以手動選取此選項。預設不會選取這類變數進行分析。）

**規則運算式 -** 將變數用於需要在檔案、資料夾和視窗標題命令中進行模式搜尋的自動化。

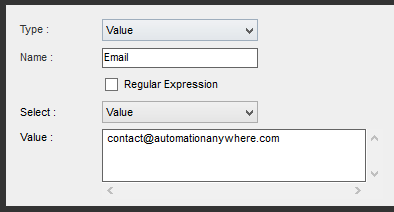
（只能在 [值] 文字方塊中輸入第一行文字。

您可以在支援萬用字元的命令中使用變數的規則運算式。例如，您可以在檔案/資料夾的某些子命令 (也就是複製、刪除、壓縮及列印多個檔案/資料夾) 中使用規則運算式。）

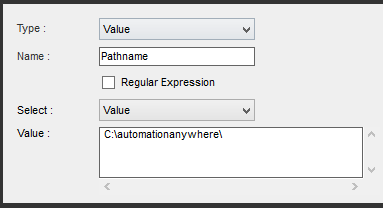
**實值類型變數的範例：**

名稱：ValueDB ，值：Oracle

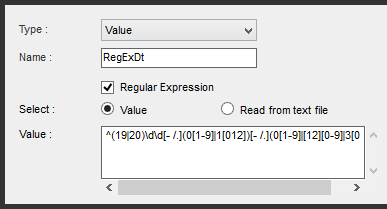
名稱：email，值：contact@automationanywhere.com



名稱：Pathname，值：C:\Automationanywhere\



名稱：RegExDt，值：^(19|20)\d\d[- /.](0[1-9]|1[012])[- /.](0[1-9]|[12][0-9]|3[01])$



**可用於實值類型變數的一般名稱範例包括：**

Pathname、CounterValue、DatabaseName、email 等等

**Q6**

val numberInt: Int = Int.MAX\_VALUE

//Int 類別中的靜態常數，MAX\_VALUE 為 Int 的最大值 2147483647

//Int 範圍為: -2147483648 ~ 2147483647

Integer.MAX\_VALUE + 1 = Integer.MIN\_VALUE = -2147483648

超過Int最大值，再大的數就要用 long (最大值 2 的63 次方 – 1) 或者 BigDecimal 表示。