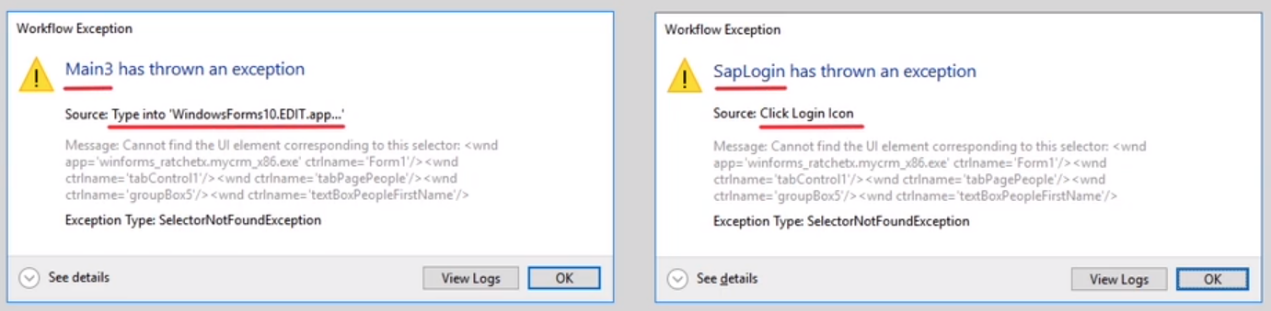
**学习大纲**：

* 如何使用调试功能
* 如何使用Find Element和Element Exists与应用程序同步
* 了解**TryCatch活动**的工作原理和使用方法

**1. UiPath的调试工具**

1) 容易被忽视的一点，也是在调试程序时非常有用的一点是，为你的Workflow和活动命名。

原因是在错误提示中，UiPath会使用Workflow和活动的名称来告诉你哪里出了问题。

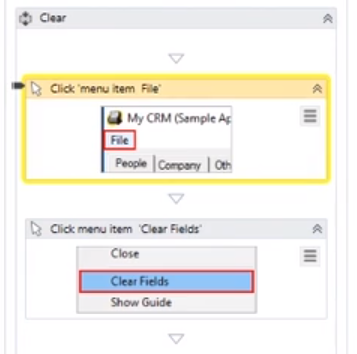


2) Start Debug工具

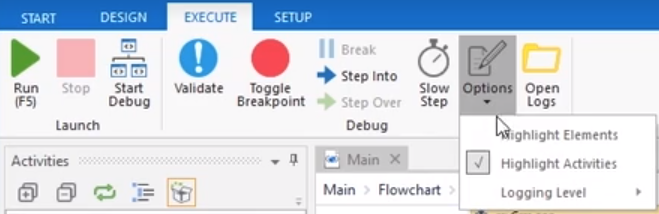


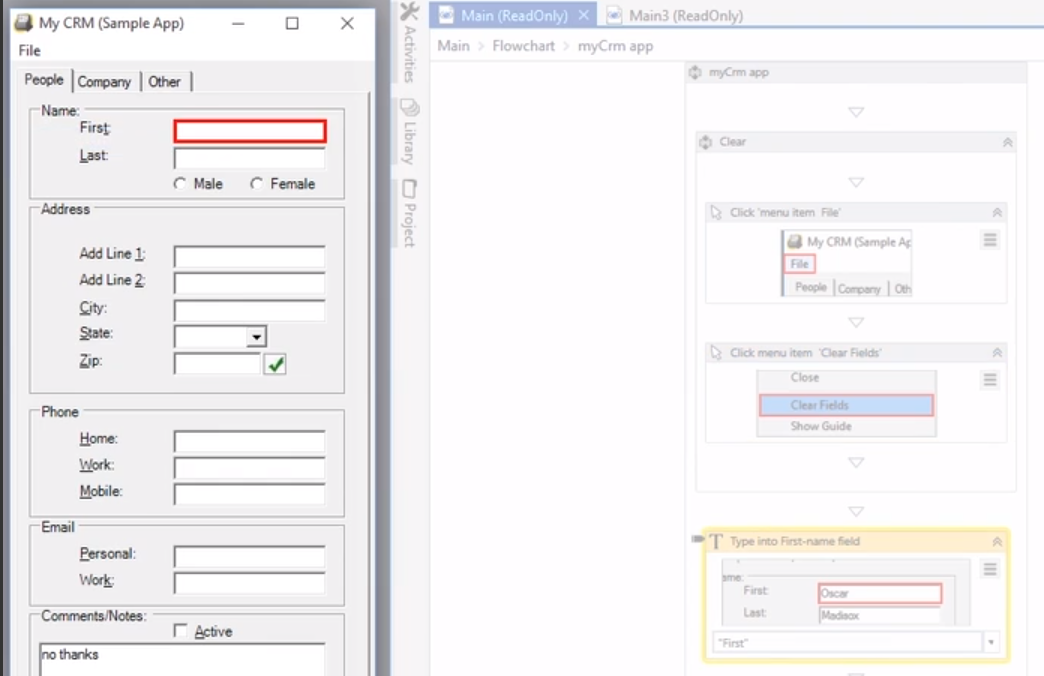
进入Execute菜单并点击Start Debug按钮，程序会在Debug模式中运行。它有以下特征：

* 运行时，正在执行的活动会高亮标示出来，以方便查看。这种方法对于解决Input和Output的问题尤其有帮助。

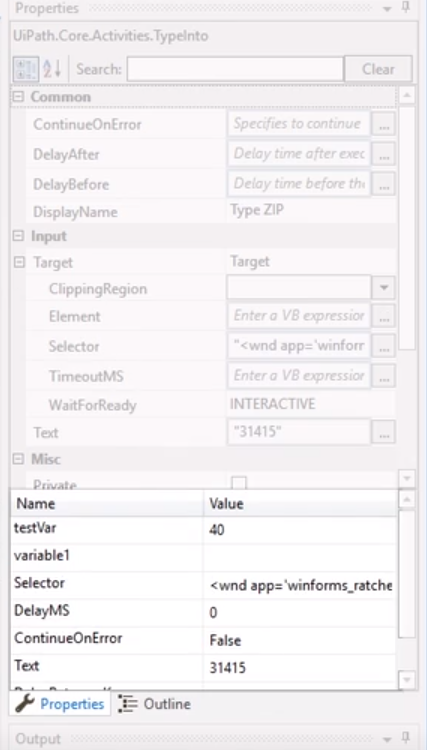


* 在Execute菜单的Options中选上Highlight Elements，运行时，活动所作用的元素也会高亮显示出来。

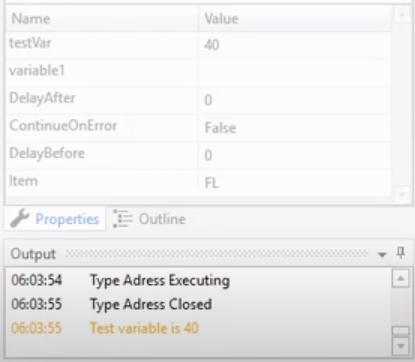




* Property Inspector：实时显示已声明的变量的当前值，适合用来解决一些程序逻辑问题。



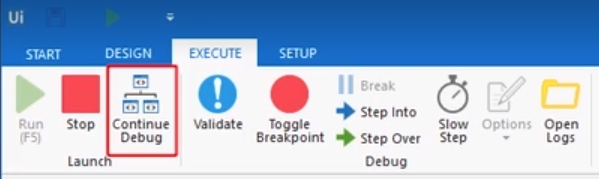
* Ouput面板：显示正在发生的活动情况，以及**Write Lines活动**和**Log Messages活动**的信息。



* Slow Step：点击按钮放慢程序的执行速度，再次点击恢复原速。在程序运行过程中也可使用或取消使用。

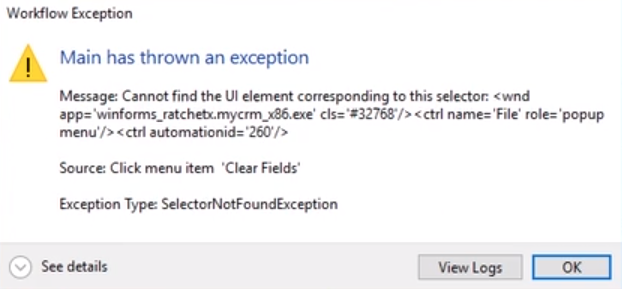
Toggle Breakpoint：为某个活动添加断点。Start Debug后程序将在断点处停止，以便查看当前的运行情况、变量的值、Output面板的信息等。

* + 下一步，你可以点击Continue Debug继续运行程序。
  + 也可以点击Step Over来一步一步的执行接下里的活动。



**2. 调试Input活动**

Input类型的活动最容易出现的问题是作用的元素未找到，或找到错误的元素。

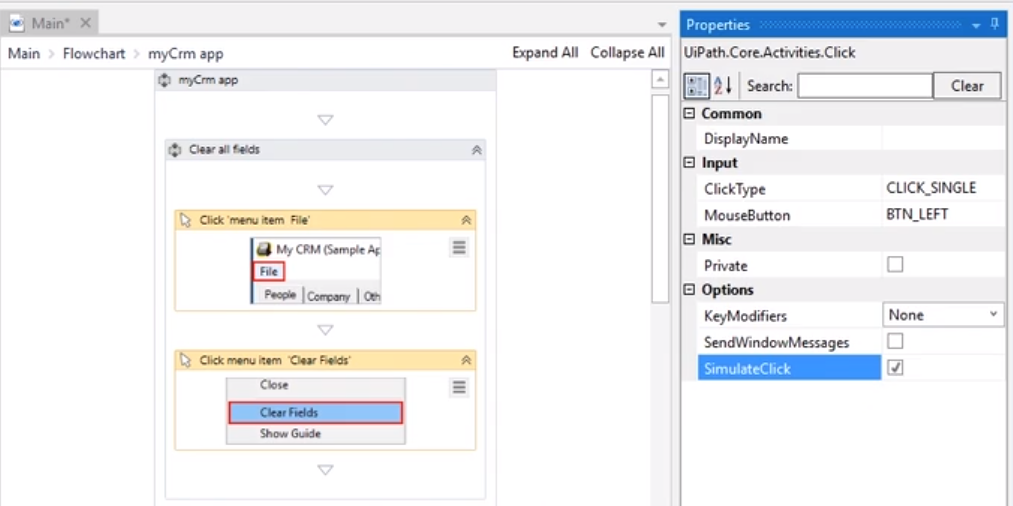


1) 检查交互的窗口是否可见。

**实例**：UiPath交互的应用程序窗口被其他窗口遮挡，导致程序无法运行。

**解决方案**：手动关掉或最小化遮挡应用程序的窗口。但是 如果你的情况不允许使用这种方法，你可以尝试更改活动的属性。在Input Method的课程中（**Click活动，Type活动**等），活动的SendWindowsMessages和SimulateClick选项都是默认不勾选的。如此，活动会模拟人操作键盘和鼠标，当应用程序的窗口在最上方时，这种方法总是有效。我们可以尝试勾选SendWindowsMessages或SimulateClick选项，来探索它们是否可以帮助我们解决问题，比如让程序在后台运行。对于该实例，只需要为Click活动勾选**SimulateClick**就可以完美解决遮挡问题。

* Tips：可以同时选中多个活动来编辑它们的共有属性。

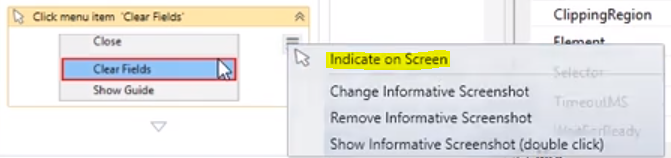


2) 更改Input的方法（比如使用其他活动）。

3) 检查Selector。

学习Selector的课程了解其原理，再进行调试。但可以尝试两个能迅速解决Selector问题的方法：

* 重新在界面上选取Selector来刷新它。



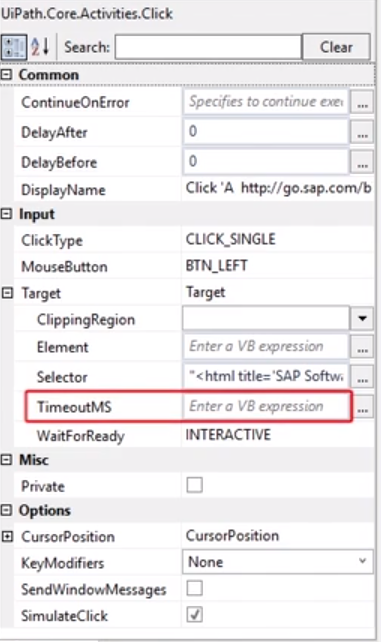
* 打开Selector Editor，选择右上角的Attach to Live Element。这可以解决Selector过于具体的问题。

**3. 调试时间和同步问题**

比如在某个网页或者应用程序里，UiPath尝试与某个元素进行交互，但该元素还未加载就绪。

**实例**：在某个网页点击登陆按钮，当登陆按钮弹出时，输入用户名和密码（参考第8课）。此处我们将输入用户名的**Type Into活动**的**Timeout参数**设置为0，人为制造一个Error并解决它。

* 所有的Click和Type活动都有一个**Timeout参数**，该参数设置UiPath等待某元素出现的最长时间，超出时间仍未找到元素，UiPath就会弹出错误提示。该参数的**默认值是30秒**（以毫秒为单位，1毫秒=1/1000秒）。

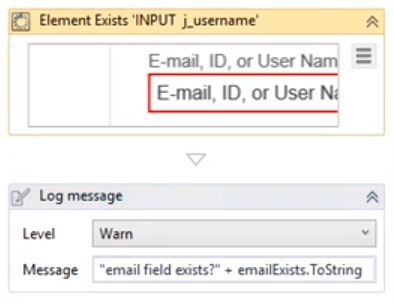


**解决方案**：有三。

* **Element/Image Exists活动**

Element/Image Exists活动不会影响Workflow的运行，只是返回一个布尔值(True/False)，默认等待时间是3秒钟（3000毫秒）。

可以在其后添加If-Else活动或者Do While循环来进行下一步操作。



* **Find Element/Image活动**

与前者不同，Find Element/Image活动会使Workflow停止，直到某个元素/图片出现，默认等待时间是30秒钟。

* **Wait Element/Image Vanish活动**

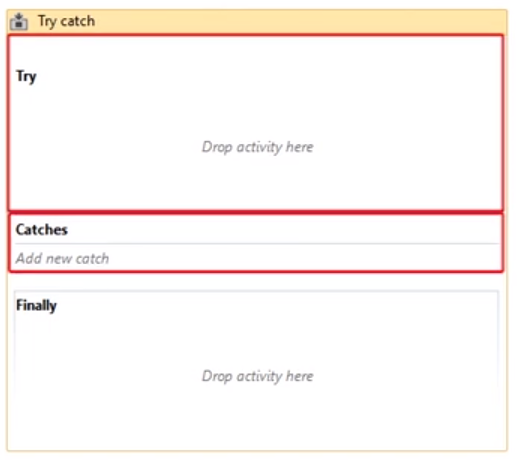
Wait Element/Image Vanish活动会使Workflow停止，直到某个元素/图片消失。

**4. 使用Try Catch活动**

我们经常需要让程序在发现错误时做出一系列的操作，有时甚至需要让它在遇到特定的错误时继续运行下去，而不是跳出错误信息并中止运行。**Try Catch活动**就是为了实现这些目的而设计的。

1) **Try Catch活动**

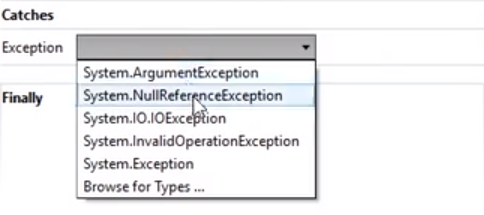
Try Catch活动由三个部分组成，Try，Catches和Finally，其中Finally是可选的。



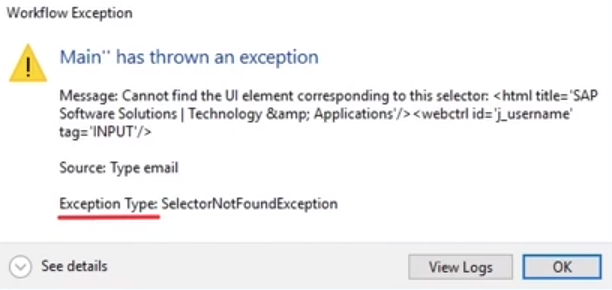
**Try**：放入可能出错的活动或流程。

**Catches**：选择错误类型，并放入发生该类错误后需要执行的操作。可添加多个Catch来捕获不同类型的错误或异常。

* 最常用的类型是**System.Exception**，它可以捕获所有种类的错误。



* 判断错误类型的方法是，看错误弹窗中的Exception Type信息。

****

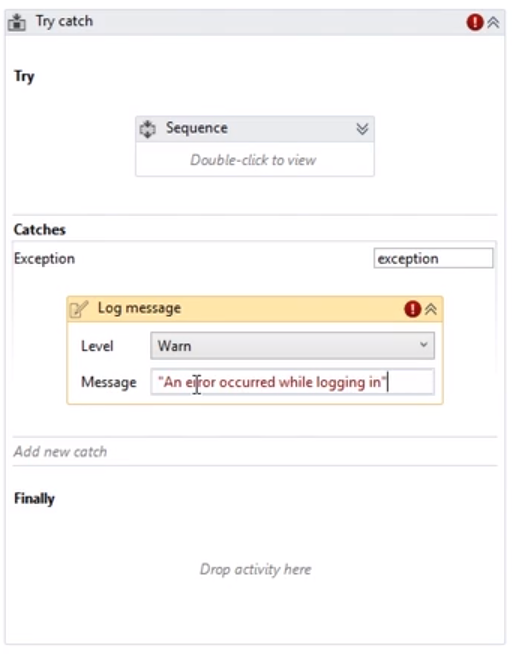
**Finally**：运行完Try和Catches之后的操作，可选。

2) 实例

沿用本课第三节的例子，在某网站登陆界面录入登陆信息，将所有活动都放入Try容器中，人为制造等待时间的错误。

**实例1**：不中止程序，但在发生错误时显示错误信息。

* 添加一个Catch容器，选择System.Exception。
* 此处使用**Message Box活动**会中止程序，因此不能用；使用**Write Line活动**会在Output面板中输出一条信息而不会中止程序，我们可以使用它；使用**Log Message活动**不但可以在Output面板里显示信息，还可以将信息写入log文件中。当你要调试的程序是在无人看管的情况下运行时，使用Log Message活动进行调试无疑更加有效。



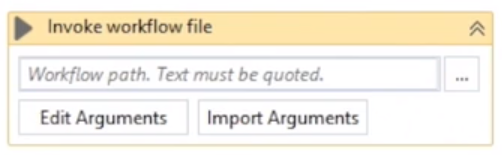
* 如果你想在捕获错误时中止程序，并显示错误信息的提示框，可以在Log Message活动后面再添加一个**Rethrow活动**。

**实例2**：你发现问题出在浏览器上，只要重新启动浏览器就能解决问题。

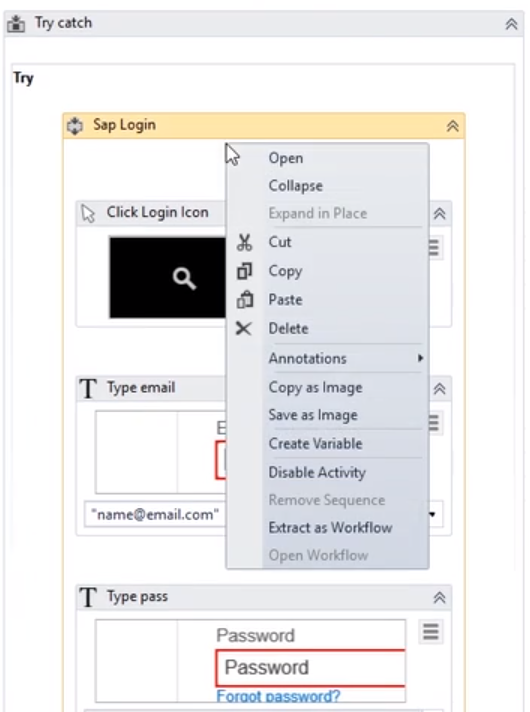
* 清空Catches容器，添加一个**Open Browser活动**并填入网站地址。
* 在Open Browser容器中添加一个**Log Message活动**，输入Recovering from Browser (从浏览器恢复)，使用Log Message活动来记录发生的情况是极好的。
* 将Try容器里的Click和Type活动复制到新的Open Brower的容器中，即在发生错误时，打开新的浏览器再次进行之前的操作。

**实例2-拓展**：我们可以发现在Try容器和Catches的Open Browser容器中的活动都是相同的。如何简化整个Workflow？

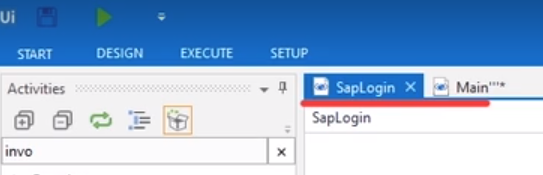
**思路**：使用**Invoke Workflow File活动**。



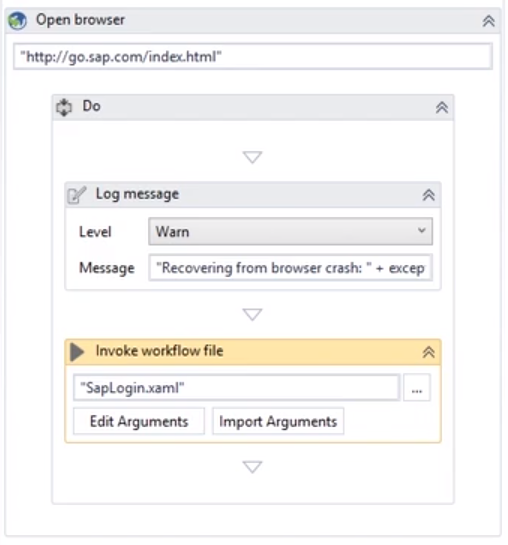
* 在Try容器的上方右击，选择Extract as Workflow，在弹出的窗口中为这个Workflow命名 (SapLogin)。



* 确认后，会自动打开提取出的Workflow，该Workflow保存在当前项目的文件夹里。Try容器中的所有活动自动替换为Invoke Workflow File活动。



* 用Invoke Workflow File活动替换Catches容器里的活动。这样就简化了整个Workflow，使整个流程更加清晰，一眼可见。

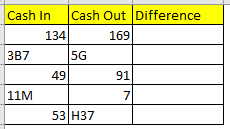


**5. 练习1**

**1) 要求**：比较Excel文件每一行的Cash In和Cash Out的不同，并写入第三列中。

* 如果Cash In或者Cash Out不是一个数字，则在第三列写入Cash In Wrong或Cash Out Wrong；
* 尽量不要使用If活动，依靠错误处理的方法来达到目的，比如**Try Catch活动**。

**2) 数据源**：



**3) 思路**：

* 因为整个流程是在Excel文件里进行的，因此先添加一个**Excel Application Scope活动**。
* 在Excel Application Scope容器中添加**Read Range活动**来读取Excel上的数据并得到一个DataTable，记得勾选Add Headers选项。
* 在Read Range活动之后添加一个**For Each Row活动**，在刚得到的DataTable中进行循环。
* 创建两个Int32型变量cashIn和cashOut，创建一个GenericValue型变量result。
* 在For Each Row的主体里添加第一个**Try Catch活动**，在Try的部分里添加一个**Assign活动**，使cashIn=**cint(row("Cash In"))**。在Catch的部分选择System.Exception并添加**Assign活动**，使result=“Cash In Wrong”。
  + **cint()**把单元格里的值从GenericValue型转换为数字型。如果单元格的值不是数字，则会报错。
* 回到第一个Try Catch活动的Try部分，在Assign活动之后再添加一个**Try Catch活动**（嵌套在第一个里面）。
* 在第二个Try Catch活动的Try部分添加**两个Assign活动**，使cashOut=cint(row("Cash out"))，result=cashIn-cashOut。在Catch的部分选择System.Exception并添加**Assign活动**，使result=“Cash Out Wrong”。
* 在第一个Try Catch活动的Finally部分添加一个**Assign活动**，使row("Difference")=result.ToString。
  + 不论执行了Try部分还是Catch部分，UiPath都会执行Finally部分的活动。
* 最后，在For Each Row活动之后添加一个**Write Range活动**，将最终的DataTable写入Excel文件中。

**6. 练习2**

**1) 要求**：现有一个Correct Errors.Xaml，它会把filesArray变量中提到的文本文件(.txt)的内容合并到一个单独的consolidate.txt文本文件里。并且如果读取文件的操作未能成功执行（因为Read Text File活动出现错误），则不把内容添加到合并的文本文件中。这个Workflow在存在几处错误，在保持原有结构的前提下，尽量通过最少的修改来修复程序和逻辑。

**2) 思路**：

* **Bug1**：执行Assign活动index=index+1时报错。原因是index变量没有初始化。对于GenericValue型变量，UiPath无法判断你需要一个数字型变量还是一个字符串型变量。
* **解决**：将index变量的类型更改为Int32型；或者不更改变量类型，直接设置index变量的默认值为0。将一个GenericValue型变量的默认值设置为""（空）则表示这是一个字符串型变量。
* **Bug2**：在循环进行了几次后报错，显示索引号超过数组边界。filesArray中有5个元素，执行循环的条件是index<=filesArray.Count，即index<=5。而数组的索引号是从0开始的，因此fileArray的最后一个元素表示为fileArray(4)。在当index=5时，filesArray(index)就超出了数组边界。
* **解决**：将循环条件更改为index<filesArray.Count。
* **Bug3**：虽然fileArray中有5个元素，但实际上只存在4个文本文件，因此最终的consolidate.txt里应该只有4行内容，但实际运行后却有5行，且第5行和第4行的内容都是第4个文本文件的内容。原因是，虽然UiPath在读取第5个文件时Catch了Read Text File活动的错误，但是由于Append Line活动和Read Text File活动不在同一个Try Catch里，Append Line活动依然被执行了（相当于把第4个文本文件的内容又黏贴了一遍）。
* **解决**：将Append Line活动和Log Message活动都放到第一个Try Catch的Try部分。

**\*本课使用过的新活动、方法、函数等：**

**Wait Element Vanish活动**

**Wait Image Vanish活动**

**Try Catch活动**

**Rethrow活动**

**Invoke Workflow File活动**

**Append Line活动**

cashIn=**cint**(row("Cash In"))