**Practical Exercise - Walkthrough**

**Part A:**

**Practically all of the activities to be done should be contained inside an Excel Application Scope. The first step after creating one of those is to read the Excel file.**

* Find and add an **Excel Application Scope** to the main area

            o       Type in the full workbook path to **Sample Columns.xlsx** into the Workbook Path parameter

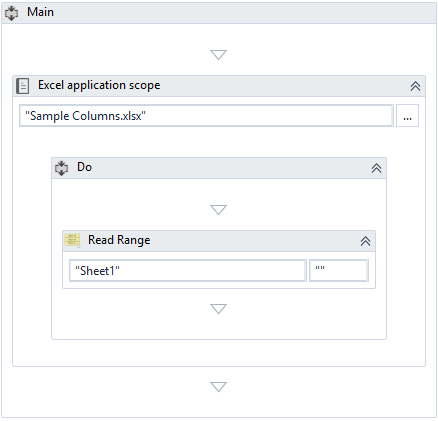
            o       Make sure the ‘Visible’ option is checked, so the activities will be performed just like a human

* Find and add a **Read Range**activity into the **Do**container of the **Excel Application Scope**.

            o       Set the Range to “” so the entire sheet is read

            o       In the output parameter, use the shortcut Ctrl+K to create a DataTable called **inputsTable**

* This is what the workflow should look like so far:



Next, use a **For Each Row** activity and sum the first two columns.

* Find an add a **For Each Row**activity and add it below the **Read Range**activity

          o       Set it to loop through the DataTable created earlier, **inputsTable**

* Create an Int32 variable called **rowIndex** - this will keep track of what row to write on later
* Find and add an **Assign**activity inside the body of the **For Each Row** activity

          o       Assign **inputsTable.Rows.IndexOf(row) +1** to **rowIndex**

                   ▪        This sets the value of rowIndex to the match the current row in the loop

                   ▪        The + 1 is because Excel Rows start counting at 1, whereas DataTables start at an index of 0 - this difference needs to be compensated for

* Below that activity, find and add two **Get Row Item** activities

          o       For the first one, set the column index to 0 and the row to **row**(the temporary loop variable)

          o       In the output parameter, use the Ctrl+K shortcut to create a variable called **valueA**

          o       For the second one, set the column index to 1 and the row to **row**

          o       In the output parameter, use the Ctrl+K shortcut to create a variable called **valueB**

* Find and add an **Assign**activity below

          o       Assign **valueA + valueB**to **valueC**(Use the variable creation shortcut here as well)

* Find and add a **Write Value** activity next

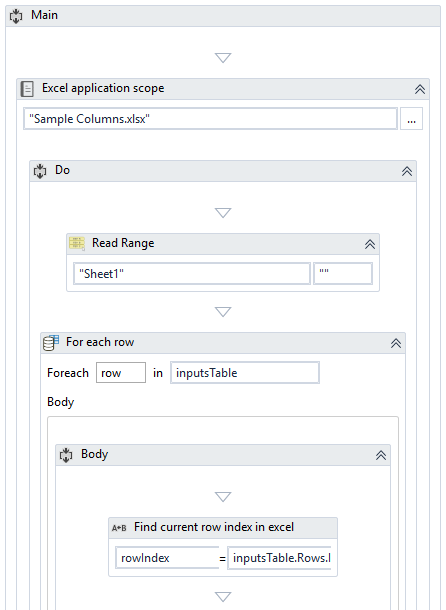
          o       Keep the sheet as Sheet1

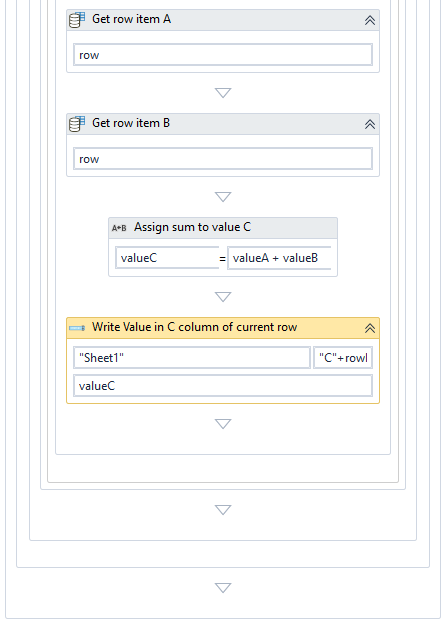
          o       Set the range (the location in the sheet to write to) to **“C” + rowIndex**.**ToString**

                   ▪        Throughout the loop, this will evaluate to “C1” then “C2,” and so on down the third column

          o       Set the value to **valueC**

* This is what the final workflow should look like:





**Part B:**

**In this part, the file will be read without an Excel Application Scope because the automation will be done internally.**

* Find and add a **Read Range**activity into the main sequence.

           o       Set the WorkBook path to the full path of the **Sample Columns.xlsx** workbook

           o       Set the Range to “” so the entire sheet is read

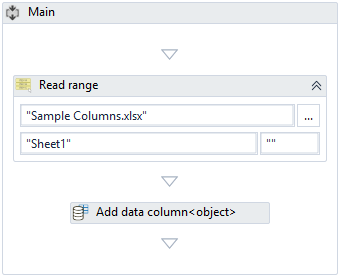
           o       In the output parameter, use the shortcut Ctrl+K to create a DataTable called **inputsTable**

* Find and add an **Add Data Column**activity below

           o       Set the ColumnName to “C”

           o       Set the DataTable parameter to **inputsTable**

           o       Set the argument type to object



* Find and add a **For Each Row**activity below that

           o       Set the activity to loop through **inputsTable**

* Find and add two **Assign**activities (necessary variables should be created with the shortcut):

           o       The first one assigns **row(0).ToString**to **valueA**

           o       The second one assigns **row(1).ToString**to **valueB**

                    ▪        These convert the row object values to more usable string values

* Find and add another **Assign** activity that assigns to **row(2)**this value:

           o       **Integer.Parse(valueA) + Integer.Parse(valueB)**

           o       This statement converts the string values to integer values using a Visual Basic method and then adds them together

* Lastly, find and add a **Write Range** activity below and outside the **For Each** activity - this will be writing the manipulated DataTable to a new sheet.

           o       Set the DataTable to **inputsTable**

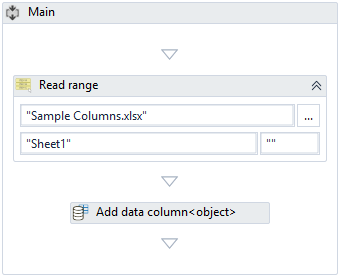
           o       The sheet name should remain as Sheet1

           o       The starting cell should be left blank, as “”

           o       Set the workbook path to a desired path that ends with the file name **Sample Columns - Completed.xlsx**

                   ▪   UiPath will create a new file if this one doesn’t already exist

* This is what the rest of the completed workflow should look like:



**Part C:**

**This part is mostly a matter of using an Excel command for the rows that need adding. It should be completely contained in an Excel Application Scope.**

* Find and add an **Excel Application Scope**activity and add it to the main sequence

o       As usual, set the path of **Sample Columns.xlsx**

o       Set the visibility option on by checking the box

* Find and add a **Read Range**activity

o       The sheet should remain as Sheet1

o       Set the output to a newly created DataTable called **inputsTable**

Count how many rows there are so the formulas can be applied to the proper section of the sheet.

* Find and add an **Assign**activity below the **Read Range**activity

          o       Assign **inputsTable.Rows.Count** to a newly created generic variable called**rowsCount**

* Find and add a **Write Value**activity, it should be set to:

          o       Write on Sheet1

          o       Write in the range from “C1:C” + rowsCount

                   ▪        This sets the range of rows in Column C to write the formula in

          o       Write the value “=SUM(A1,B1)”

                   ▪        In Excel, this value will automatically iterate through the descending rows

* This is what the final workflow should look like:

