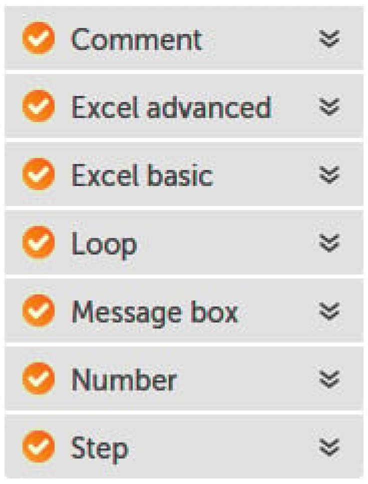
Lab 11: Automating Excel

In this lab, we will cover the following:

- Opening, closing, and saving Excel workbooks
- Reading and writing data within Excel worksheets
- Working with data in Excel
- Running macros

We will be taking the same approach as the previous lab, with a number of walk-throughs exploring the different Excel actions. As before, we will be building our bots with multiple steps to help us keep things structured. We will be using the following packages:



The basic Excel package works with files with the `.xlsx` extension only, whereas the advanced package works with Excel files with extensions of `.xlsx` , `.xls` , `.xlsm` , `.xlsb` , and `.csv` . As we progress through this lab, we will be using both packages, and you will learn about the different actions available from each one. We will start by creating a bot that opens, saves, and closes an Excel workbook. For this, we will be using the Excel basic package and the sample file that is part of the GitHub repository that was downloaded previously.

Technical requirements

In order to install the Automation Anywhere Bot agent, the following requirements are necessary:

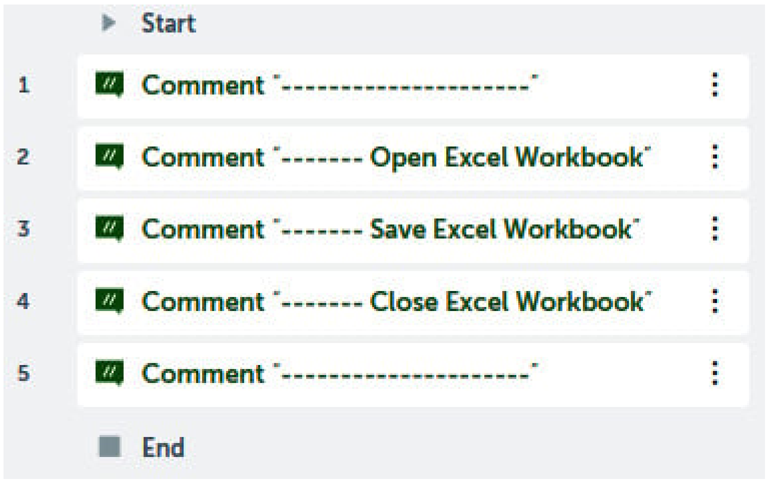
- Web browser: Google Chrome
- You must have completed registration for Automation Anywhere Community Edition
- You must have logged on successfully to Automation Anywhere Community Edition
- A successfully registered local device
- Successfully downloaded sample data from GitHub

Opening, closing, and saving Excel workbooks

The first thing we need to learn about Excel is how to open, close, and save workbooks. So, we'll start off with a simple walk-through. You will open an Excel workbook, save it, and close it. For this walk-through, we will use the sample workbook `Chapter11_Catalog.xlsx` from the GitHub repository.

Let's start this walk-through by executing the following steps:

1. Log into **Control Room**.
2. Create a new bot and call it `Lab 11 – Excel Basic` in the folder `\Bot\`.
3. As always, we begin by adding some comments to use as a template for our bot. Add a new **Comment** action as `"---------------------"` on line **1** and click on **Save**.
4. Add a new **Comment** action as `"------- Open Excel Worksheet"` on line **2** and click on **Save**.
5. Add a new **Comment** action as `"------- Save Excel Worksheet"` on line **3** and click on **Save**.
6. Add a new **Comment** action as `"------- Close Excel Worksheet"` on line **4** and click on **Save**.
7. Add a new **Comment** action as `"---------------------"` on line **5** and click on **Save**. Our initial development interface should look like this:



Start

1 Comment "---------------------"

2 Comment "------- Open Excel Workbook"

3 Comment "------- Save Excel Workbook"

4 Comment "------- Close Excel Workbook"

5 Comment "---------------------"

End

8. To open the workbook, drag the **Excel basic: Open** action just below line **2**.
9. Set the following properties for the **Excel basic: Open** action on line **3**:

   Session name: `xl_data`

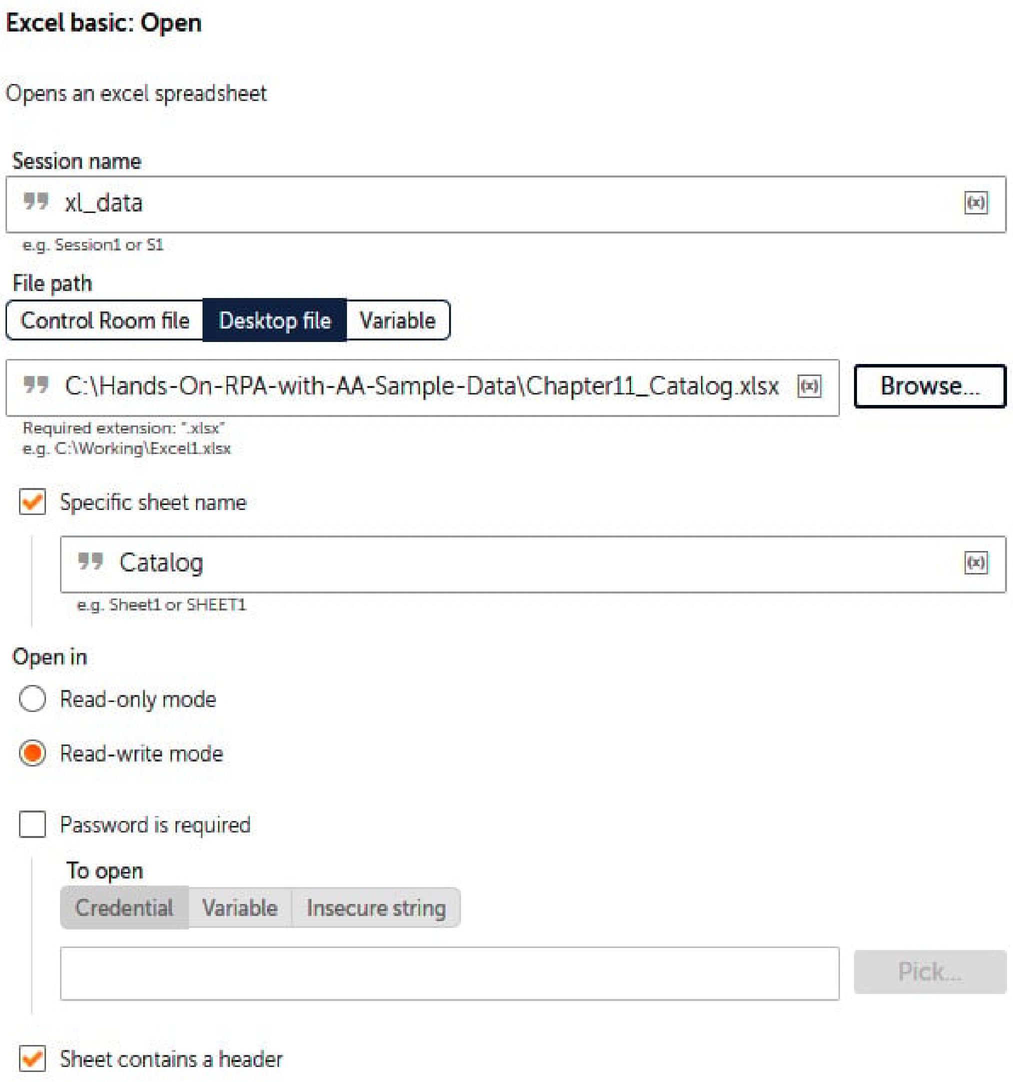
   File path: **Desktop file** - `C:\Hands-On-RPA-with-AA-Sample-Data\Chapter11_Catalog.xlsx`

   Specific sheet name: *Checked* - `Catalog`

   Open in: **Read-write mode**

   Sheet contains a header: *Checked*

   The properties should look like this:


Excel basic: Open

Opens an excel spreadsheet

Session name: xl_data
e.g. Session1 or S1

File path: Control Room file [✓ Desktop file] Variable

C:\Hands-On-RPA-with-AA-Sample-Data\Chapter11_Catalog.xlsx

Required extension: ".xlsx"
e.g. C:\Working\Excel1.xlsx

[✓] Specific sheet name

Catalog
e.g. Sheet1 or SHEET1

Open in
Read-only mode [✓ Read-write mode]

[ ] Password is required

To open: [✓ Credential] Variable Insecure string

[✓] Sheet contains a header

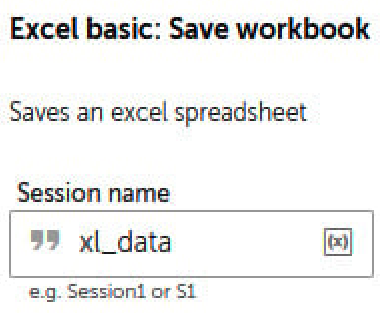
10. Click on **Save**.

11. To save the workbook, drag the **Excel basic: Save workbook** action just below line **4**.

12. Set the following properties for the **Excel basic: Save workbook** action on line **5**:

    Session name: `xl_data`

    The properties should look like this:



Excel basic: Save workbook

Saves an excel spreadsheet

Session name: xl_data

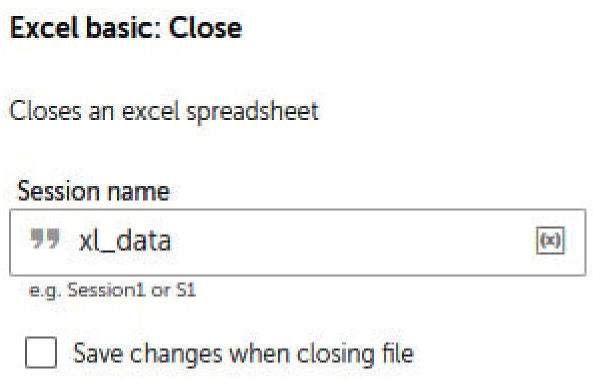
e.g. Session1 or S1

13. Click on **Save**.
14. To close the workbook, drag the **Excel basic: Close** action just below line **6**.
15. Set the following properties for the **Excel basic: Close** action on line **7**:

    Session name: `xl_data`

    Save changes when closing file: *Unchecked*

    The properties should look like this:



Excel basic: Close

Closes an excel spreadsheet

Session name: xl_data

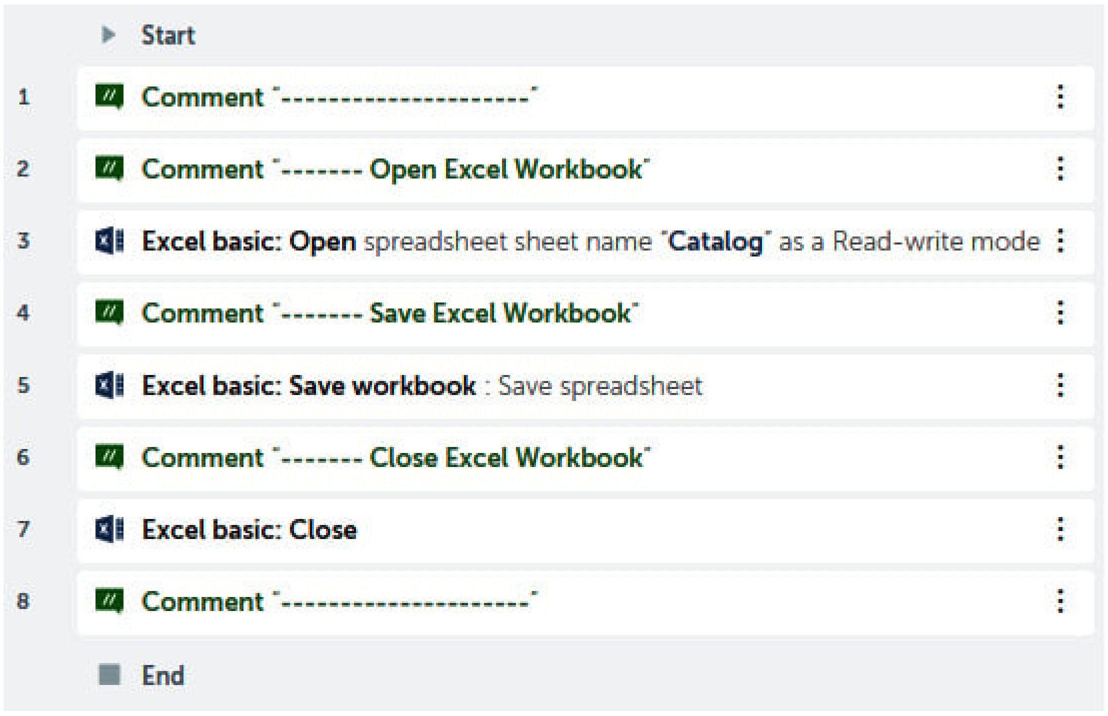
e.g. Session1 or S1

☐ Save changes when closing file

16. Click on **Save**.

You could choose to save the workbook when closing it by checking the **Save changes when closing file** box. In some cases, you may need to save your workbook intermittently without closing. In these instances, the **Excel basic: Save workbook** action should be used.

We have looked at the basics---opening, saving, and closing an Excel workbook. Your development window should look like this:



```
    Start
1   Comment "--------------------"
2   Comment "------- Open Excel Workbook"
3   Excel basic: Open spreadsheet sheet name "Catalog" as a Read-write mode
4   Comment "------- Save Excel Workbook"
5   Excel basic: Save workbook : Save spreadsheet
6   Comment "------- Close Excel Workbook"
7   Excel basic: Close
8   Comment "--------------------"
    End
```

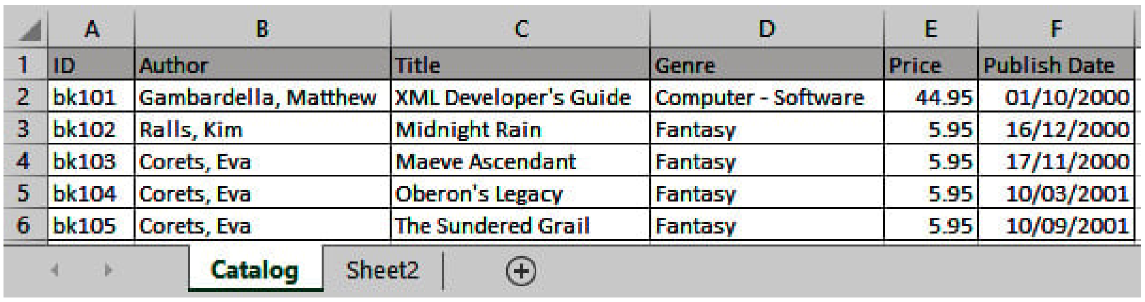
Now that we can open a workbook, the next stage is to learn how to read data from an Excel worksheet.

Reading and writing data within Excel worksheets

Working with data using Excel is a key part of most business roles. Having the ability to automate tasks using Excel can free substantial time from our daily routine. The following walk-throughs will look at reading datasets from Excel followed by writing data to Excel. Like the previous lab, we will be adding new steps at each stage to make our bot more structured.

Reading from Excel worksheets

Data in Excel is usually presented as a table, which means it consists of a fixed set of columns with each row as a record or transaction. The sample data file has a worksheet named `Catalog`. The dataset looks like this:



	A	B	C	D	E	F
1	ID	Author	Title	Genre	Price	Publish Date
2	bk101	Gambardella, Matthew	XML Developer's Guide	Computer - Software	44.95	01/10/2000
3	bk102	Ralls, Kim	Midnight Rain	Fantasy	5.95	16/12/2000
4	bk103	Corets, Eva	Maeve Ascendant	Fantasy	5.95	17/11/2000
5	bk104	Corets, Eva	Oberon's Legacy	Fantasy	5.95	10/03/2001
6	bk105	Corets, Eva	The Sundered Grail	Fantasy	5.95	10/09/2001

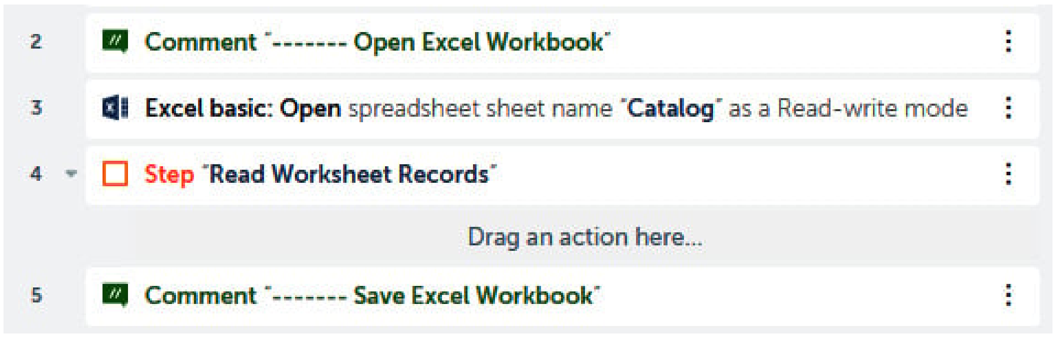
Catalog
Sheet2

In this walk-through, we want our bot to read each record and show the `Title` and `Price` in a message box. To read the record, we need a `Record` type variable, which will be named `recBook`. Once a record is assigned to a `Record` type variable, it is accessed using an index. The index is zero-based so the first column is identified as index `0`. In this case, the `Title` is the third column, giving it an index of `2`, and the `Price` is the fifth column, making it have an index value of `4`. We will continue to build on our bot that we have already created.

Let's start this walk-through by executing the following steps:

1. Create a `Record` type variable called `recBook`.
2. Add a **Step** action just below line **3.**
3. Set the **Title** property of the **Step** on line **4** as `Read Worksheet Records`.
4. Click on **Save**.

   Your development interface should look like this:



2 Comment "------- Open Excel Workbook"

3 Excel basic: Open spreadsheet sheet name "Catalog" as a Read-write mode

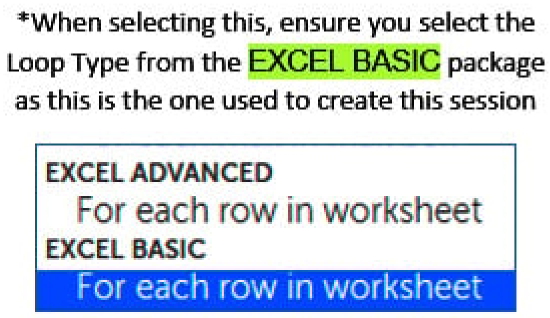
4 Step "Read Worksheet Records"

Drag an action here...

5 Comment "------- Save Excel Workbook"

5. The `Catalog` worksheet has already been specified when we opened the workbook. To loop through each row from the dataset, add a **Loop** action just below line number **4** ensuring it remains within the **Step** on line **4**.
6. Set the following properties for the **Loop** action on line **5**:

   Loop Type*: **For each row in worksheet**



*When selecting this, ensure you select the Loop Type from the EXCEL BASIC package as this is the one used to create this session

EXCEL ADVANCED

For each row in worksheet

EXCEL BASIC

For each row in worksheet

```
**Session name**: `xl_data`

**Loop through**: **All rows**

**Assign the current value to this variable**: **recBook - Record**

The properties should look like this:
```

```
![](./images/Figure_11.10_B15646.jpg)
```

7. Click on **Save**.

8. That's all there is to it. All that's left is to create a message box to display the book `Title` and `Price`. Add a **Message box** action just below line **5**, ensuring it remains within the **Loop** on line **5**.

9. Set the following properties for the **Message box** action on line **6**:

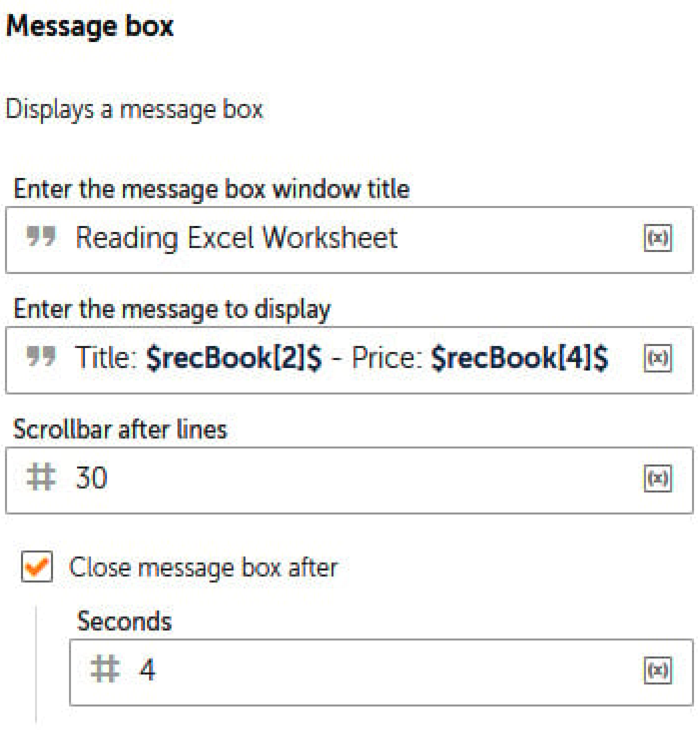
   Enter the message box window title: `Reading Excel Worksheet`

   Enter the message to display: `$recBook[2]$ - Price: $recBook[4]$`

   Close message box after: *Checked*

   Seconds: `4`

   The properties should look like this:



Message box

Displays a message box

Enter the message box window title: Reading Excel Worksheet

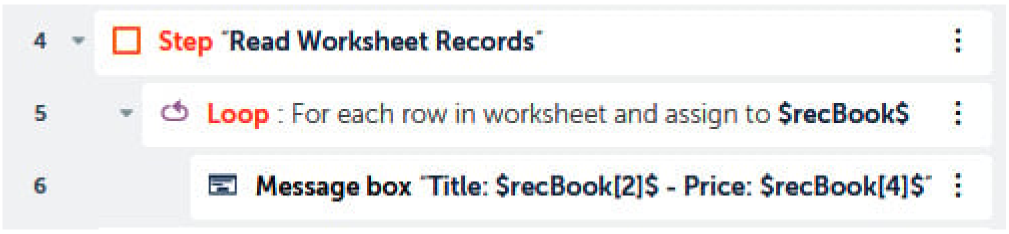
Enter the message to display: Title: $recBook[2]$ - Price: $recBook[4]$

Scrollbar after lines: 30

✓ Close message box after

Seconds: 4

10. Click on **Save**, the development interface for this section should look like this:



4 Step "Read Worksheet Records"

5 Loop : For each row in worksheet and assign to $recBook$

6 Message box "Title: $recBook[2]$ - Price: $recBook[4]$"

You are now ready to test your bot. Go ahead and run it. It should read each record from the `Catalog` worksheet, displaying the `Title` and `Price`. Now that we can read from an Excel worksheet, naturally, we also need to know how to write data back to a spreadsheet. In the next section, we will learn how to insert values into our Excel worksheet. As inserting values into Excel can be a common task we do manually, knowing how to automate this can be a valuable skill to have.

Writing to Excel worksheets

In this walk-through, you will learn how to write to Excel. We will continue building on our current bot. We know our dataset consists of a list of books. The task of our bot is to write the total number of books at the bottom of the list. To achieve this, we will need two `Number` type variables: one to store the number of records, and another to store the row to write the results to. As there is a header row in our dataset, the first available blank row will be `number of records + 2`. For each `Number` type variable, a `String` will also be needed as output can only be a string variable. As we are looping through the dataset, we will add an **Increment** action to get the number of records.

Let's start this walk-through by executing the following steps:

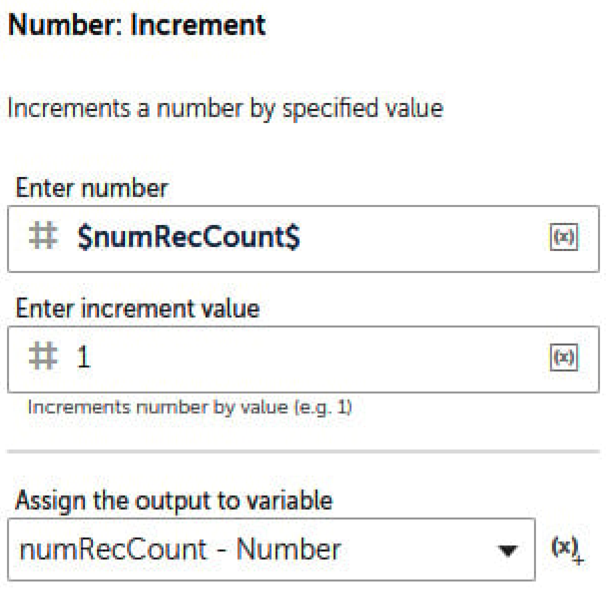
1. Create two `Number` type variables called `numRecCount` and `numResultRow`.
2. Create two `String` type variables called `strRecCount` and `strResultRow`.
3. To use an increment to get the number of records, add the **Number: Increment** action just below line **5**, ensuring it remains within the **Loop** on line **5**.
4. Set the following properties for the **Number: Increment** action on line **6**:

   Enter number: `$numRecCount$`

   Enter increment value: `1`

   Assign the output to variable: **numRecCount - Number**

   The properties should look like this:



Number: Increment

Increments a number by specified value

Enter number

$numRecCount$

Enter increment value

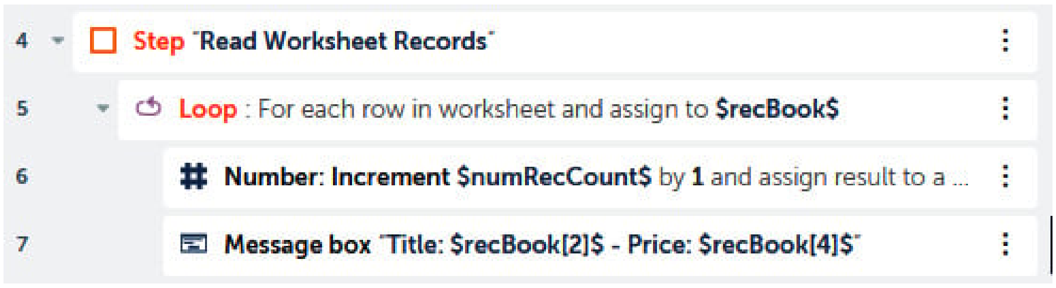
1

Increments number by value (e.g. 1)

Assign the output to variable

numRecCount - Number

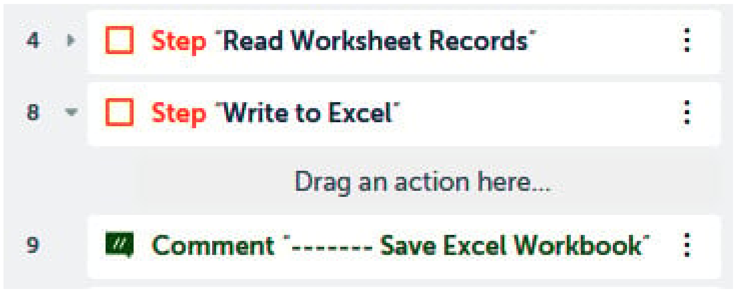
5. Click on **Save**. The development interface for the **Loop** should look like this:


4 Step "Read Worksheet Records"
5 Loop : For each row in worksheet and assign to $recBook$
6 Number: Increment $numRecCount$ by 1 and assign result to a ...
7 Message box "Title: $recBook[2]$ - Price: $recBook[4]$"

6. Add a **Step** action just below line **7**, ensuring it is not indented within the **Loop** on line **5**.
7. Set the **Title** property of this **Step** on line **8** as `Write to Excel`.

   Click on **Save** and collapse the **Step** on line **4**.

   Your development interface should look like this:

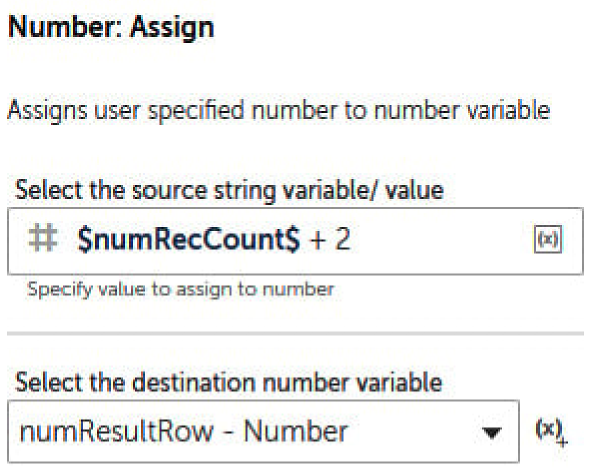

4 Step "Read Worksheet Records"
8 Step "Write to Excel"
Drag an action here...
9 Comment "------- Save Excel Workbook"

8. To get the output row, we know it's `number of records + 2`; add the **Number: Assign** action just below line **8**, ensuring it remains within the **Step** on line **8**.
9. Set the following properties for the **Number: Assign** action on line **9**:

   Select the source string variable/ value: `$numRecCount$ + 2`

   Select the destination number variable: **numResultRow - Number**

   The properties should look like this:


Number: Assign

Assigns user specified number to number variable

Select the source string variable/ value

$numRecCount$ + 2

Specify value to assign to number

Select the destination number variable

numResultRow - Number

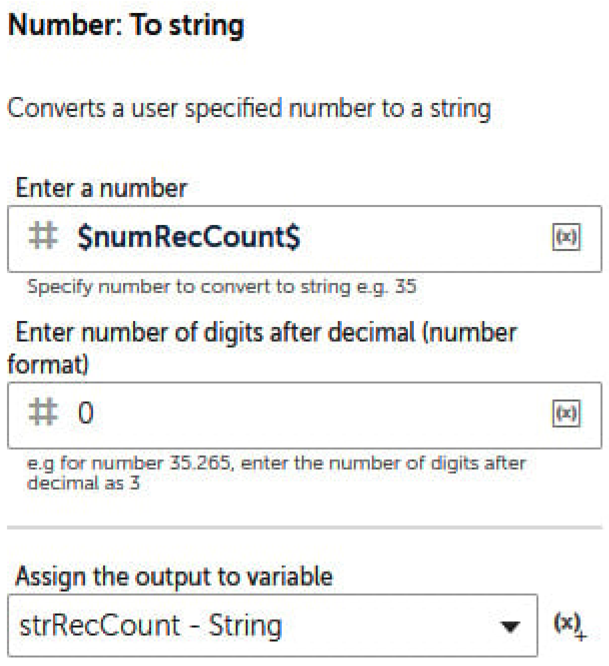
10. Click on **Save**.
11. Now we have the output row and the records count, they both need to be converted to a `String` variable type. Add the **Number: To string** action just below line **9**, ensuring it remains within the **Step** on line **8**.
12. Set the following properties for the **Number: To string** action on line **10**:

    Enter a number: `$numRecCount$`

    Enter number of digits after decimal: `0`

    Assign the output to variable: **strRecCount - String**

    The properties should look like this:


Number: To string

Converts a user specified number to a string

Enter a number

$numRecCount$

Specify number to convert to string e.g. 35

Enter number of digits after decimal (number format)

0

e.g for number 35.265, enter the number of digits after decimal as 3

Assign the output to variable

strRecCount - String

13. Click on **Save**.

14. Do the same for the output row by adding another **Number: To string** action just below line **10**, ensuring it remains within the **Step** on line **8**.

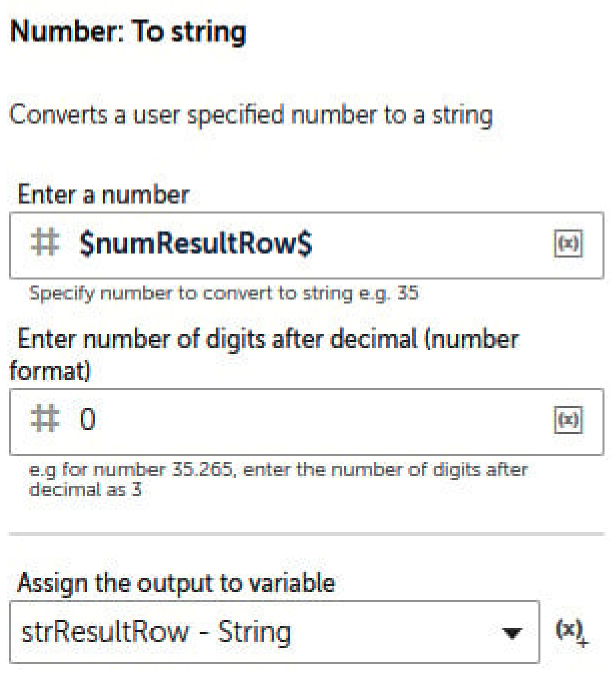
15. Set the following properties for the **Number: To string** action on line **11**:

    Enter a number: `$numResultRow$`

    Enter number of digits after decimal: `0`

    Assign the output to variable: **strResultRow - String**

    The properties should look like this:



Number: To string

Converts a user specified number to a string

Enter a number: $numResultRow$

Specify number to convert to string e.g. 35

Enter number of digits after decimal (number format): 0

e.g for number 35.265, enter the number of digits after decimal as 3

Assign the output to variable: strResultRow - String

16. Click on **Save**.

17. To write this back to our worksheet, let's put the text `Total:` in column `A` at the end of our list. Add the **Excel basic: Set cell** action just below line **11**, ensuring it remains within the **Step** on line **8**.

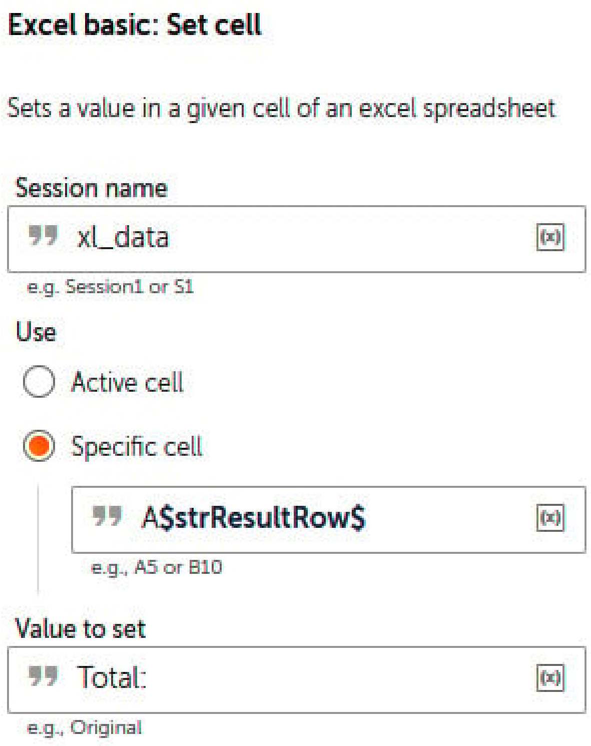
18. Set the following properties for the **Excel basic: Set cell** action on line **12**:

    Session name: `xl_data`

    Use: **Specific cell** -- `A$strResultRow$`

    Value to set: `Total:`

    The properties should look like this:


Excel basic: Set cell

Sets a value in a given cell of an excel spreadsheet

Session name: xl_data

e.g. Session1 or S1

Use

( ) Active cell

[✓ Specific cell]

A$strResultRow$

e.g., A5 or B10

Value to set: Total:

e.g., Original

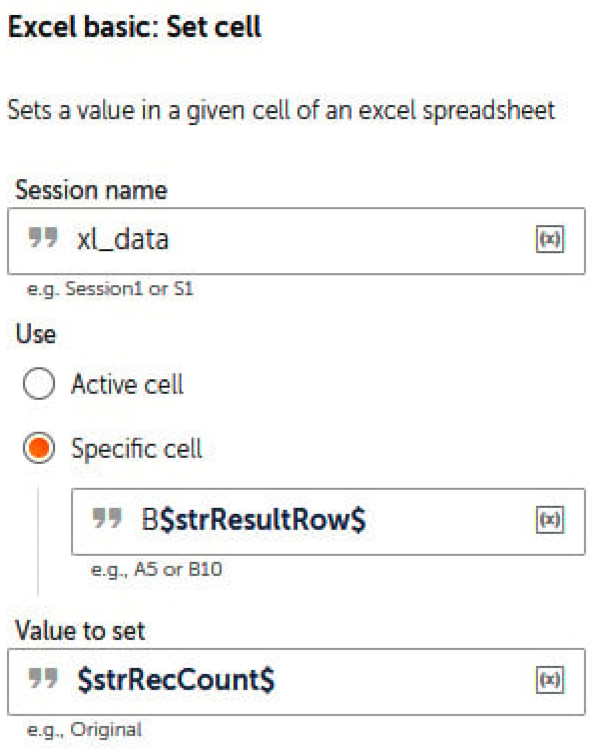
19. Click on **Save**.
20. Finally, to write the record count to our worksheet in column `B`, add the **Excel basic: Set cell** action just below line **12**, ensuring it remains within the **Step** on line **8**.
21. Set the following properties for the **Excel basic: Set cell** action on line **13**:

    Session name: `xl_data`

    Use: **Specific cell** -- `B$strResultRow$`

    Value to set: `$strRecCount$`

    The properties should look like this:



Excel basic: Set cell

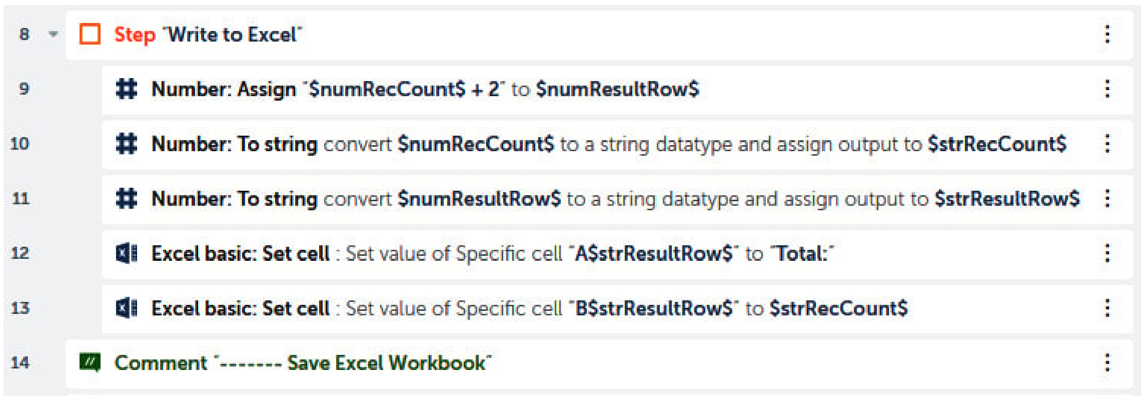
Sets a value in a given cell of an excel spreadsheet

Session name: xl_data
e.g. Session1 or S1

Use
( ) Active cell
(●) Specific cell
B$strResultRow$
e.g., A5 or B10

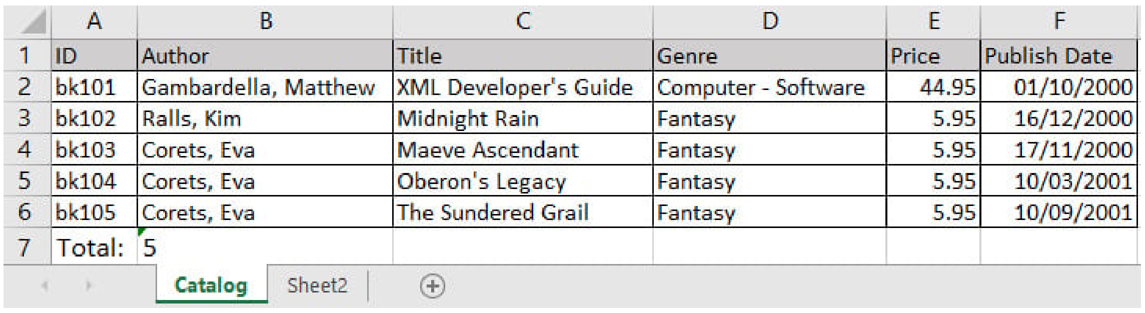
Value to set: $strRecCount$
e.g., Original

22. Click on **Save**. The development interface should look like this:



```
8  Step "Write to Excel"
9      Number: Assign "$numRecCount$ + 2" to $numResultRow$
10     Number: To string convert $numRecCount$ to a string datatype and assign output to $strRecCount$
11     Number: To string convert $numResultRow$ to a string datatype and assign output to $strResultRow$
12     Excel basic: Set cell : Set value of Specific cell "A$strResultRow$" to "Total:"
13     Excel basic: Set cell : Set value of Specific cell "B$strResultRow$" to $strRecCount$
14 Comment "------- Save Excel Workbook"
```

All done, which is great. You can go ahead and run your bot. Your bot should display the book title and price for every record. Once the bot has completed, open the sample workbook. It should now also contain a total count, as shown in the following screenshot:

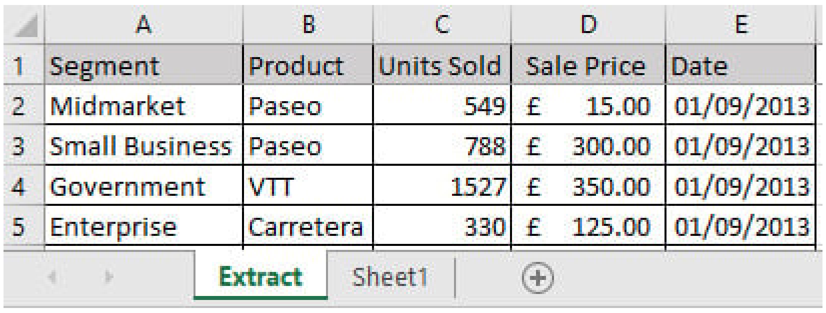


	A	B	C	D	E	F
1	ID	Author	Title	Genre	Price	Publish Date
2	bk101	Gambardella, Matthew	XML Developer's Guide	Computer - Software	44.95	01/10/2000
3	bk102	Ralls, Kim	Midnight Rain	Fantasy	5.95	16/12/2000
4	bk103	Corets, Eva	Maeve Ascendant	Fantasy	5.95	17/11/2000
5	bk104	Corets, Eva	Oberon's Legacy	Fantasy	5.95	10/03/2001
6	bk105	Corets, Eva	The Sundered Grail	Fantasy	5.95	10/09/2001
7	Total:	5				

You should now be comfortable with reading and writing to Excel. This is just the start of how you can automate Excel using Automation Anywhere. As mentioned earlier, there are two Excel packages available with Automation Anywhere. We have so far been looking at the Excel basic package. In the next section, we'll move onto the Excel advanced package and discover what other Excel tasks can be automated. We'll start by looking at manipulating data within worksheets, such as by inserting columns and formulas as well as sorting data.

Working with data in Excel

In the previous section, we looked at the basics of Excel automation, such as opening, reading, writing, saving, and closing Excel workbooks. In this section, we will explore the Automation Anywhere Excel advanced package. The walk-through will involve building a bot that performs a specific role. The bot will replace a manual task that involves updating an Excel spreadsheet. The file used is available as part of the GitHub repository `Chapter11_SampleData.xlsm`:



	A	B	C	D	E
1	Segment	Product	Units Sold	Sale Price	Date
2	Midmarket	Paseo	549	£ 15.00	01/09/2013
3	Small Business	Paseo	788	£ 300.00	01/09/2013
4	Government	VTT	1527	£ 350.00	01/09/2013
5	Enterprise	Carretera	330	£ 125.00	01/09/2013

This file contains a data table of sales achieved by a fictitious organization. In order to automate the file, we perform the following steps:

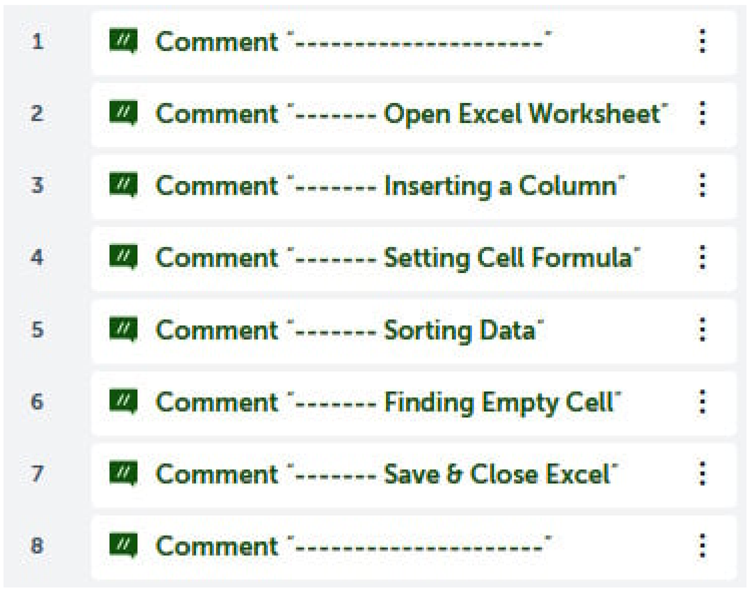
1. Open the Excel file `Chapter11_SampleData.xlsm`.
2. Insert a new column after column `4` and call it `Amount` on the worksheet `Extract`.
3. Add a formula in the new column to calculate the sales amount (`Units Sold` * `Sale Price`).
4. Sort the data table by the `Segment` column in ascending order.
5. In the last row, add an `Amount` column and sum the `total value` of orders.
6. Save and close the workbook.

Let's start this walk-through by executing the following steps:

1. Log into **Control Room**.

2. Create a new bot and call it `Lab 11 - Excel Advanced` in the `\Bot\` folder.

3. Starting with the comments, add a new **Comment** action as `"---------------------"` on line **1** and click on **Save**.
4. Add a new **Comment** action as `"------- Open Excel Worksheet"` on line **2** and click on **Save**.
5. Add a new **Comment** action as `"------- Inserting a Column"` on line **3** and click on **Save**.
6. Add a new **Comment** action as `"------- Setting Cell Formula"` on line **4** and click on **Save**.
7. Add a new **Comment** action as `"------- Sorting Data"` on line **5** and click on **Save**.
8. Add a new **Comment** action as `"------- Finding Empty Cell"` on line **6** and click on **Save**.
9. Add a new **Comment** action as `"------- Save & Close Excel"` on line **7** and click on **Save**.
10. Add a new **Comment** action as `"---------------------"` on line **8** and click on **Save**. Our development interface should look like this:



```
1 Comment "---------------------"
2 Comment "------- Open Excel Worksheet"
3 Comment "------- Inserting a Column"
4 Comment "------- Setting Cell Formula"
5 Comment "------- Sorting Data"
6 Comment "------- Finding Empty Cell"
7 Comment "------- Save & Close Excel"
8 Comment "---------------------"
```

11. To open the workbook, drag the **Excel advanced: Open** action just below line **2**.
12. Set the following properties for **Excel advanced: Open** action on line **3**:

    Session name: `xl_data`

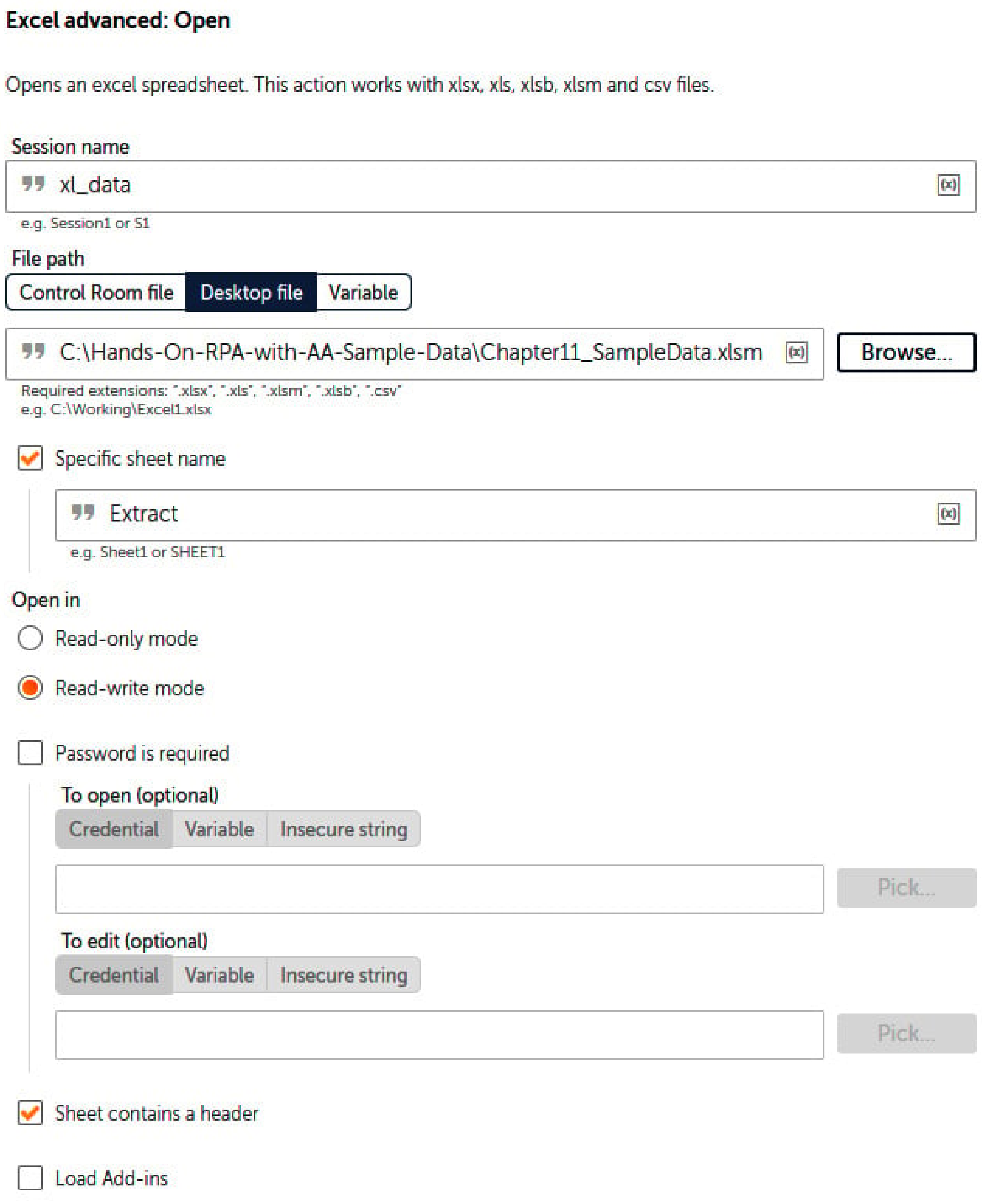
    File path: **Desktop file** - `C:\Hands-On-RPA-with-AA-Sample-Data\Chapter11_SampleData.xlsm`

    Specific sheet name: *Checked* - `Extract`

    Open in: **Read-write mode**

    Sheet contains a header: *Checked*

    The properties should look like this:


Excel advanced: Open

Opens an excel spreadsheet. This action works with xlsx, xls, xlsb, xlsm and csv files.

Session name: xl_data
e.g. Session1 or S1

File path: Control Room file [✓ Desktop file] Variable
C:\Hands-On-RPA-with-AA-Sample-Data\Chapter11_SampleData.xlsm
Browse...
Required extensions: ".xlsx", ".xls", ".xlsm", ".xlsb", ".csv"
e.g. C:\Working\Excel1.xlsx

☑ Specific sheet name: Extract
e.g. Sheet1 or SHEET1

Open in
○ Read-only mode
● Read-write mode

☐ Password is required
To open (optional): Credential | Variable | Insecure string — Pick...
To edit (optional): Credential | Variable | Insecure string — Pick...

☑ Sheet contains a header

☐ Load Add-ins

13. Click on **Save**.

The bot will now have the file open, ready to start processing. The first task is to insert a column.

Inserting a column

We need to add a new column and call it `Amount`. This will be column `E`, which is the sixth column in our data table. The data table in the sample worksheet is named `Data`:

1. To insert a new column as column `5` called `Amount`, drag the **Excel advanced: Insert table column** action just below line **4**.

2. Set the following properties for the **Excel advanced: Insert table column** action on line **5**:

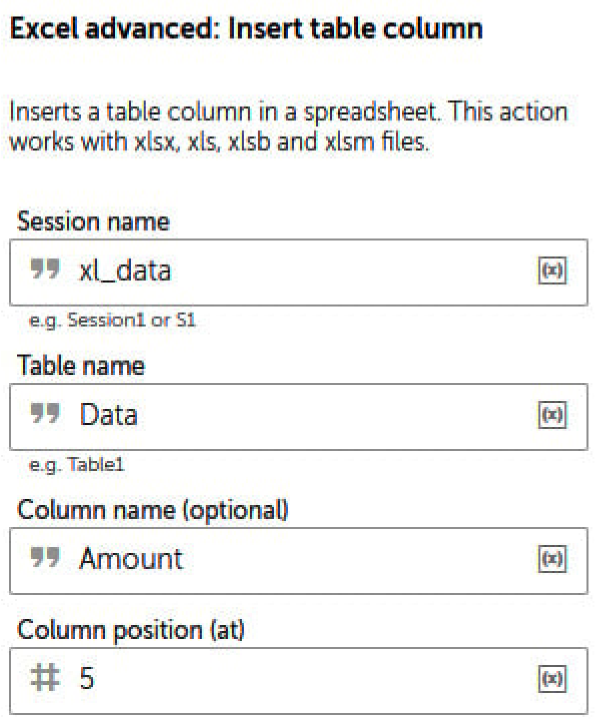
Session name: `xl_data`

Table name: `Data`

Column name: `Amount`

Column position: `5`

The properties should look like this:



Excel advanced: Insert table column

Inserts a table column in a spreadsheet. This action works with xlsx, xls, xlsb and xlsm files.

Session name

xl_data

e.g. Session1 or S1

Table name

Data

e.g. Table1

Column name (optional)

Amount

Column position (at)

5

3. Click on **Save**.

The instruction to add the new column is now complete. We'll move on to the next task -- setting a formula in the new column.

Setting a cell formula

Now that there is a column called `Amount` (column `E`), it needs to be populated. The value of this column is to be set as `Units Sold` * `Sale Price`. We know the first row with data is `2`, making the initial cell `E2`:

1. To set the formula in cell `E2`, add the **Excel advanced: Set cell formula** action just below line **6**.

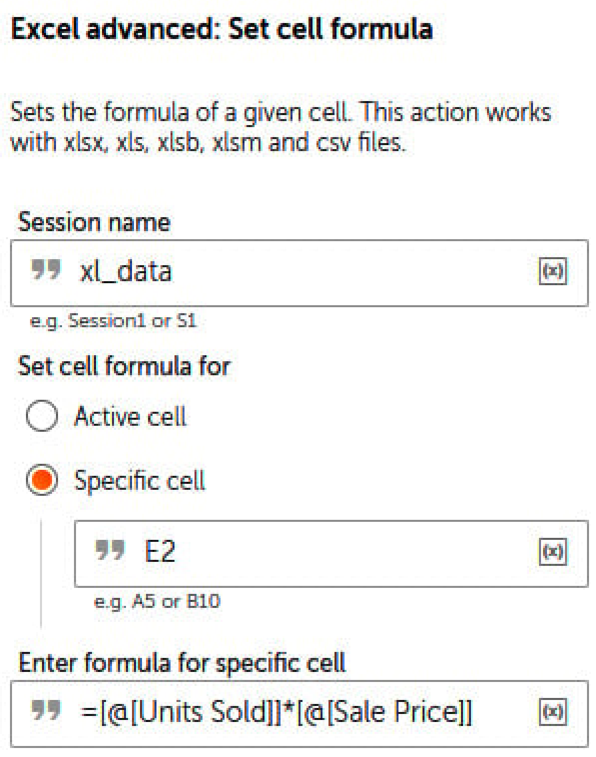
2. Set the following properties for the **Excel advanced: Set cell formula** action on line **7**:

   Session name: `xl_data`

   Set cell formula for: **Specific cell** -- `E2`

   Enter formula for specific cell: `=[@[Units Sold]]*[@[Sale Price]]`

   The properties should look like this:


Excel advanced: Set cell formula

Sets the formula of a given cell. This action works with xlsx, xls, xlsb, xlsm and csv files.

Session name: xl_data

e.g. Session1 or S1

Set cell formula for: Active cell [✓ Specific cell]

E2

e.g. A5 or B10

Enter formula for specific cell: =[@[Units Sold]]*[@[Sale Price]]

3. Click on **Save**.

This formula will populate itself for each record within the data table. As we are automating a specific manual task, the next step in our criteria is to sort the data table.

Sorting data

A part of the requirement is to sort the data table by the `Segment` column (column `A`). This should be sorted in ascending order:

1. To sort the data table, add the **Excel advanced: Sort table** action just below line **8**.

2. Set the following properties for the **Excel advanced: Sort table** action on line **9**:

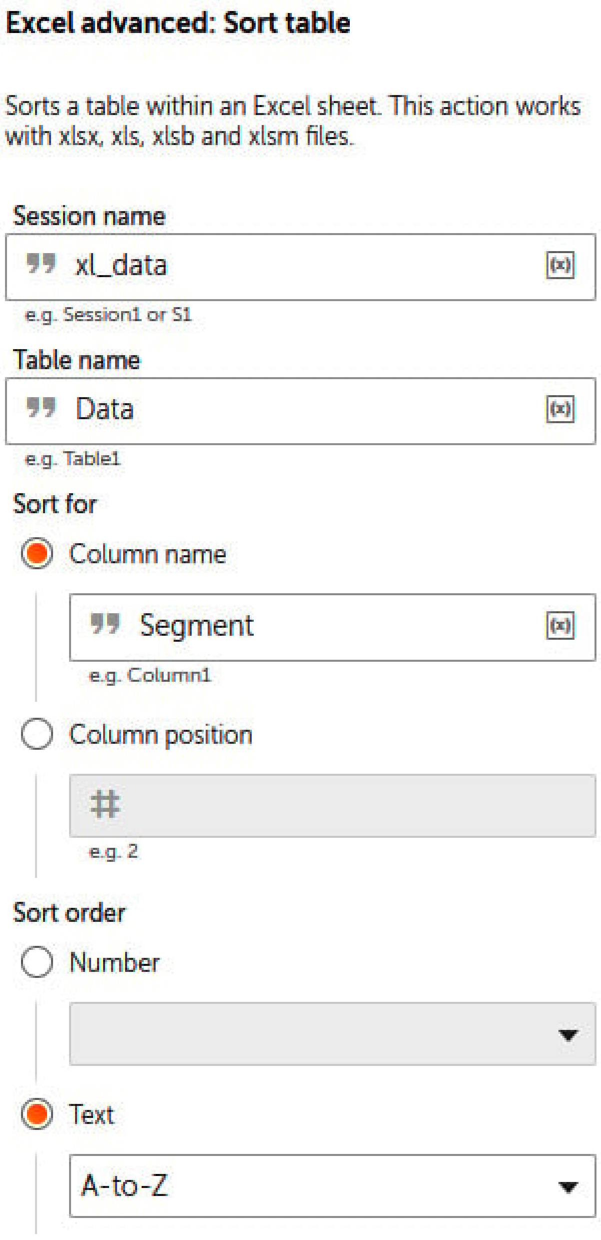
   Session name: `xl_data`

   Table name: `Data`

   Sort for: **Column name** - `Segment`

   Sort order: **Text** - **A-to-Z**

   The properties should look like this:



Excel advanced: Sort table

Sorts a table within an Excel sheet. This action works with xlsx, xls, xlsb and xlsm files.

Session name: xl_data
e.g. Session1 or S1

Table name: Data
e.g. Table1

Sort for
[✓ Column name]: Segment
e.g. Column1
Column position:
e.g. 2

Sort order
Number:
[✓ Text]: A-to-Z

3. Click on **Save**.

The data table should now be sorted as required. The next step is to add the total sum of sales at the end of the data table.

Finding empty cells

To add the sum of the `Amount` column, we need to identify which cell to enter this sum formula into. We can get this by identifying the first empty row, which would be the target row for our sum. We already know the column is `E` as we created this earlier. A `String` type variable will be needed to store the cell address:

1. Create a `String` type variable and call it `strTotalCell`.

2. To get the first empty cell in column `E`, add the **Excel advanced: Find next empty cell** action just below line **10**.

3. Set the following properties for the **Excel advanced: Find next empty cell** action on line **11**:

   Session name: `xl_data`

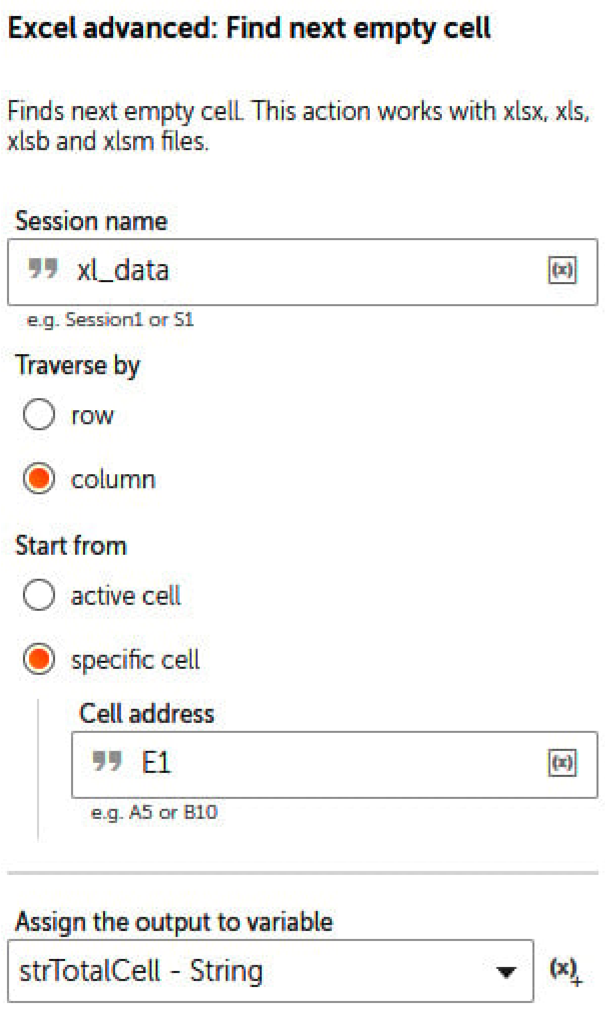
   Transverse by: **column**

   Start from: **specific cell**

   Cell address: `E1`

   Assign the output to variable: **strTotalCell - String**

   The properties should look like this:



Excel advanced: Find next empty cell

Finds next empty cell. This action works with xlsx, xls, xlsb and xlsm files.

Session name: xl_data
e.g. Session1 or S1

Traverse by: row [✓ column]

Start from: active cell [✓ specific cell]

Cell address: E1
e.g. A5 or B10

Assign the output to variable: strTotalCell - String

4. Click on **Save**.

5. Now we have the cell address for the total sales amount, we can add the sum formula by dragging the **Excel advanced: Set cell formula** action just below line **11**.

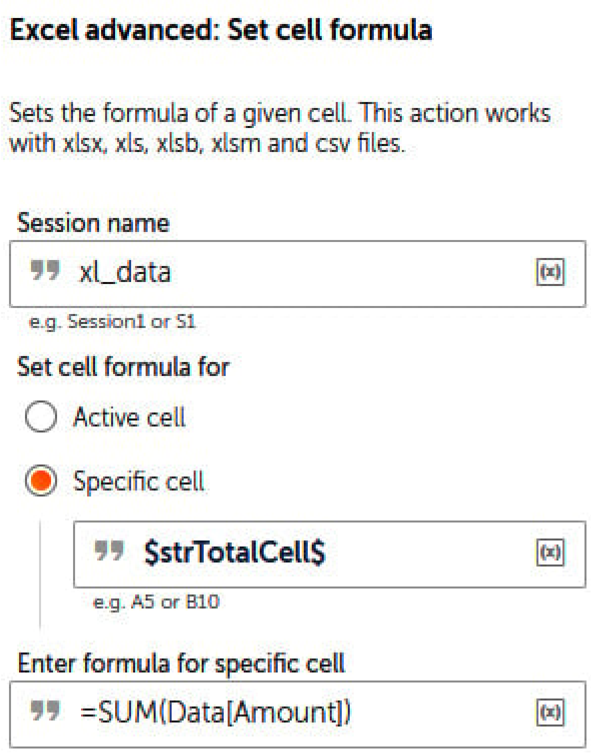
6. Set the following properties for the **Excel advanced: Set cell formula** action on line **12**:

Session name: `xl_data`

Set cell formula for: **Specific cell** -- `$strTotalCell$`

Enter formula for specific cell: `=SUM(Data[Amount])`

The properties should look like this:



Excel advanced: Set cell formula

Sets the formula of a given cell. This action works with xlsx, xls, xlsb, xlsm and csv files.

Session name: xl_data

e.g. Session1 or S1

Set cell formula for

( ) Active cell

[✓ Specific cell]

$strTotalCell$

e.g. A5 or B10

Enter formula for specific cell: =SUM(Data[Amount])

7. Click on **Save**.

Great -- you have now instructed your bot to calculate the total sum of sales and to set this value in the last row of the data table. All that is left to do is to close our workbook.

Saving and closing the workbook

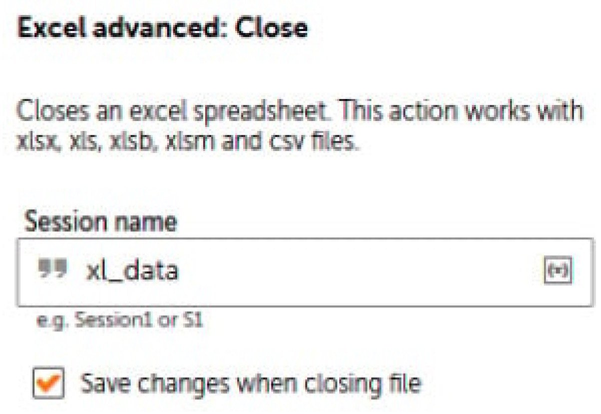
The last task for the bot to perform is to save and close our worksheet. As the workbook was opened using the **Excel Advanced** package, it needs to be closed with it too:

1. To save and close the workbook, drag the **Excel advanced: Close** action just below line **13**.
2. Set the following properties for the **Excel advanced: Close** action on line **14**:

   Session name: `xl_data`

   Save changes when closing file: *Checked*

   The properties should look like this:


Excel advanced: Close

Closes an excel spreadsheet. This action works with xlsx, xls, xlsb, xlsm and csv files.

Session name

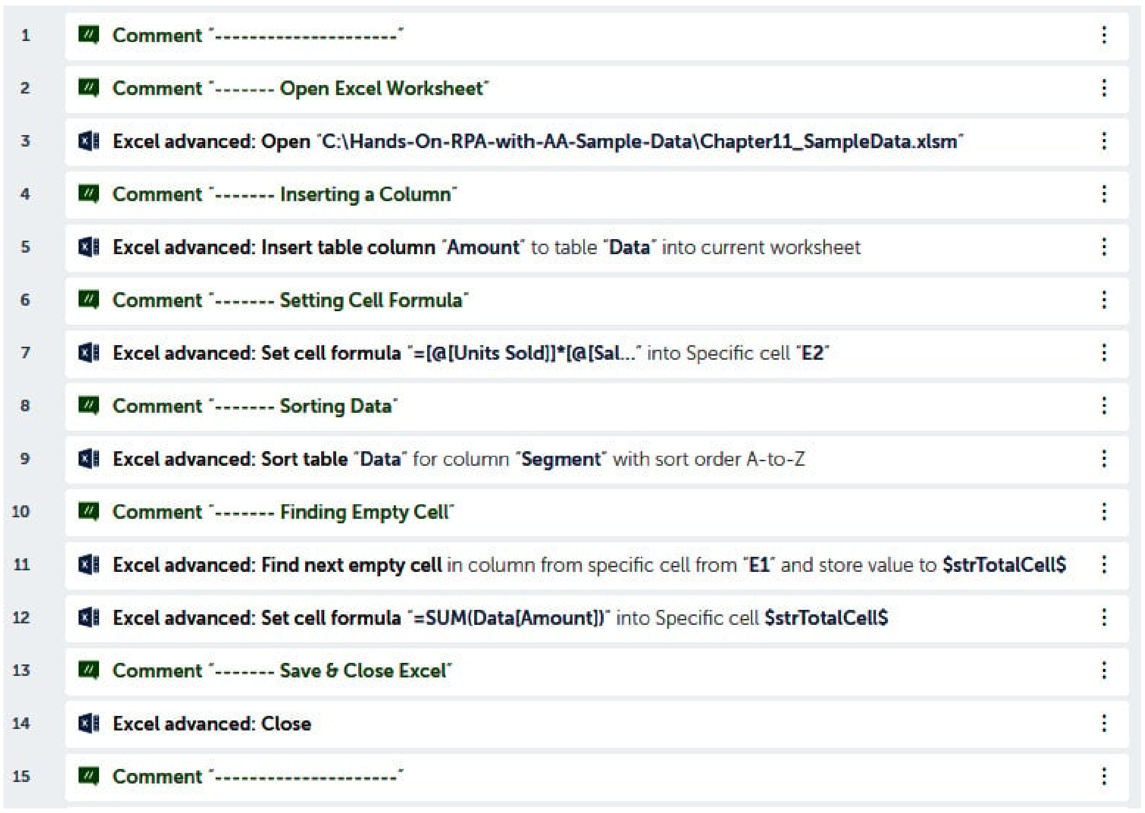
xl_data

e.g. Session1 or S1

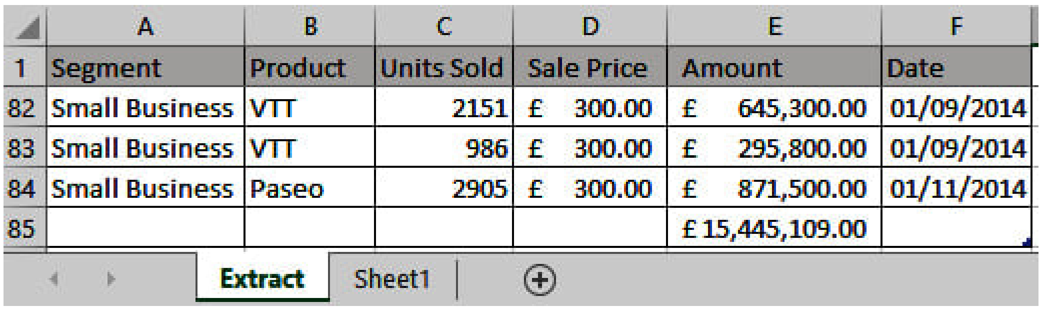
☑ Save changes when closing file

3. Click on **Save**.

Our bot is now complete and should be good to go and perform our task to completion. We have looked at some of the more advanced actions available for Excel. This has included actions such as inserting columns and setting formulas. Your development window should look like this:


```
1   Comment "--------------------"
2   Comment "------- Open Excel Worksheet"
3   Excel advanced: Open "C:\Hands-On-RPA-with-AA-Sample-Data\Chapter11_SampleData.xlsm"
4   Comment "------- Inserting a Column"
5   Excel advanced: Insert table column "Amount" to table "Data" into current worksheet
6   Comment "------- Setting Cell Formula"
7   Excel advanced: Set cell formula "=[@[Units Sold]]*[@[Sal..." into Specific cell "E2"
8   Comment "------- Sorting Data"
9   Excel advanced: Sort table "Data" for column "Segment" with sort order A-to-Z
10  Comment "------- Finding Empty Cell"
11  Excel advanced: Find next empty cell in column from specific cell from "E1" and store value to $strTotalCell$
12  Excel advanced: Set cell formula "=SUM(Data[Amount])" into Specific cell $strTotalCell$
13  Comment "------- Save & Close Excel"
14  Excel advanced: Close
15  Comment "--------------------"
```

That's all done; time to run your bot. Your bot should perform all the tasks seamlessly and edit the sample file by inserting a new column and populating it with a formula. It should also add the sum at the bottom of the data table. Once you have executed the bot, open the sample file. It should look like this:



	A	B	C	D	E	F
1	Segment	Product	Units Sold	Sale Price	Amount	Date
82	Small Business	VTT	2151	£ 300.00	£ 645,300.00	01/09/2014
83	Small Business	VTT	986	£ 300.00	£ 295,800.00	01/09/2014
84	Small Business	Paseo	2905	£ 300.00	£ 871,500.00	01/11/2014
85					£ 15,445,109.00	

Extract
Sheet1

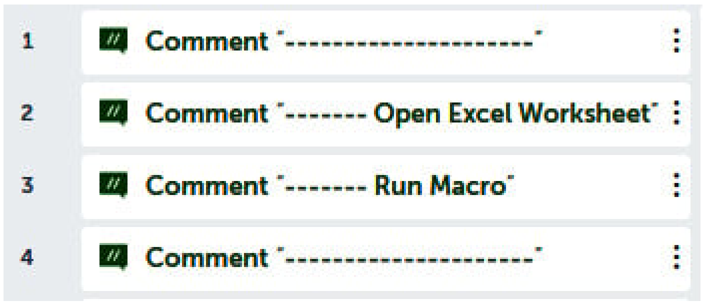
Hopefully, you will now be confident with performing numerous Excel tasks using Automation Anywhere. A key feature of Excel is having the ability to create and run macros. As macros are used widely in business, Automation Anywhere can also be used to trigger Excel macros. In the next section, you will learn how to run macros. This will be ideal if you are automating a manual process that involves running Excel macros.

Running macros

In this section, we will learn about running Excel macros. The sample file has a macro called `procFilterSegment`. This macro will filter the `Segment` column by the given argument. In the following walk-through, the bot will open the sample workbook in read-only mode and run this macro. We will send an `Enterprise` argument to the macro. This should result in the bot applying a filter of `Enterprise` to the `Segment` column.

Let's start this walk-through by executing the following steps:

1. Log into **Control Room**.
2. Create a new bot and call it `Chapter11 – Excel Macro` in the `\Bot\` folder.
3. Starting with the comments, add a new **Comment** action as `"---------------------"` on line **1** and click on **Save**.
4. Add a new **Comment** action as `"------- Open Excel Worksheet"` on line **2** and click on **Save**.
5. Add a new **Comment** action as `"------- Run Macro"` on line **3** and click on **Save**.
6. Add a new **Comment** action as `"---------------------"` on line **4** and click on **Save**. Our development interface should look like this:



```
1  Comment "---------------------"
2  Comment "------- Open Excel Worksheet"
3  Comment "------- Run Macro"
4  Comment "---------------------"
```

7. To open the workbook, drag the **Excel advanced: Open** action just below line number **2**.
8. Set the following properties for the **Excel advanced: Open** action on line **3**:

Session name: `xl_data`

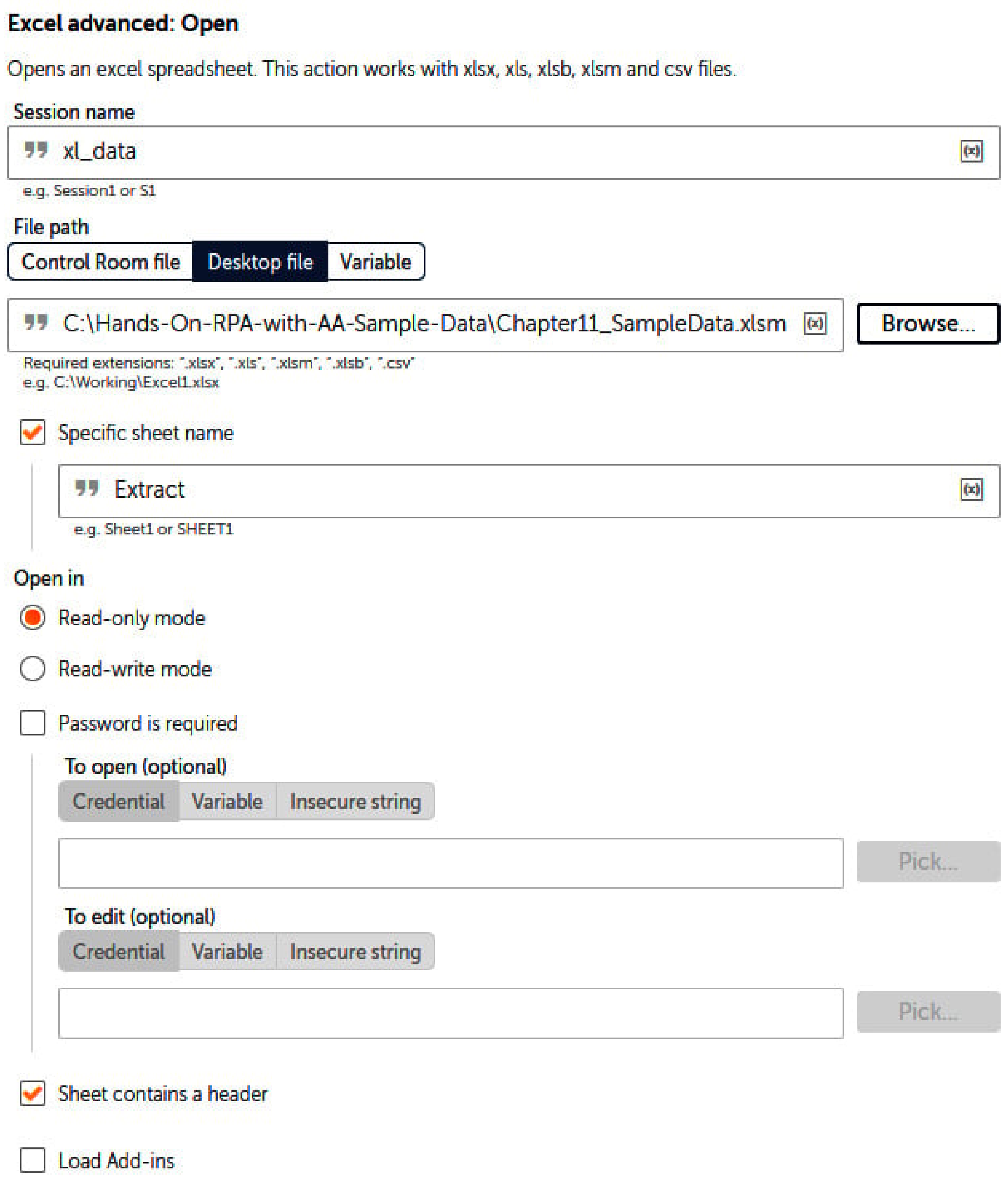
File path: **Desktop file** - `C:\Hands-On-RPA-with-AA-Sample-Data\Chapter11_SampleData.xlsm`

Specific sheet name: *Checked* -- `Extract`

Open in: **Read-only mode**

Sheet contains a header: *Checked*

The properties should look like this:



Excel advanced: Open

Opens an excel spreadsheet. This action works with xlsx, xls, xlsb, xlsm and csv files.

Session name: xl_data

e.g. Session1 or S1

File path: Control Room file | [✓ Desktop file] | Variable

C:\Hands-On-RPA-with-AA-Sample-Data\Chapter11_SampleData.xlsm

Required extensions: ".xlsx", ".xls", ".xlsm", ".xlsb", ".csv"
e.g. C:\Working\Excel1.xlsx

[✓] Specific sheet name: Extract

e.g. Sheet1 or SHEET1

Open in:
[✓ Read-only mode]
Read-write mode

[ ] Password is required

To open (optional): Credential | Variable | Insecure string

To edit (optional): Credential | Variable | Insecure string

[✓] Sheet contains a header

[ ] Load Add-ins

9. Click on **Save**.
10. To run the macro `procFilterSegment` and pass an argument with it, drag the **Excel advanced: Run macro** action just below line **4**.

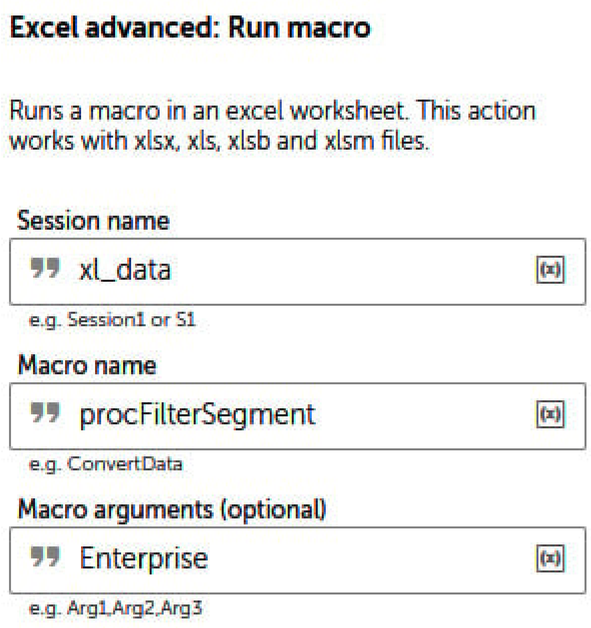
11. Set the following properties for the **Excel advanced: Run macro** action on line **5**:

    Session name: `xl_data`

    Macro name: `procFilterSegment`

    Macro arguments: `Enterprise`

    The properties should look like this:



Excel advanced: Run macro

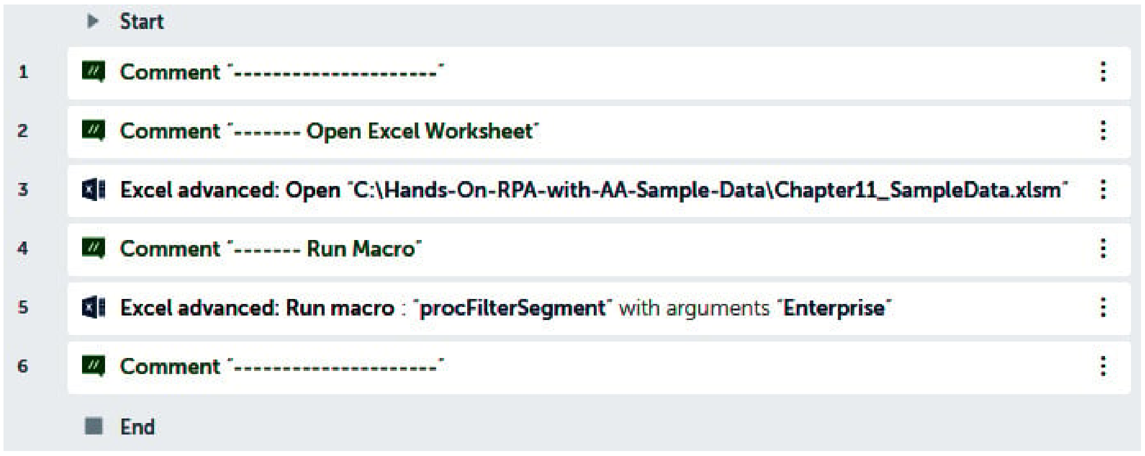
Runs a macro in an excel worksheet. This action works with xlsx, xls, xlsb and xlsm files.

Session name: xl_data
e.g. Session1 or S1

Macro name: procFilterSegment
e.g. ConvertData

Macro arguments (optional): Enterprise
e.g. Arg1,Arg2,Arg3

12. Click on **Save**. The development interface should look like this:



Start
1 Comment "---------------------"
2 Comment "------- Open Excel Worksheet"
3 Excel advanced: Open "C:\Hands-On-RPA-with-AA-Sample-Data\Chapter11_SampleData.xlsm"
4 Comment "------- Run Macro"
5 Excel advanced: Run macro : "procFilterSegment" with arguments "Enterprise"
6 Comment "---------------------"
End

It's testing time again; try running your bot. You will find it passes the argument through to the macro that applies the filter to the `Segment` column. Now you should be confident with using Automation Anywhere to automate your Excel-based manual tasks.

Summary

A wide selection of actions from both the Excel basic and advanced package has been covered in this lab. The walk-throughs will have given you the opportunity to build three individual bots all performing different tasks. You should be confident with opening, closing, and saving Excel documents, as well as reading and writing to them. The skills gained should also include working with data within Excel documents, such as sorting, inserting columns, adding formulas, as well as running macros.

The next lab continues with Microsoft applications. In this case, we will be looking at Microsoft Word. You will learn how to create and edit documents. You will build a role-based bot that will include working with Excel to generate Word documents.