

UNIT-III

RPA Tool Introduction & Basics: Introduction to RPA Tool, The User Interface, Variables, Managing Variables, Naming Best Practices, The Variables Panel, Generic Value Variables, Text Variables, True or False Variables, Number Variables, Array Variables, Date and Time Variables, Data Table Variables, Managing Arguments, Naming Best Practices, The Arguments Panel, Using Arguments, About Imported Namespaces, Importing New Namespaces, Control Flow, Control Flow Introduction, If Else Statements, Loops, Advanced Control Flow, Sequences, Flowcharts, About Control Flow, Control Flow Activities, The Assign Activity, The Delay Activity, The Do While Activity, The If Activity, The Switch Activity, The While Activity, The For Each Activity, The Break Activity, Data Manipulation, Data Manipulation Introduction, Scalar variables, collections and Tables, Text Manipulation, Data Manipulation, Gathering and Assembling Data.

Introduction to RPA Tool:

Robotic Process Automation (RPA) is a technology that allows you to automate repetitive tasks by creating bots that can interact with your computer and software applications. RPA tools are software platforms that you can use to create and manage bots.

RPA bots are software programs that are designed to mimic the actions of a human user. They can be used to automate a wide variety of tasks, such as:

- Extracting data from websites and spreadsheets.
- Processing invoices and other documents.
- Generating reports.
- Managing customer interactions.
- Updating databases.

RPA bots can be used to improve efficiency, accuracy, and productivity. They can also help you to free up your employees to focus on more strategic tasks.

Benefits of RPA

The benefits of RPA are many and varied. Some of the most common benefits include:

- **Improved efficiency:** RPA bots can automate repetitive tasks, which can free up your employees to focus on more strategic tasks.
- **Increased accuracy:** RPA bots can help to reduce errors in data entry and processing.
- **Improved productivity:** RPA bots can help to improve productivity by automating tasks that would otherwise be done manually.
- **Reduced costs:** RPA bots can help to reduce costs by eliminating the need for manual labor.
- **Improved compliance:** RPA bots can help to improve compliance by ensuring that tasks are performed consistently and accurately.

Types of RPA

There are three main types of RPA:

- **Desktop RPA:** This type of RPA automates tasks that are performed on a single computer.
- **Web RPA:** This type of RPA automates tasks that are performed on websites.
- **Hybrid RPA:** This type of RPA combines desktop and web RPA.

RPA Tools

There are many different RPA tools available, each with its own strengths and weaknesses. Some of the most popular RPA tools include:

UiPath

- Opens in a new window
- [w en.wikipedia.org](https://en.wikipedia.org)
- UiPath RPA tool logo

Automation Anywhere

- Opens in a new window
- www.automationanywhere.com
- Automation Anywhere RPA tool logo

Blue Prism

- Opens in a new window
- [Eblog.expertrec.com](https://blog.expertrec.com)
- Blue Prism RPA tool logo

WorkFusion



- [Opens in a new window](#)
- [G www.gartner.com](https://www.gartner.com)
- WorkFusion RPA tool logo

Pegasystems



- [Opens in a new window](#)
- [G www.gartner.com](https://www.gartner.com)
- Pegasystems RPA tool logo

How to Get Started with RPA

If you are interested in getting started with RPA, there are a few things you need to do:

1. Choose an RPA tool.
2. Identify the tasks that you want to automate.
3. Create a bot to automate the tasks.
4. Test and deploy the bot.
5. Monitor the bot and make changes as needed.

Choosing an RPA Tool

The first step in getting started with RPA is to choose an RPA tool. There are many different RPA tools available, so it is important to choose one that is right for your needs. Some factors to consider when choosing an RPA tool include:

- The features and capabilities of the tool.
- The cost of the tool.
- The level of support that is offered by the vendor.
- The ease of use of the tool.

Identifying the Tasks to Automate

The next step is to identify the tasks that you want to automate. Not all tasks are suitable for automation, so it is important to carefully consider which tasks will provide the most benefit. Some factors to consider when identifying tasks to automate include:

- The frequency of the task.
- The complexity of the task.
- The accuracy requirements of the task.
- The cost of automating the task.

Creating a Bot

Once you have identified the tasks that you want to automate, you can create a bot to automate the tasks. This involves using the RPA tool to create a program that will perform the tasks. The RPA tool will typically provide a graphical user interface (GUI) that makes it easy to create bots.

Testing and Deploying the Bot

Once you have created a bot, you need to test it to make sure that it works correctly. You should also deploy the bot to production so that it can start automating the tasks.

Monitoring the Bot

Once the bot is in production, you need to monitor it to make sure that it is working correctly. You should also make changes to the bot as needed.

RPA is a powerful technology that can be used to automate a wide variety of tasks. If you are looking for a way to improve efficiency, accuracy, and productivity, then RPA is a good option

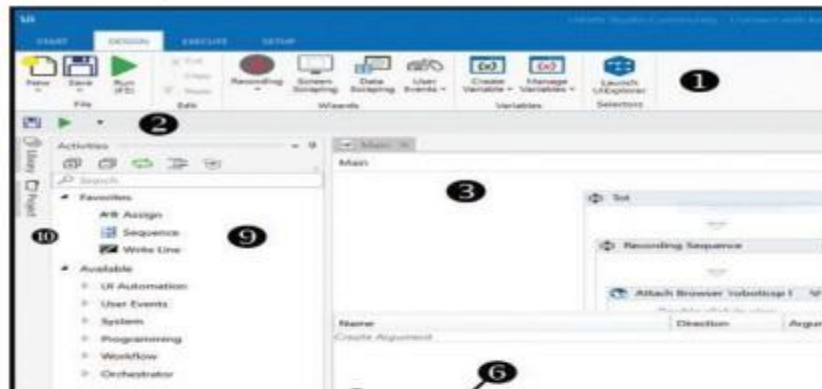
The User Interface

When you first open UiPath Studio, you are directed to the page.

You can either open an old project or create a new one.

Let us say we are making a new project.

We click on Blank and name it. We will then be directed to a screen, which will display the following:



1. The Ribbon
2. Quick Access Toolbar
3. Designer panel
4. Properties panel
5. Outline panel
6. Arguments panel
7. Variable panel
8. Import panel
9. Activity panel
10. Library panel
11. Project panel
12. Output panel

1. The Ribbon

- This panel located at the top of the user interface and consists of four tabs:

1. **START**: This is used to start new projects or to open projects previously made.
2. **DESIGN**: This is to create new sequences, Flowcharts, or state machines, or to manage variables:
3. **EXECUTE**: This is used to run projects or to stop them, and also to debug Projects
4. **SETUP**: This panel is for deployment and configuration options; it has three tools available:
 - Publish**: This is used to publish a project or create a shortcut for it and schedule tasks
 - Setup Extensions**: This is used to install extensions for Chrome, Firefox, Java, and Silver light
 - Reset Settings**: This is used to reset all settings to defaults:

2. The Quick Access Toolbar

- This panel gives the user a shortcut to the most used commands.
- One can also add new commands to this panel. This is located above the Ribbon on the user interface.
- The Quick Access Toolbar has been circled in the following screenshot and is indicated by the arrow: can be moved above or below the Ribbon. By default, there are two buttons available, **Save** and **Run**, which are also available in the **DESIGN** tab of the Ribbon.

3. Designer panel

- This is the panel where one defines the steps and activities of the projects.
- It is where a developer does most of the things to record activities or manually drop activities on the canvas.
- In UiPath, this is equivalent to the code windows of Microsoft Visual Studio. When we develop a Robot, this is the window where we will be organizing various activities in a flow or chain to accomplish a task.
- The project a user makes is clearly displayed on the Designer panel and the user has the option of making any changes to it.

4. Properties panel



- The panel located on the right-hand side of the user interface is for viewing the properties of the activities and for making any changes, if required.
- You need to select an activity first and then go to the **Properties** panel to view or change any of its properties

5. Activities panel

- Located on the left-hand side of the user interface, this panel contains all the activities that can be used in building the project.
- The activities can easily be used in making a project by simply dragging and dropping the required activity into the required location in the Designer panel.

6. Project panel

- With the **Project** panel, you can view the details of your current project and open it in a **Windows Explorer** window.
- It is located on the extreme left-hand side of the design panel, below the **Library** panel

7. Outline panel

- As the name suggests, this panel gives a basic outline of the project.
- The activities that make up the workflow are visible in this panel.
- Using this, you may see a high-level outline of the project and you can drill down to see deeper.
- This panel is especially helpful of large automation projects, where one may otherwise have a tough time going through it

8. Output panel

- This panel displays the output of the *log message* or *writes line* activities. It also displays the output during the debugging process.
- This panel also shows errors, warnings, information, and traces of the executed project. It is very helpful during debugging. The desired level of detail can be changed in **Execute | Options | Log activities**



9. Library panel

- With this panel, you can reuse automation snippets. It is located on the extreme left-hand side of the Designer panel:

10. Variable panel

- This allows the user to create variables and make changes to them. This is located below the Designer panel.
- In UiPath Studio, variables are used to store multiple types of data ranging from words, numbers, arrays, dates, times, and timetables.
- As the name suggests, the value of the variable can be changed.
- An important point to note is that variables can only be created if there is an activity in the Designer panel.
- To create new variables, you can go to the **DESIGN** tab on the Ribbon and click on **create variable**, and then choose the type of variable. Otherwise, one can simply go to the Variable panel located below the Designer panel and create a variable.

Variables

Variables are used to store data in RPA. They can be used to store text, numbers, dates, and other types of data.

Why are variables important in RPA?

Variables are important in RPA because they allow you to store data that can be used by the bot. This data can be used to automate tasks, make decisions, and interact with other applications.

For example, you could use a variable to store the name of a file that the bot needs to process. The bot could then use this variable to open the file and extract the data that it needs.

How to manage variables in RPA?

You can manage variables in the variables panel of the RPA tool. The variables panel allows you to create new variables, delete existing variables, and modify the values of existing variables.

- To create a new variable, click on the "New Variable" button in the variables panel. Enter a name for the variable and select the type of data that the variable will store.
- To delete an existing variable, select the variable and click on the "Delete" button.
- To modify the value of an existing variable, double-click on the variable and enter the new value.

Naming best practices for variables in RPA

When naming variables, it is important to follow some best practices. Here are a few tips:

- Use descriptive names that make it clear what the variable is used for.
- Avoid using abbreviations or acronyms.
- Do not use spaces or special characters in the name.

Here are some examples of good variable names:

- customerName
- orderNumber
- dueDate

Here are some examples of bad variable names:

- cName
- o#
- Date

Generic value variables

Generic value variables are the most basic type of variable in RPA. They can store any type of data, including text, numbers, dates, and Boolean values. Generic value variables are often used to store temporary data that is used by the bot.

- Text variables: Text variables are used to store text data. They can be used to store the names of files, folders, and websites. Text variables are often used to store data that is entered by the user or retrieved from a website.
- True or false variables: True or false variables are used to store Boolean values, which can be either true or false. True or false variables are often used to store the results of logical tests. For example, you could use a true or false variable to store the result of a test to see if a file exists.
- Number variables: Number variables are used to store numeric data. Number variables are often used to store the results of calculations. For example, you could use a number variable to store the sum of two numbers.
- Array variables: Array variables are used to store a list of values. Array variables are often used to store data that is collected from a website or application. For example, you could use an array variable to store the names of all of the files in a folder.
- Date and time variables: Date and time variables are used to store date and time data. Date and time variables are often used to store the start and end dates of a workflow. For example, you could use a date and time variable to store the start date of a promotion.
- Data table variables: Data table variables are used to store a table of data. Data table variables are often used to store data that is collected from a spreadsheet or database. For example, you could use a data table variable to store the results of a query from a database.

Argument

- While variables pass data from one activity to another in a project, arguments are used for passing data from one project to another.
- Like variables, they can be of various types-String, Integer, Boolean, Array, Generic, and so on.
- Since arguments are used to transfer data between different workflows, they also have an added property of *direction*. There are four types of direction:

In

Out

In/Out

Property

These depend on whether we are giving or receiving data to or from another workflow

Arguments

Arguments are used to pass data to and from activities. Arguments are typically defined in the arguments panel of the RPA tool.

Managing Arguments

You can manage arguments in the arguments panel of the RPA tool. The arguments panel allows you to create new arguments, delete existing arguments, and modify the values of existing arguments.

Naming Best Practices

When naming arguments, it is important to follow some best practices. Here are a few tips:

Use descriptive names that make it clear what the argument is used for.

Avoid using abbreviations or acronyms.

Do not use spaces or special characters in the name.

The Arguments Panel

The arguments panel is where you can define arguments for activities. The arguments panel is typically divided into two sections:

the input arguments section and the output arguments section.

Input arguments are arguments that are passed to the activity.

Output arguments are arguments that are returned by the activity.

Using Arguments

To use an argument, you need to specify the name of the argument in the activity's configuration. For example, if you have an activity that has an argument called "filename", you would specify the filename argument in the activity's configuration like this:

filename = "myfile.txt"

Here are some examples of how arguments can be used in RPA:

- You could use an argument to pass the name of a file to an activity that opens the file.
- You could use an argument to pass the results of a calculation to an activity that displays the results.
- You could use an argument to pass the value of a checkbox to an activity that updates a database.

By using arguments, you can make your RPA bots more flexible and reusable.

Imported Namespaces

In RPA, namespaces are used to organize activities. Namespaces are typically defined in the project settings.

When you import a namespace, you are adding the activities in that namespace to the project. This allows you to use the activities in the namespace in your bots.

Importing New Namespaces

To import a new namespace, you need to specify the namespace in the project settings. You can do this by opening the project settings and clicking on the "Namespaces" tab. In the "Namespaces" tab, you can add a new namespace by clicking on the "Add" button and entering the name of the namespace.

Here are some examples of namespaces that are commonly imported in RPA:

UiPath.Activities: This namespace contains the activities that are used to interact with the operating system, applications, and other resources.

UiPath.UIAutomation: This namespace contains the activities that are used to interact with the user interface of applications.

UiPath.Excel: This namespace contains the activities that are used to interact with Excel spreadsheets.

UiPath.Database: This namespace contains the activities that are used to interact with databases.

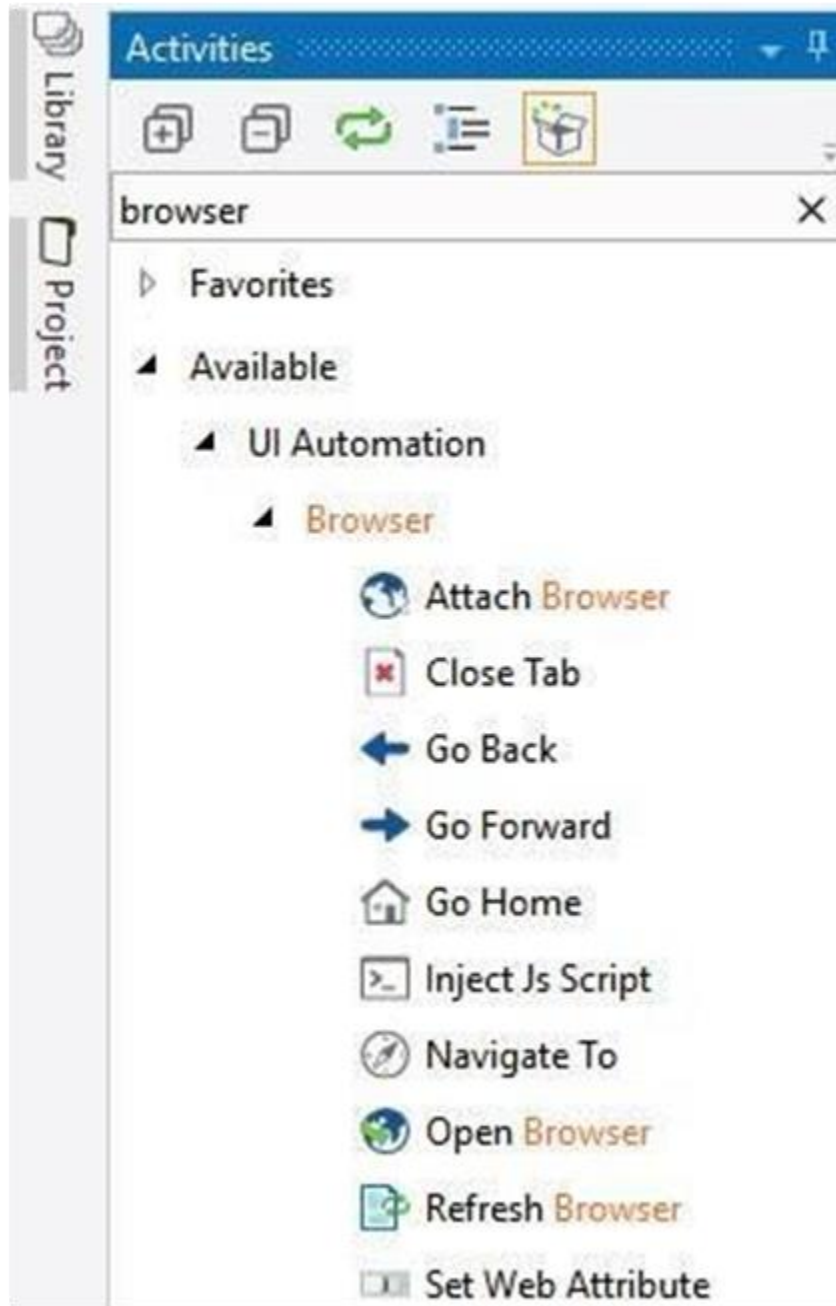
By importing the namespaces that you need, you can make your RPA bots more powerful and versatile.

Here are some of the benefits of importing namespaces in RPA:

- It allows you to use the activities in the namespace in your bots.
- It makes your bots more organized and easier to maintain.
- It allows you to reuse activities from other bots.
- It allows you to keep your bots up-to-date with the latest activities.

Activities

- Activity represents the unit of an action. Each activity performs some action. When these activities combine together, it becomes a process.
- Every activity resides on the Activities panel of the main Designer panel. You can search for a particular activity and use it in your project.
- For example, when we search for browser, all the browser activities will appear in the Activities panel, as shown in the following screenshot:



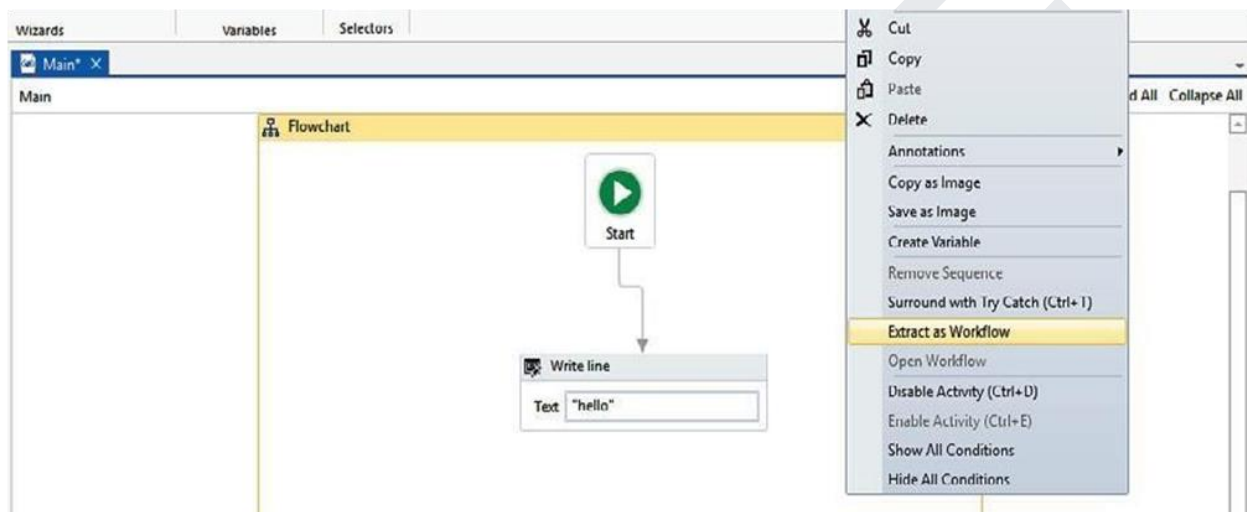
Using activities with workflows

We have seen how we can easily search for a particular activity. Now, let us see how to use them in a workflow:

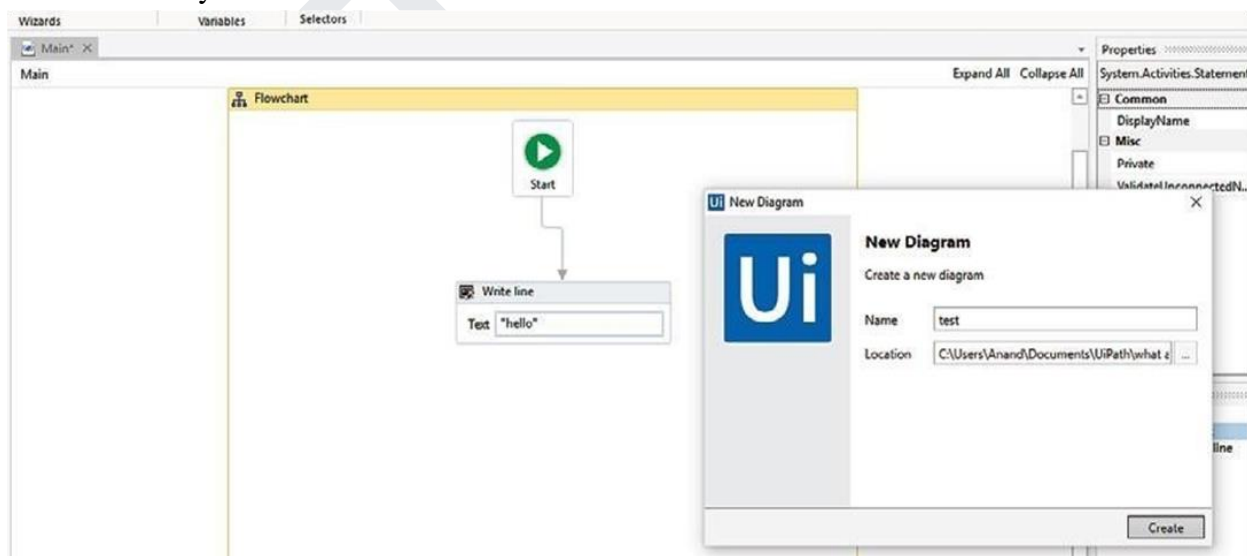
1. Search for Flowchart in the same way that we have searched for the browser activities in the Activities panel search bar. Drag and drop the Flowchart activity inside the Designer panel.
2. The Flowchart appears in the Designer panel and we have a given Start node. The Start node specifies where the execution begins.

3. We are ready to use different activities in our Flowchart. You can use any activity/activities inside the Flowchart. For the sake of simplicity, let us just use a Write line activity.
4. Drag and drop the Write line activity inside the Flowchart. Set its text property by providing a string value. Connect this Write line activity with the Start node by right-clicking on the Write line activity and selecting Set as Start Node.

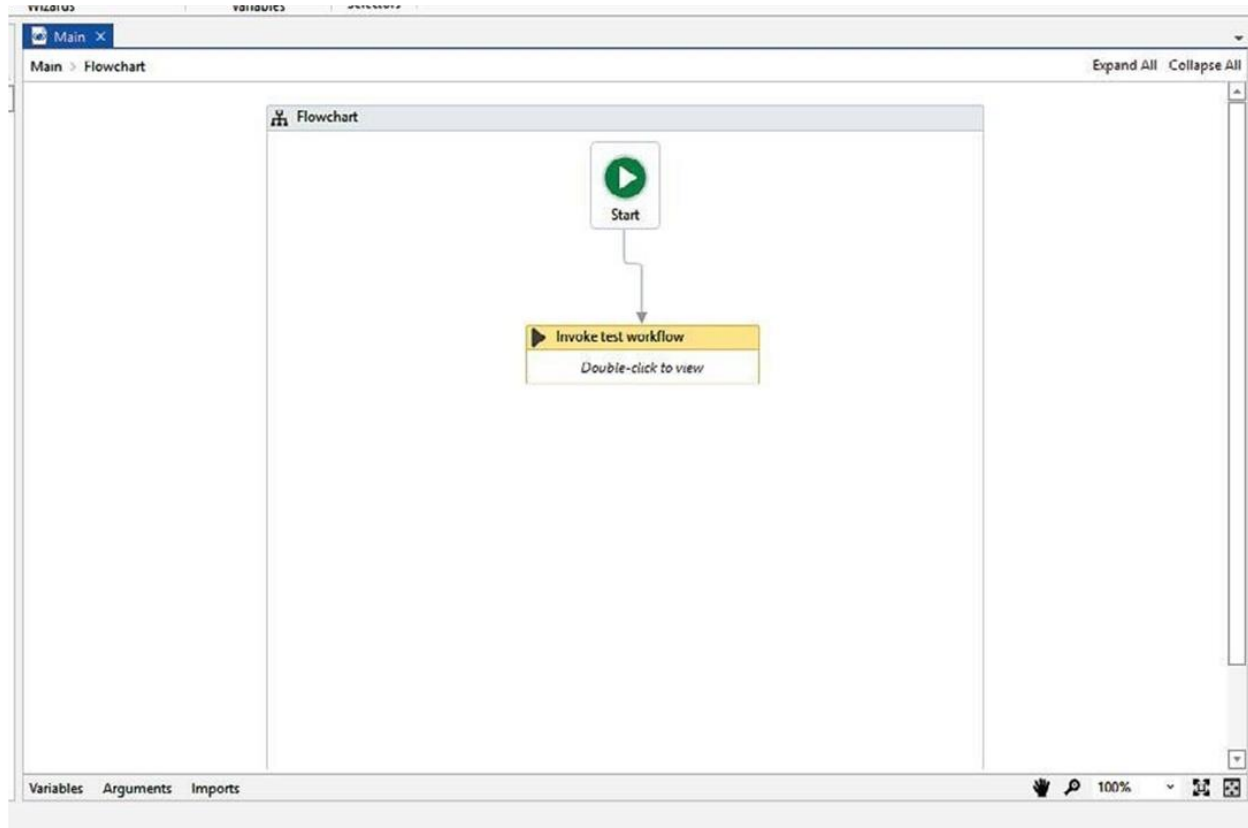
Creating different workflows and combining them into a logical Sequence will enhance our code quality, maintainability, reliability, and readability. Right-click on the main Designer panel and choose Extract as Workflow:



A window will pop up asking for the name. Give it a meaningful name and click on Create. This will be the name of your workflow:



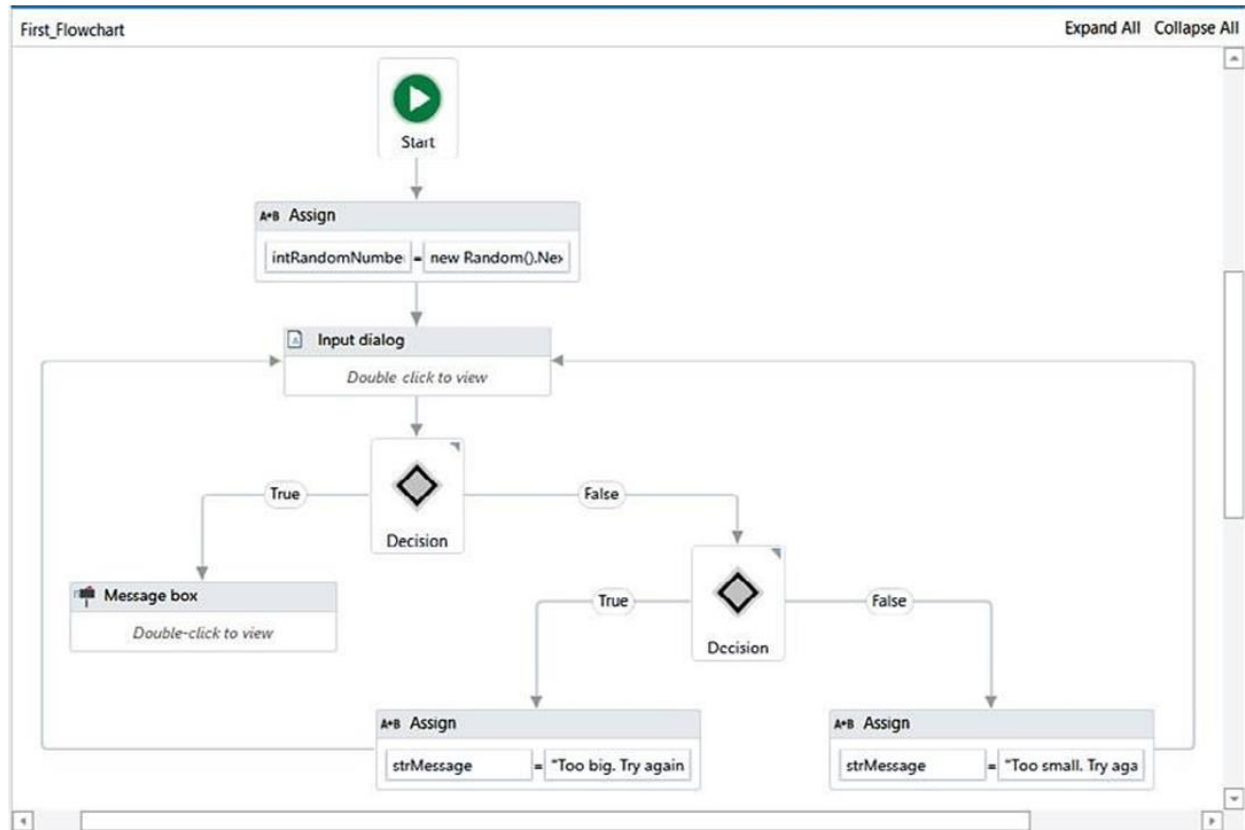
We have just used activities and extracted them in a workflow. If you check the main Designer panel, it looks like the following screenshot:



It automatically generates the Invoke test Workflow activity. Now, when we run the program, it will invoke the workflow that we have extracted (double-click on the Invoke test workflow activity to see which workflow it is going to invoke and where it is generated).

What Flowcharts are and when to use them

A Flowchart is generally used for complex business processes. It provides decision-making Facilities and can be used for both small and large projects. Here, we can add activities in Different ways:



A Flowchart provides multiple branching logical operators to make decisions. A Flowchart is able to run in reverse. Also, it can be used inside Sequences. A Flowchart facilitates reusability for distinct projects. Once we create it to use in a project, it can be used for a different but similar project.

A Flowchart's branches are set to true/false by default. However, its names can be manually changed from the Properties panel.

For example, enter two numbers and check whether their sum is less than 20.

Perform the following steps:

1. First, add a Flowchart from the Activities panel into the Designer panel.
2. Add a Sequence activity within the Flowchart.
3. Take two Input dialog activities (for entering the numbers to be added) inside the Sequence activity.
4. Create the variables x and y to save the values.
5. Next, add a Message box activity to perform a mathematical operation. In our case, the sum of the two numbers is less than 20: $x + y < 20$
6. Now, add a Flow Decision activity to check the mathematical operation.
7. If true, the Flow Decision will flow toward the true branch. Otherwise, it will flow towards the false branch.

Control flow

Control flow refers to the order or the particular manner in which actions are performed in an automation. UiPath provides numerous activities for performing the decision-making process.

These activities, present in the Activities panel, are put into the workflow either using the double-click method or the drag and drop method.

Control flow is the order in which the activities in an RPA bot are executed. Control flow is typically achieved using activities such as the If activity, the Switch activity, and the While activity.

Different types of control flow activities are as follows:

- The Assign activity
- The Delay activity
- The Break activity
- The While activity
- The Do While activity
- The For each activity
- The If activity
- The Switch activity

The Assign activity

The Assign activity is used to designate a value to the variable.

The Assign activity can be used for different purposes, such as incrementing the value of a variable in a loop, or using the results of a sum, difference, multiplication, or division of variables and assigning it to another variable.

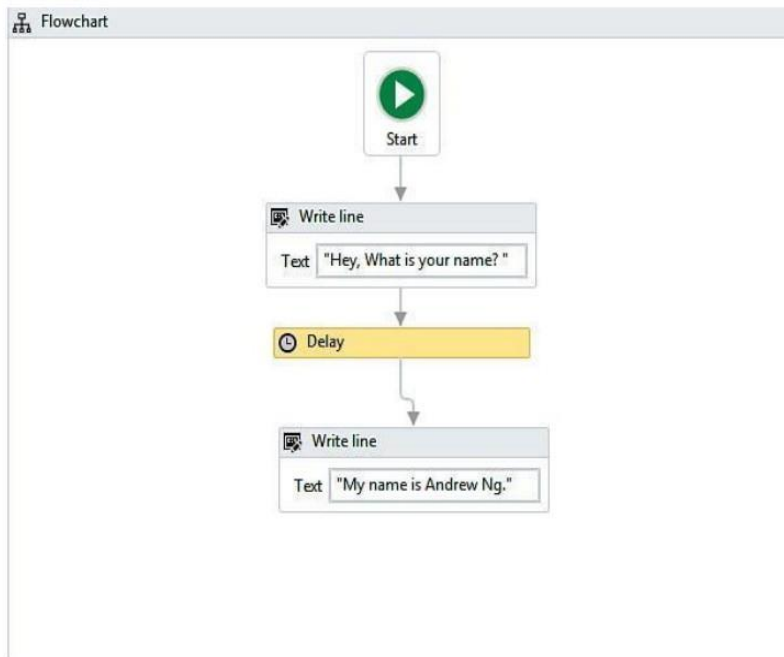
The Delay activity

- The Delay activity, as the name suggests, is used to delay or slow down an automation by pausing it for a defined period of time.
- The workflow continues after the specified period of time. It is in the hh:mm:ss format.
- This activity plays a significant role when we need a waiting period during automation, perhaps say, a waiting period required for a particular application to open.

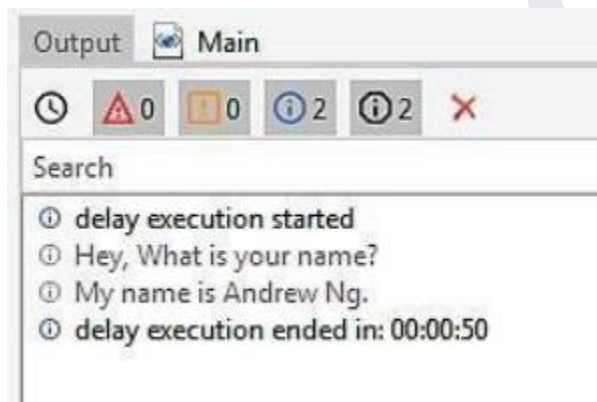
Example: To better understand how the Delay activity works; let us see an example of an automation that writes two messages to the Output panel with a delay of 50 seconds.

Perform the following steps:

1. First, create a new Flowchart.
2. Add a Write line activity from the Activities panel and connect it to the Start node.
3. Select the Write line activity. Now, type the following text into the Text box: "Hey, what is your name".
4. Next, add a Delay activity and connect it to the Write line activity.
5. Select the Delay activity and go to the Properties panel. In the Duration field, set 00:00:50. This is a 50-second delay between the two logged messages.
6. Take another Write line activity and connect it to the Delay activity. In the Text field, write "My name is Andrew Ng,":



7. After clicking on the Run button, the Output panel shows the message that delays it by 50 seconds:

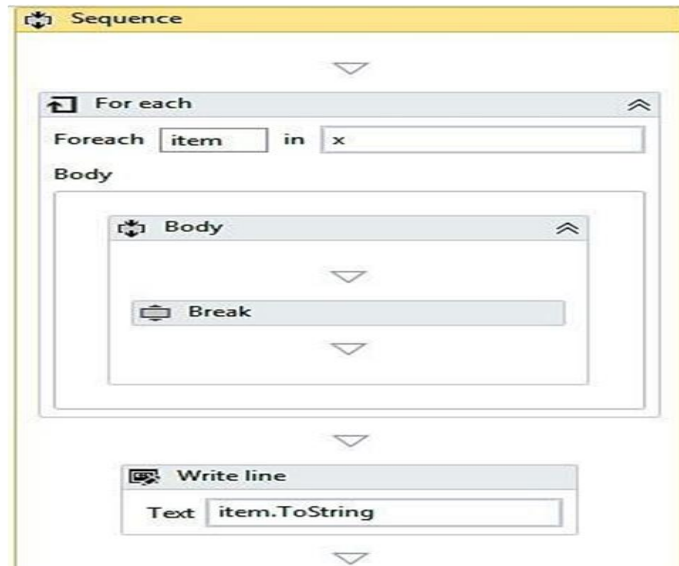


The Break activity

- The Break activity is used to break/stop the loop at a particular point, and then continue to the next activity according to the requirement.
- It cannot be used for any other activity apart from the For each activity.
- It is useful when we want to break the loop to continue to the next activity in the For each activity.

In this example, we will use the Break activity to execute only one iteration. Perform the following steps:

1. Add a Sequence activity to the Designer panel.
2. Next, add a For each activity inside the Sequence (as mentioned in the preceding section, to use the Break activity, we need the For each activity):



3. Create two variables; an integer variable named Item, and an array integer variable named X. Then, set them to the text field.
4. Now, assign a default value to the integer variable X.
5. Add a Break activity inside the body of the loop.
6. Under the For Each activity, add a Write line activity.
7. In the Write line activity, type Item to string in the text field.
8. When we click the Run button, it will display one element, as shown in the following screenshot. This is due to the Break activity, which has stopped execution after the first iteration:

A loop can simply be created by connecting the end of the workflow to the point where we want the workflow to resume. The While, Do while, and For each activities mentioned among the various control flow activities are examples of loops.

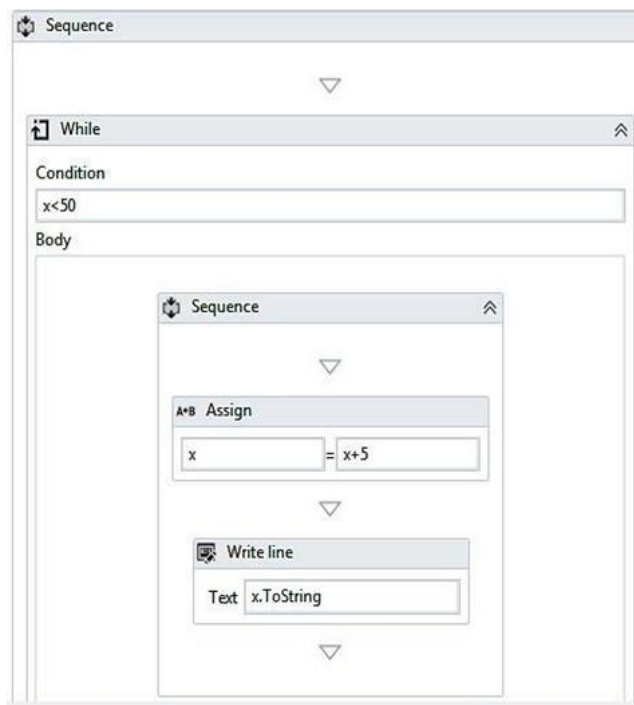
The While activity

The While activity is used in automation to execute a statement or process based on a certain condition. If found true, the loop is executed; that is, the process is executed repeatedly. The project only exits from the loop when the condition does not hold true. This Activity is useful while iterating through an array of elements.

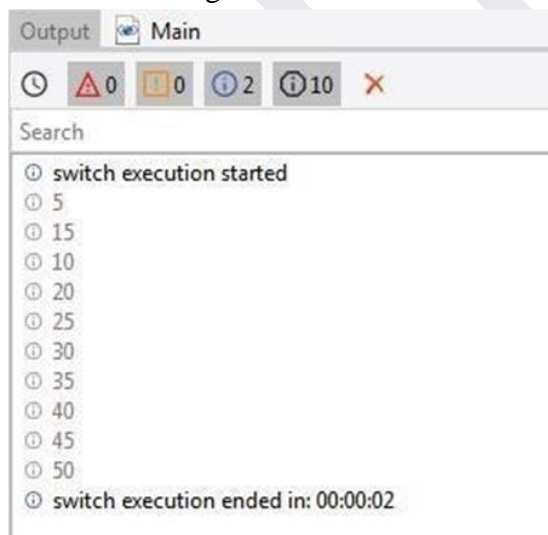
In the example, we will see how an integer variable will increase from 5 to 50 in increments of 5. Perform the following steps:

1. On a Blank project, add a Sequence activity.
2. Now, create an integer type variable X. Set its default value to 5.

3. Next, add a While activity to the Sequence.
4. In the condition field, set $X < 5$.
5. Add an Assign activity to the body section of the While loop.
6. Now, go to the Properties panel of the Assign activity and type in the text field Integer variable for value field integer $X+5$.
7. Drag and drop a Write line activity and specify the variable name X and apply ToString method on this variable:



8. Now, click the Run button. The output will display in the Output panel, as shown in the following screenshot:



The Do while activity

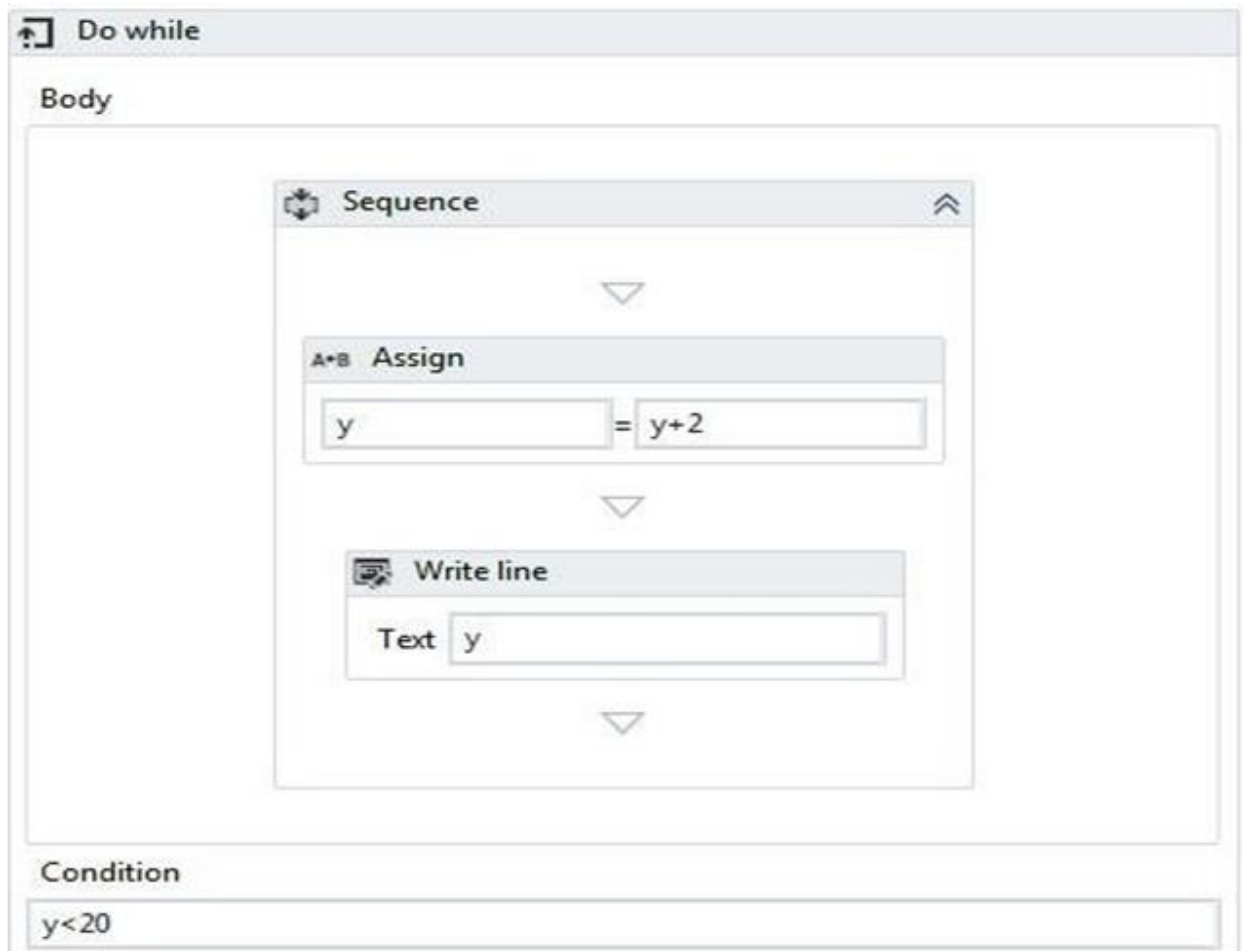
The Do while activity is used in automation when it is required to execute a statement based on the fulfillment of a certain condition.

While activity executes a statement, then checks whether the condition is fulfilled. If the condition is not fulfilled, it exits the loop.

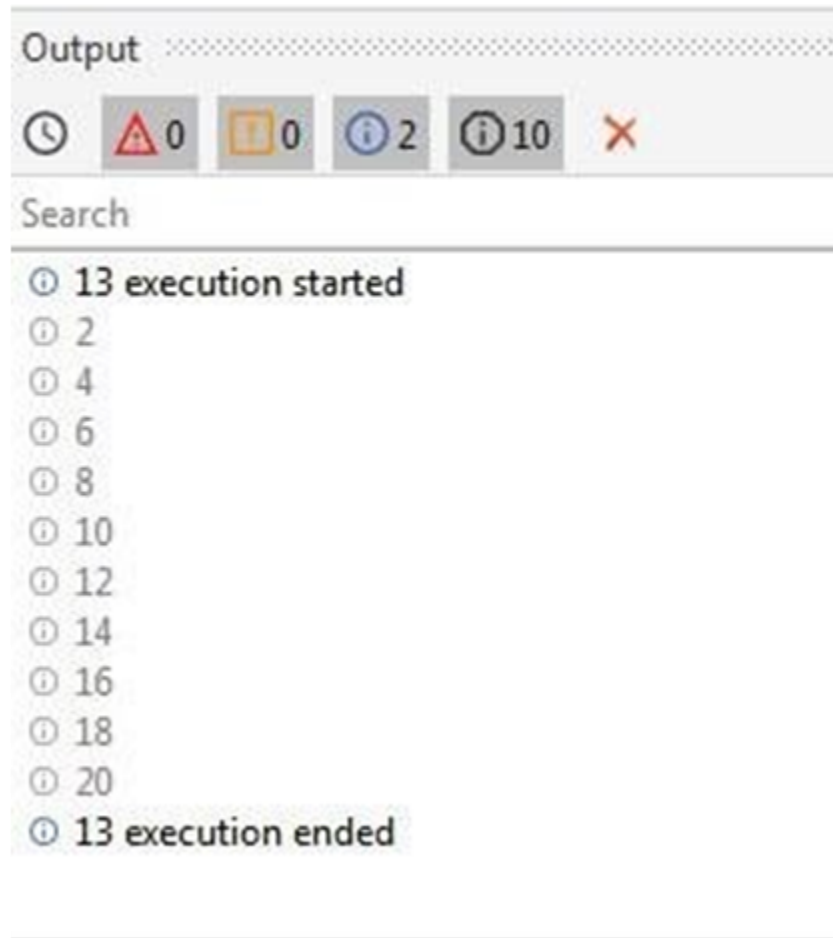
example to understand how the Do while activity works in automation. Take an integer variable. Starting with this variable, we shall generate all multiples of 2, less than 20.

Perform the following steps:

1. Add a Sequence to the Designer panel.
2. Add a Do while activity from the Activities panel.
3. In the body section of the Do while activity, add an Assign activity.
4. Now, select the Assign activity. Go to the Properties panel and create an integer variable y. Set its default value to 2.
5. Set $y+2$ in the value section of the Assign activity to increment the result each time by 2 until the loop is executed.
6. Add a Write line activity inside the Assign activity.
7. In the text field of the Write line activity, type y.
8. In the condition section, set the condition $y < 20$. The loop will continue until the condition holds true:



9. On clicking the Run button, the output displayed will be as follows:



The For each activity

The For each activity works by iterating each element from the collection of items or list of elements, one at a time.

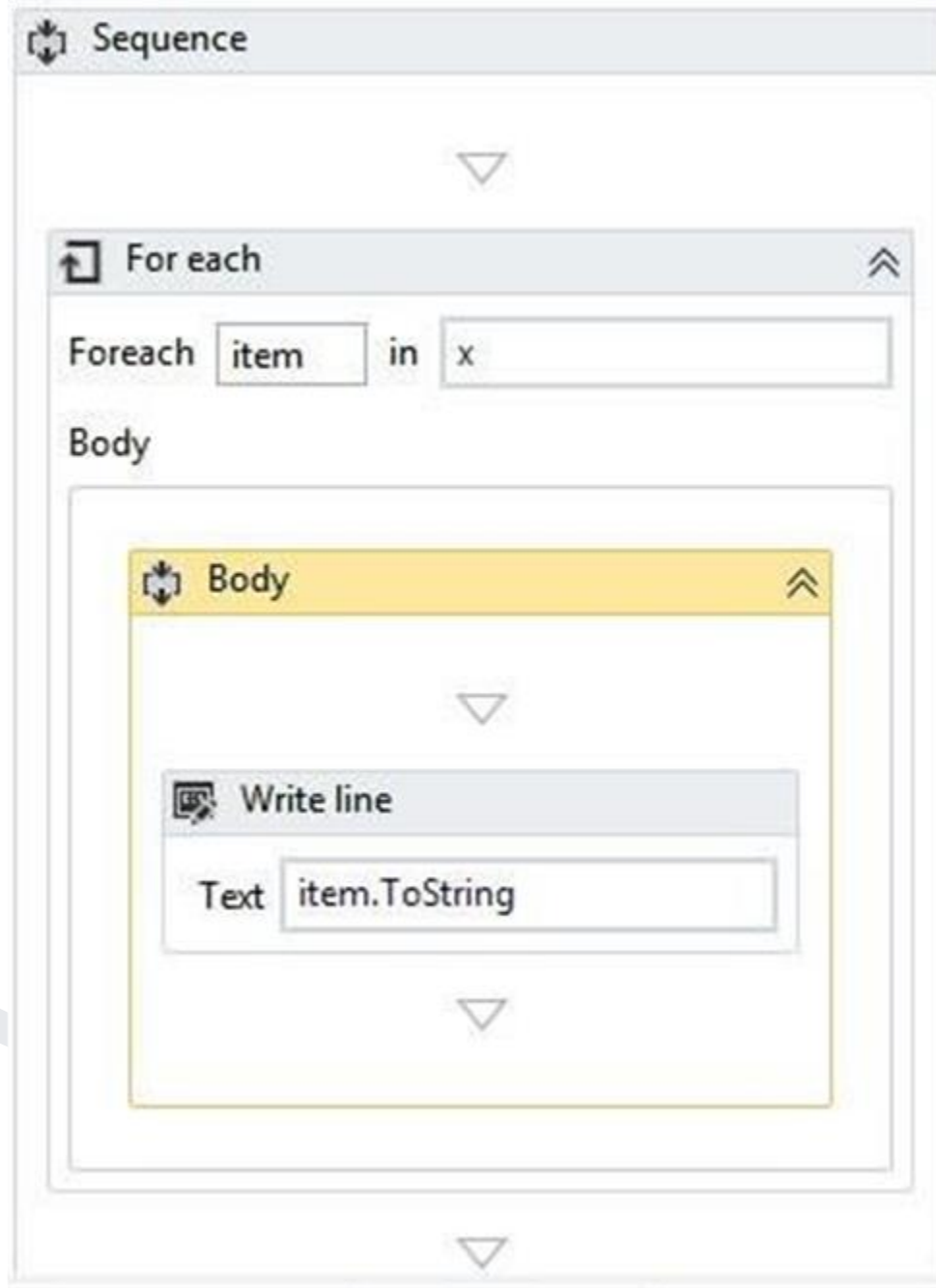
In the process, it will execute all the actions that are available inside the body. Thus, it iterates through the data and processes each piece of information separately.

Example, we shall use the For each activity to go through a collection of even numbers and display each element one at a time.

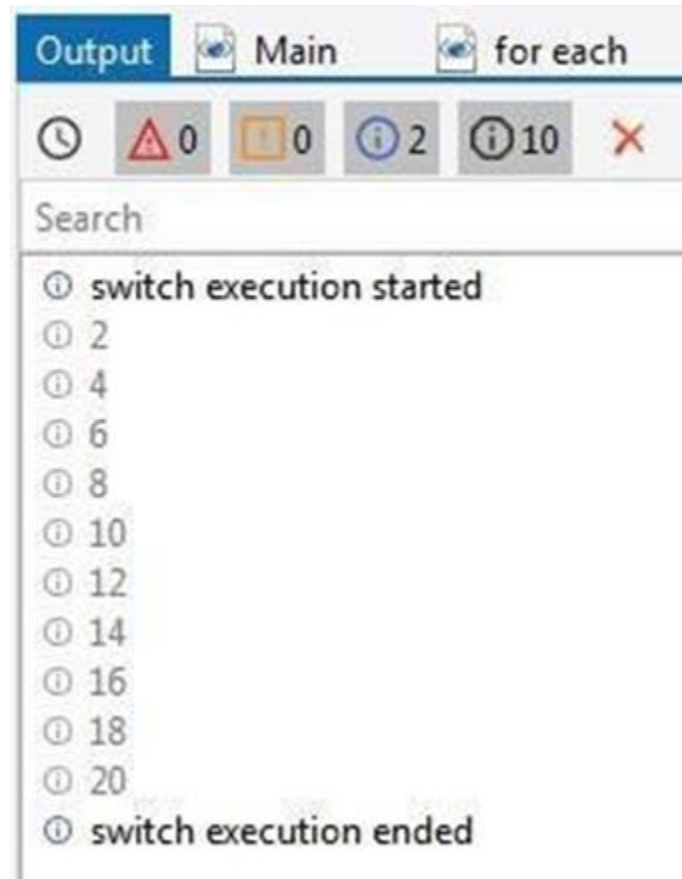
Perform the following steps:

1. Start with a Blank project in UiPath.
2. Add a Sequence activity to the Designer panel.
3. Next, add a For each activity within the Sequence and create an integer type array variable, X.
4. In the default value of the variable, put in ({2,4,6,8,10,12,14,16,18,20}).

5. Add a Write line activity to the Designer Panel (this activity is used to display the results).
6. In the Text field of the Write line activity, type item. ToString to display the output:



7. Now, run the program. You will see that each number of the array is displayed one by one because of the use of the For each activity:



The If activity and the Switch activity are the Control flow's decision-making activities.

The If activity

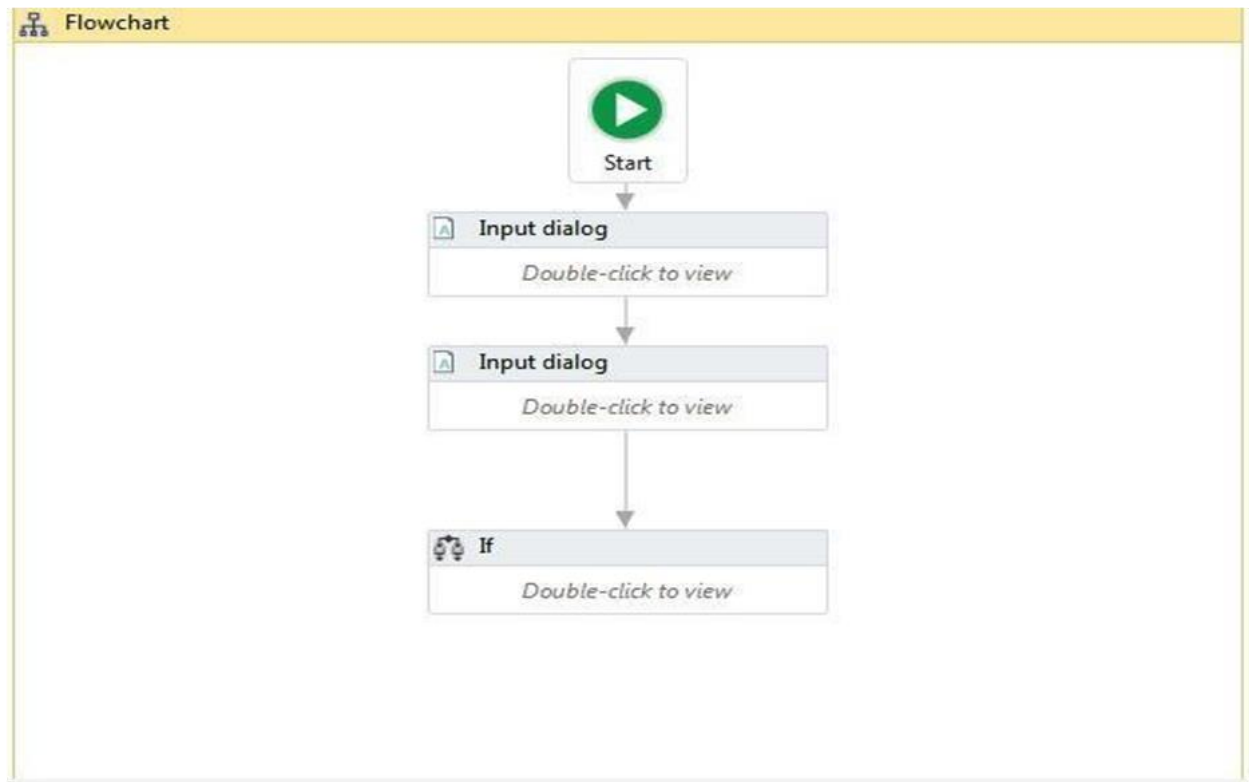
The If activity consists of a statement with two conditions: true or false.

If the statement is true, then the first condition is executed; if not, the second condition is executed.

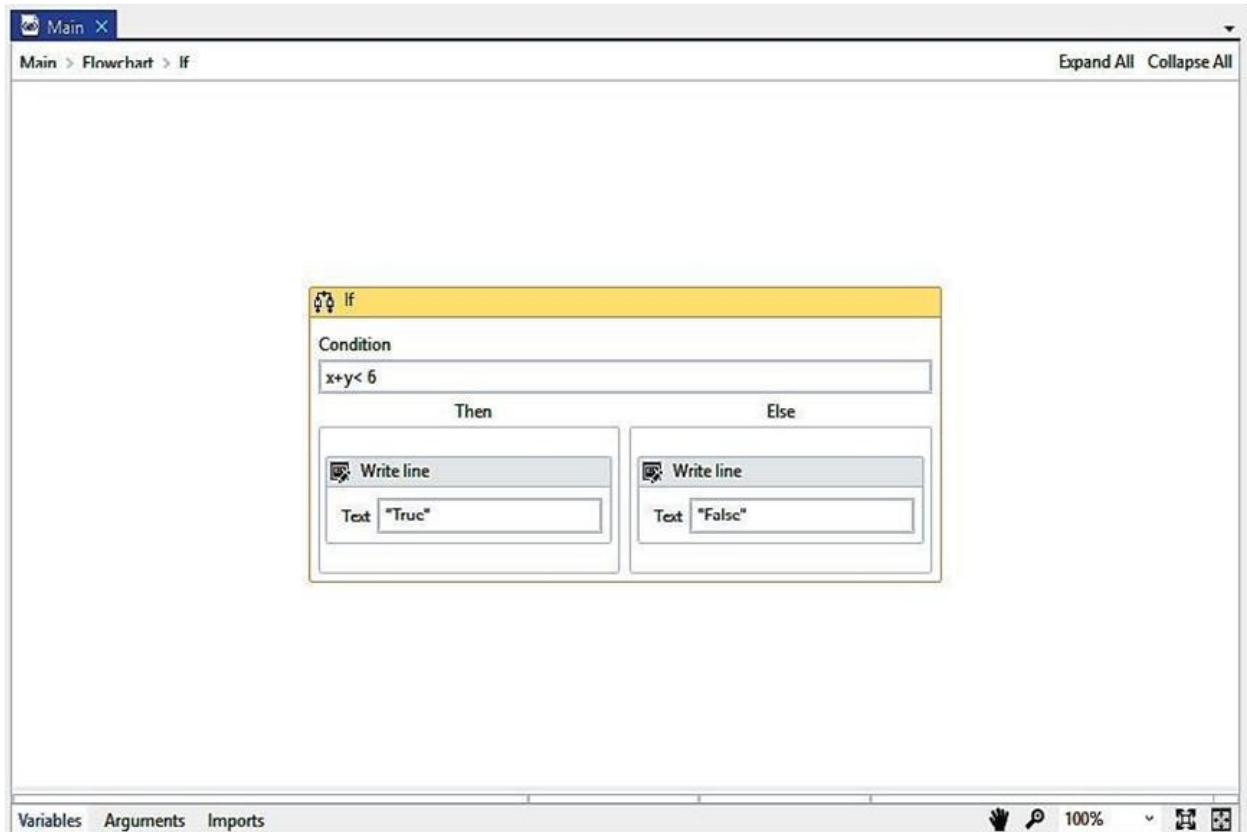
This is useful when we have to take decisions on the basis of statements.

example that checks whether the sum of any two numbers is less than 6. Perform the following steps:

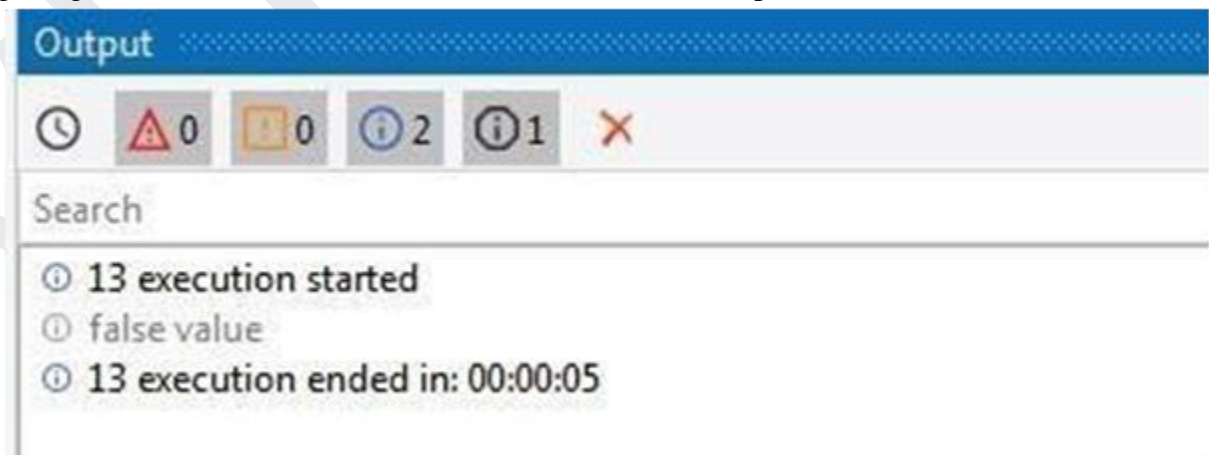
1. Add a Flowchart from the Activities panel.
2. Add two Input dialog activities. Create two integer variables, x and y.
3. In the Properties panel, change the label name and title name of both the Input dialog activities.
4. Now, specify the name of these two variables in the Result property of both the Input dialog activities.
5. Now add the If activity to the Designer panel.



6. In the condition part, $X+Y < 6$ check whether it is true or false. Add two Write line activities and type "True" in one and "False" in the other:



7. Click the Run button to check the output. If the condition holds true then it will show the true value; otherwise, it will show the false value, as shown in the second screenshot (in our case, we put in the values of x and y as x and y, respectively, thus getting a sum of 13, which is not less than 6; hence, the output shows it as false value):



The Switch activity

The Switch activity can be used to make a choice.

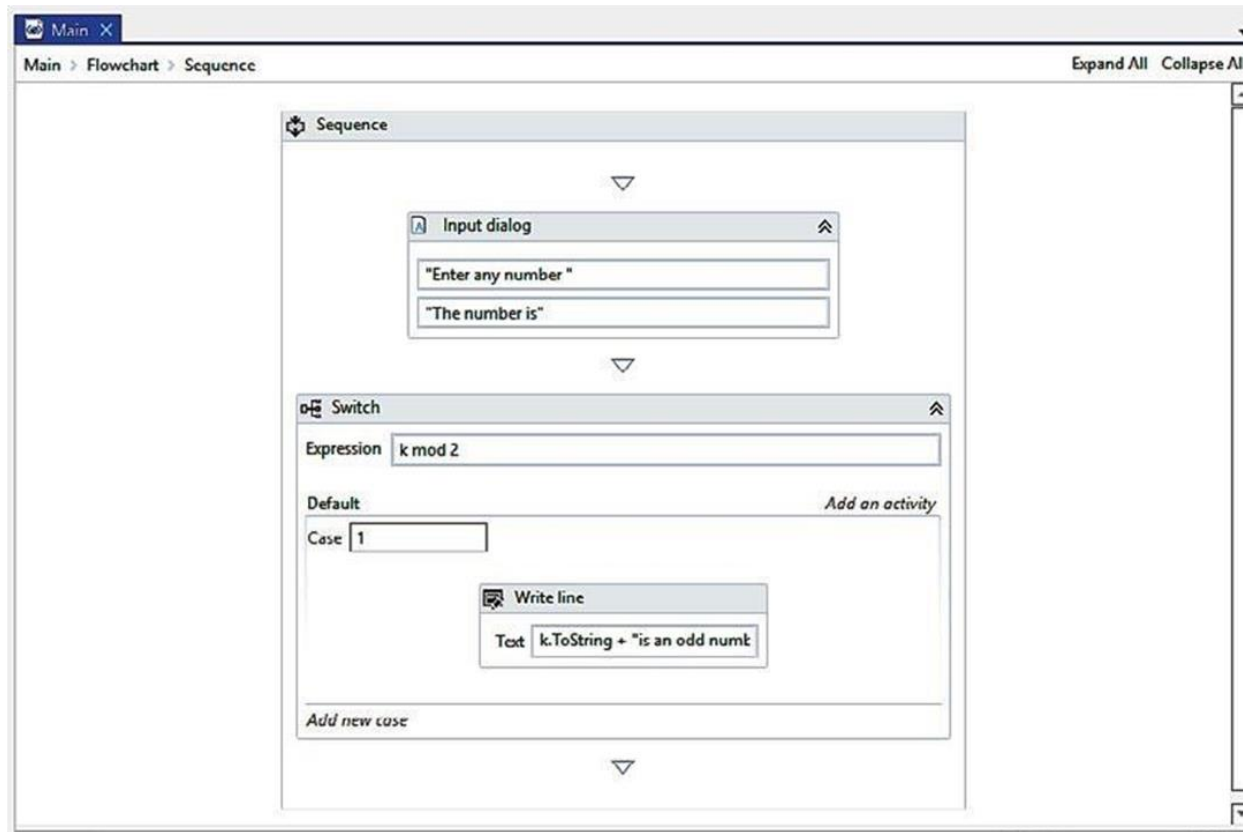
- When we have various options available and want to execute one option, we frequently use the Switch activity.
- By default, the Switch activity takes an integer argument. If we want to take a desired argument, then we can change it from the Properties panel, from the Type Argument list.
- The Switch activity is very useful in the categorization of data according to one's own choice.

Example where we have to check whether a given number is odd or even. We know that all odd numbers, when divided by 2, leave a remainder of 1.

On the other hand, even numbers, on being divided by 2, leave a remainder of 0. Hence, we will have two cases getting a remainder of 1 or 0.

Perform the following steps:

1. Add a Sequence activity.
2. Add an Input dialog activity inside the Sequence.
3. Now, create an integer type variable k.
4. Specify the newly created variable's name in the Result property inside the Properties panel.
5. Add the Switch activity under the Input dialog activity.
6. In the Expression field, set $k \text{ mode } 2$ to check whether the number is divisible by 2 or not.
7. Add a Write line activity to the Default section and type the k. To string + "is an even number" in the text field.

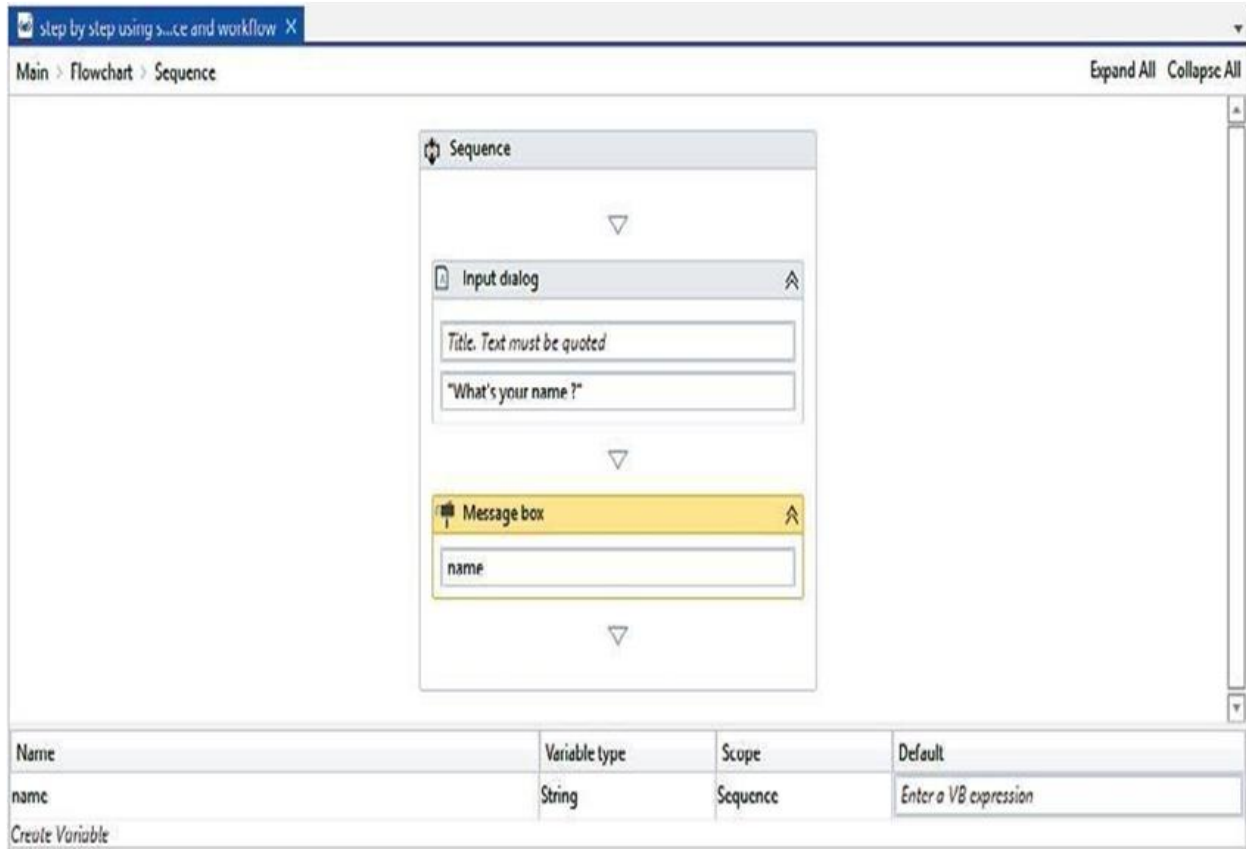


Step-by-step example using Sequence and Flowchart

- A Sequence and a Flowchart are similar concepts.
- They are both used to contain logical steps or actions.

How to use a Sequence

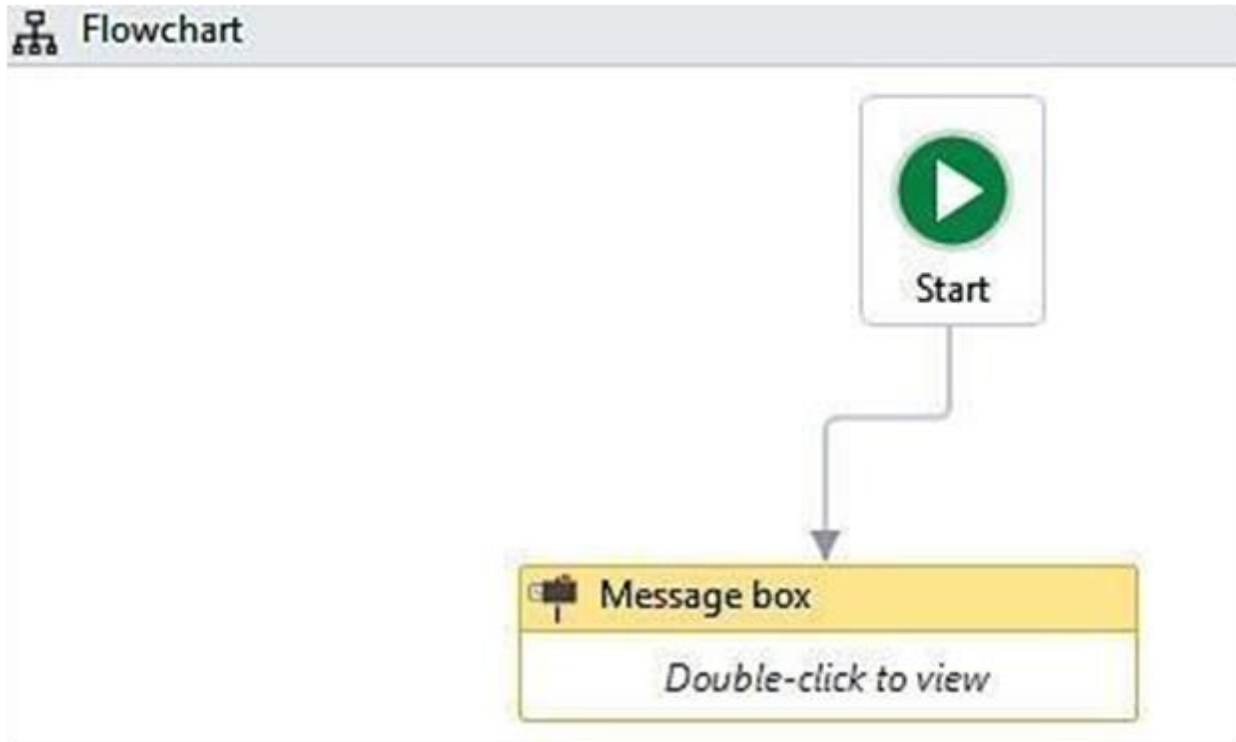
- There may be different Sequences doing their jobs. We can easily put similar Sequences into a workflow; each workflow represents a task.
- It is very easy to test a separate workflow alone.
- Perform the following steps:
 1. Drag and drop a Flowchart onto the Designer panel. Drag and drop a Sequence activity. Connect the Sequence activity with the Start node.
 2. Double-click on the Sequence activity. Drag and drop an Input dialog activity and a Message box activity. Specify a message in the label property of the Input dialog activity.
 3. Create a variable of type String. Give it a name. Also, specify this newly created variable's name in the content property of the Message box activity:
 4. Hit the Run button or press F5 to see the result.



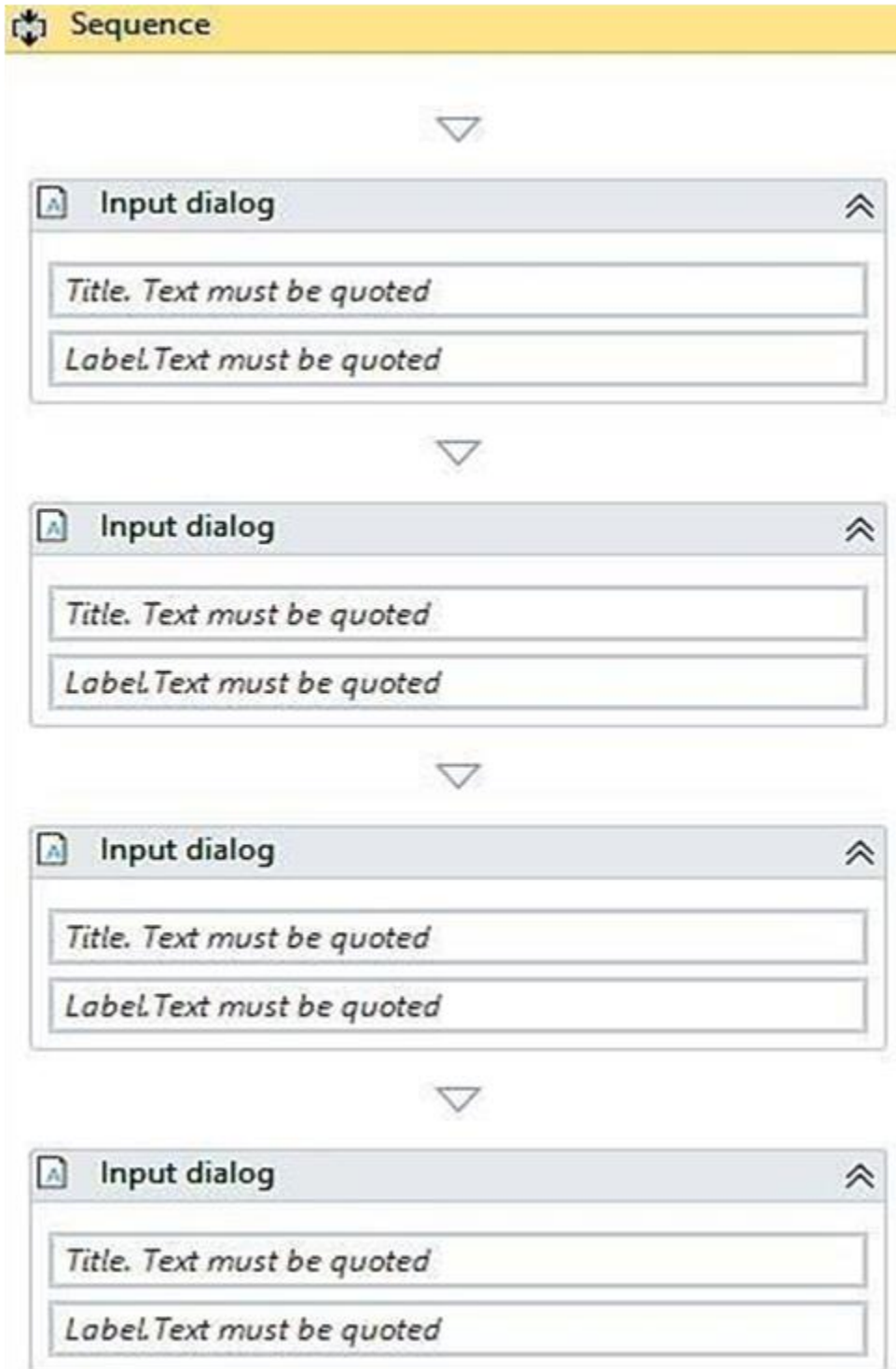
We can see clearly that we have used two activities inside the Sequence that are logically related (one for inputting the name and the other for popping it up). Here, the Sequence contains two activities.

How to use a Flowchart

- A Flowchart is a container. It can contain activities inside it.
- Let us drag and drop a Message box activity inside the Flowchart. Double click on the
- Message box and type "Hello World!" in the area where the text is to be quoted. Press F5 to see the result):

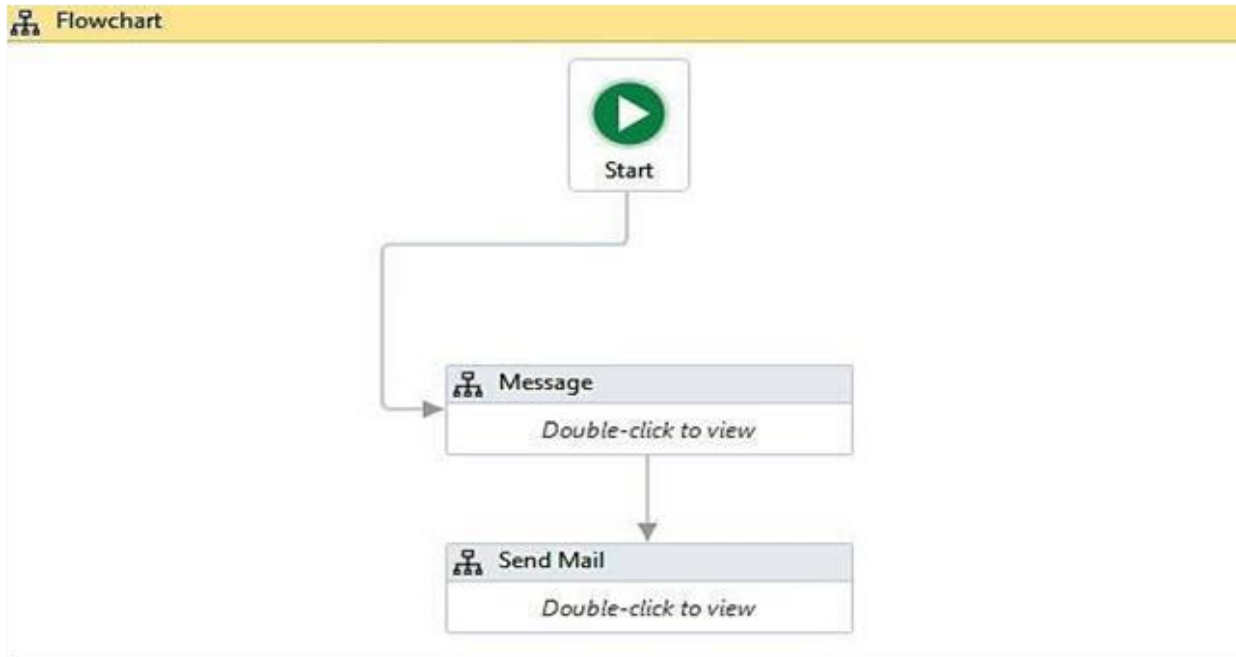


- So, when the program has only a few steps, we can use activities directly inside the Flowchart.
- However, it becomes more complex when we have a large number of steps. That is why it is necessary to arrange the related activities into Sequences and then group the Sequences into a Flowchart.
- Example to see how to use Sequences in the Flowchart. Perform the following steps:
 1. Drag and drop two Flowchart activities on the main Flowchart. Rename them as Send mail and Message. We have two different workflows.
 - The Send Mail workflow will send the mail to an email address.
 - The Message workflow has the message body of that email and will ask the user for a name, message, sender, and receiver.
 2. We have to implement the desired steps in both workflows. For that, we are using a Sequence inside the Flowchart.
 - Double click on the Flowchart. Drag and drop a Sequence activity inside both Flowcharts.
 - Connect the Sequence to the Start node by right-clicking on the Sequence and selecting the Set as Start node option.
 3. Double click on the Sequence in the Message Flowchart.
 - Drag and drop four Input dialog activities for the name, message, sender, and receiver.



4. Double click on the Send Mail Flowchart. Double click on the Sequence. You can drag and drop the email activities here.

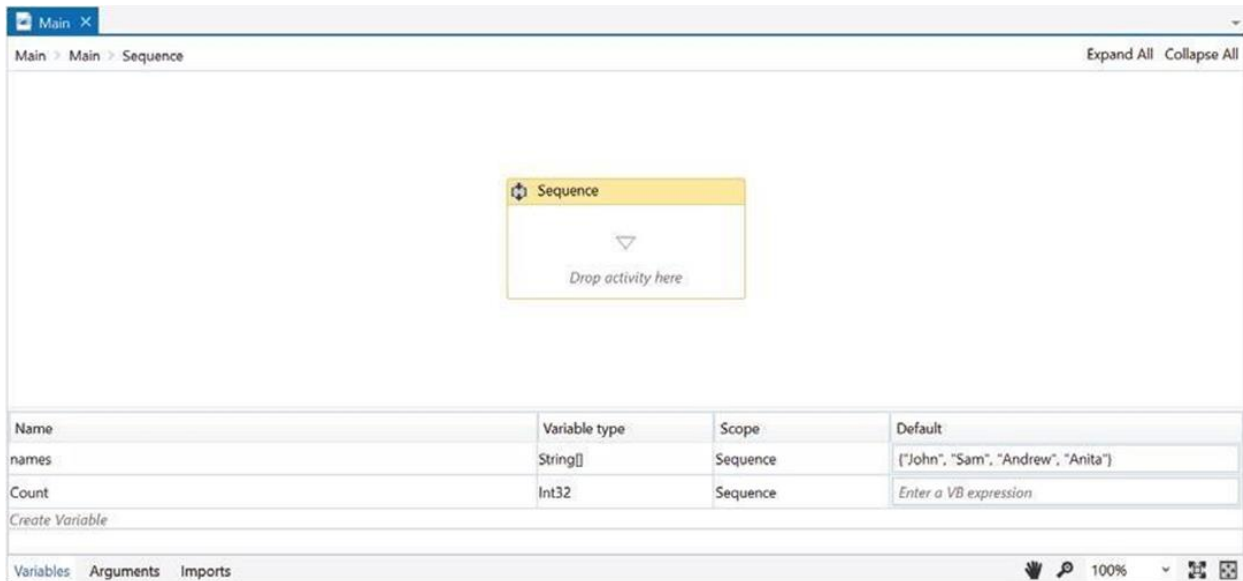
5. That's it. Now, go to the main Flowchart. Connect the Message Flowchart to the Start node. Also, connect the Send Mail activity to the Message Flowchart:



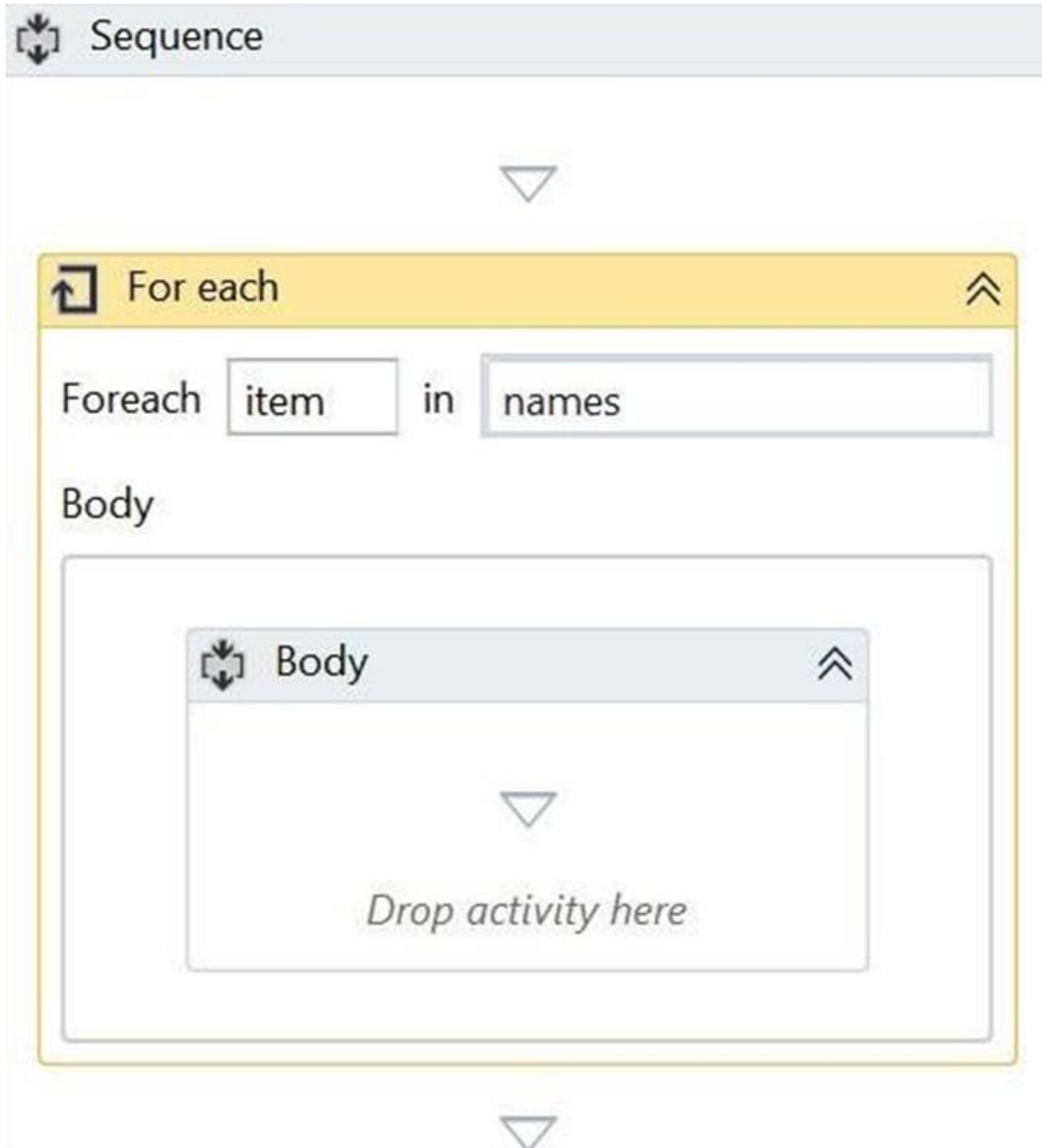
1. Run the program and visualize it.

Step-by-step example using Sequence and Control flow:

- Consider an array of names. Say we have to find out how many of them start with the letter a.
- We will then create an automation where the number of names starting with a is counted and the result is displayed.
- Perform the following steps:
 1. Drag and drop a Flowchart activity from the Activities panel.
 2. Drag and drop a Sequence activity inside the Flowchart. Connect the Sequence to the Start node by right-clicking on the Sequence activity and selecting the Set as Start node option.
 3. Double click on the Sequence activity. Create a variable. Give it a name (in our case, we will create an array of type string and name the variable as name). Set the variable type to Array of [T]. When asked for the type of array, select String. Also, initialize the array in the Default section of the variable by giving it a default values. For example, {"john", "sam", "Andrew", "Anitha"}.
 4. Create a variable of type integer Count for storing the result. Set the variable type to Int32:

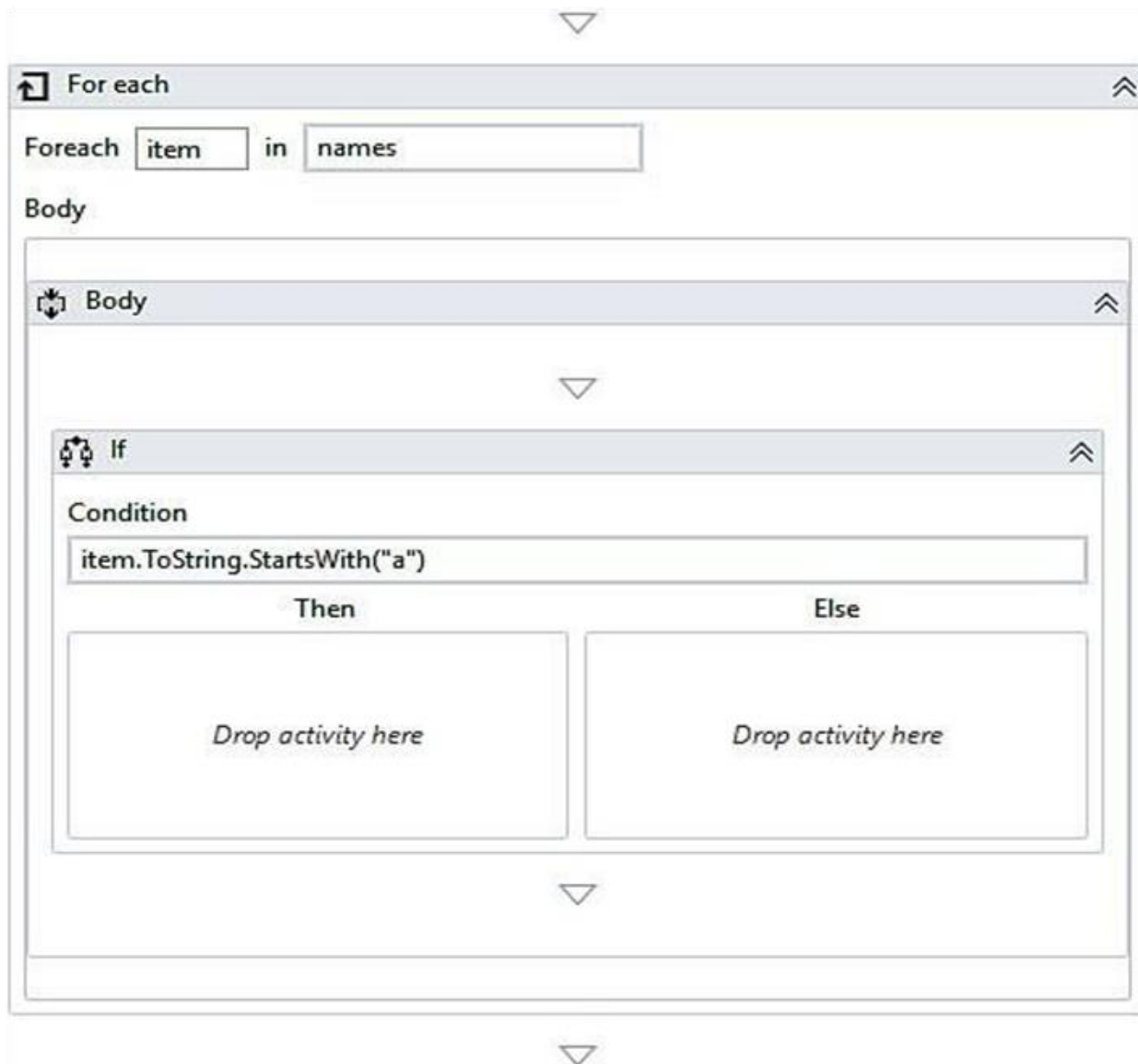


5. Drag and drop a For each activity inside the Sequence. Also, specify the array name in the expression box of the For each activity. The For each activity is used to iterate over the array. It will pick up one name from the array each time until it reaches the end.



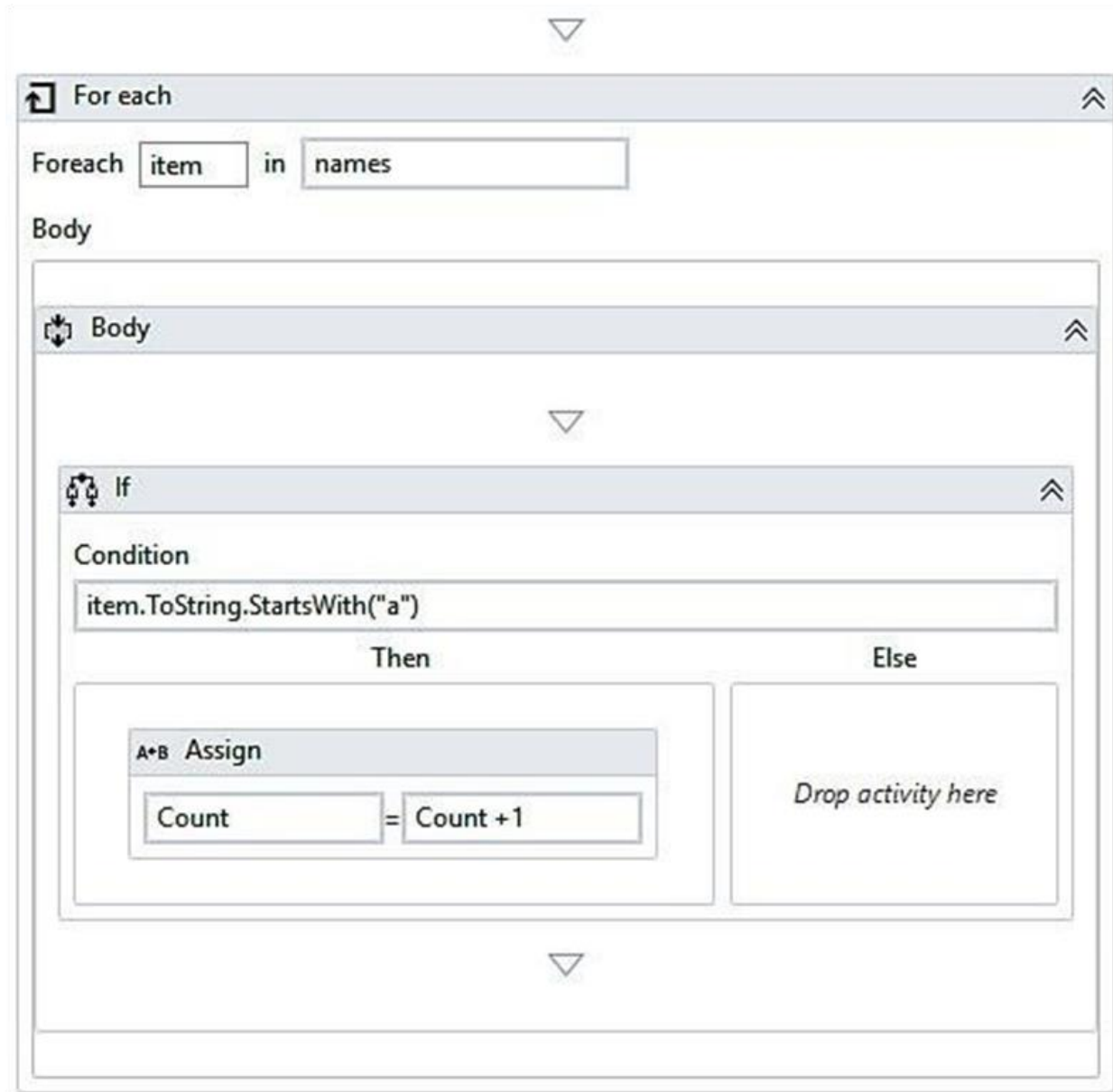
6. Drag and drop the If activity from the Activities panel and place it inside the For each activity at the location where Drop activity here is mentioned.
- Specify the condition in the expression box of the If activity. The If activity is used to check for a particular condition/expression.
 - If that expression is satisfied, the Then block will be executed. Otherwise, the Else block will be executed.
 - We have specified the expression as `Item.ToString().Startswith('a')`. This expression specifies the name present in the item variable starts with the letter 'a'.
 - The For each activity iterates over the array, picks up one name at a time,

- and stores it as a variable, item:

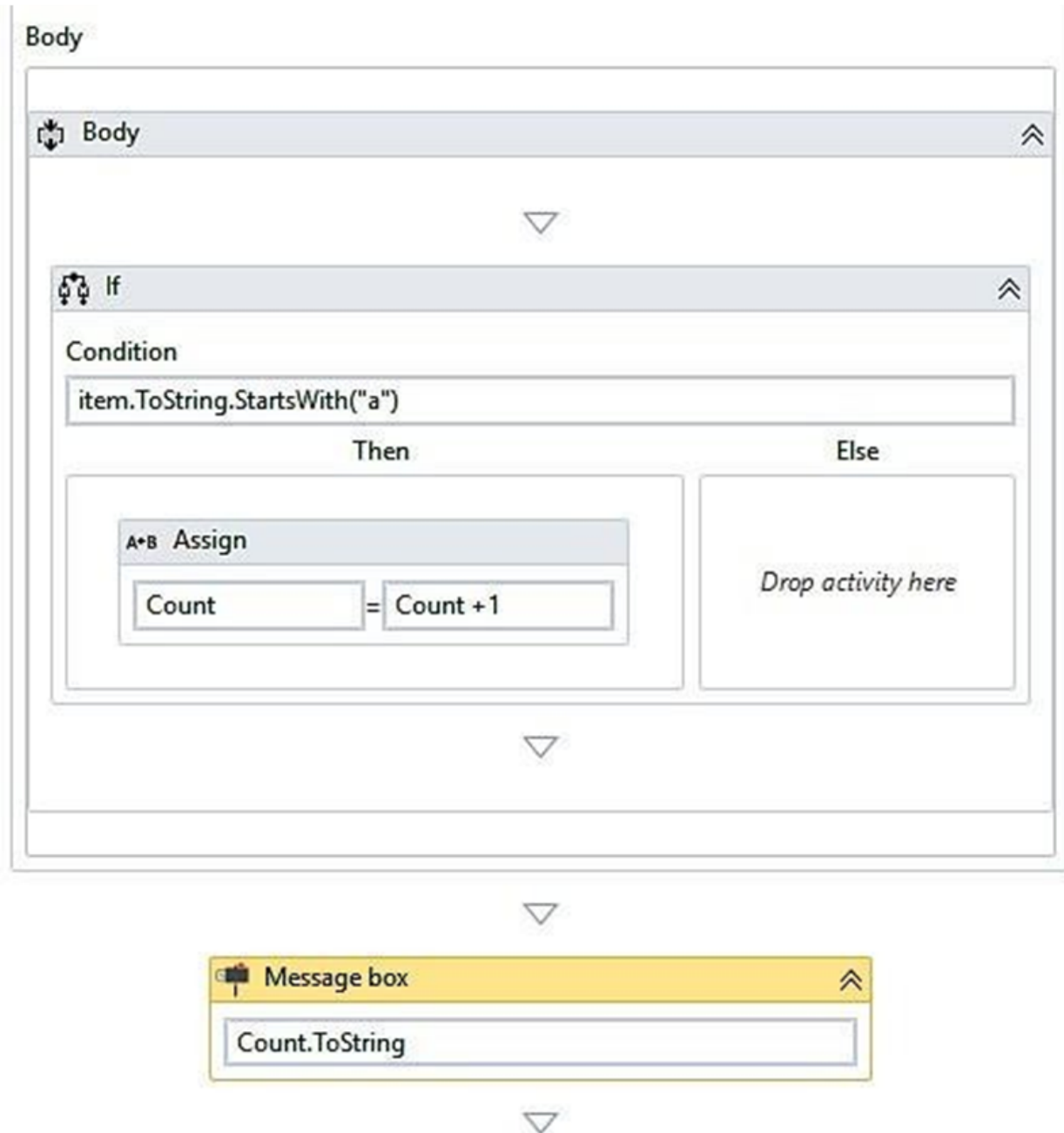


7. Now, we are going to use the count variable and increment it each time a name from an array starts with the letter a.

- we have to use the A+B Assign activity. Drag and drop the A+B Assign activity inside the If activity
- Set the To property to count(variable name) and the Value property to Count+1(to increment its value) of the A+B Assign activity:



8. Just drag and drop the Message box activity inside the Sequence activity. Specify the count variable in the expression box of the Message box activity. But remember, the variable that we have created is of type Int32, so, it cannot be used with the Message box activity without converting it to a string. To convert it to a string, we have the '.tostring' method available in UiPath Studio. Just apply it to the variable and select '.tostring':



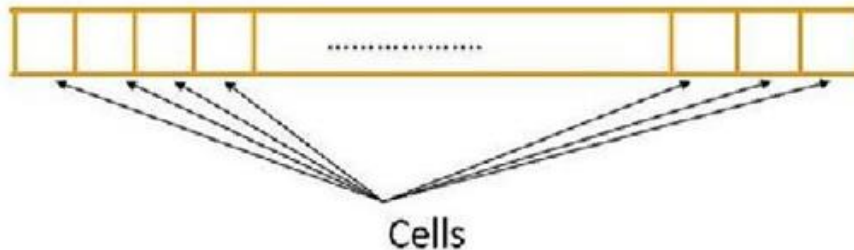
Data Manipulation

Data manipulation is the process of changing data whether it is adding, removing, or updating it.

Variables and scope

Before discussing variables, let us take a look at Memory and its structure:

Memory

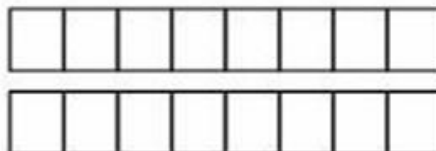


Each cell can store 1 bit of information having the value 0 or 1

Memory consists of millions of memory Cells and each memory cell stores data in the form of 0s and 1s (binary digits).

Each cell has a unique address, and by using this address, the cell can be accessed.

2 bytes:



one 16-bits memory cell:



When data is stored in memory, its content gets split into further smaller forms (binary digits). As shown in the preceding diagram, 2 bytes of data consists of several memory cells.

A variable is the name that is given to a particular chunk of memory cells or simply a block of memory and is used to hold data.

A variable is used to store data. Data is present around us in different Types-it can be an mp3 file, text file, string, numbers, and so on.

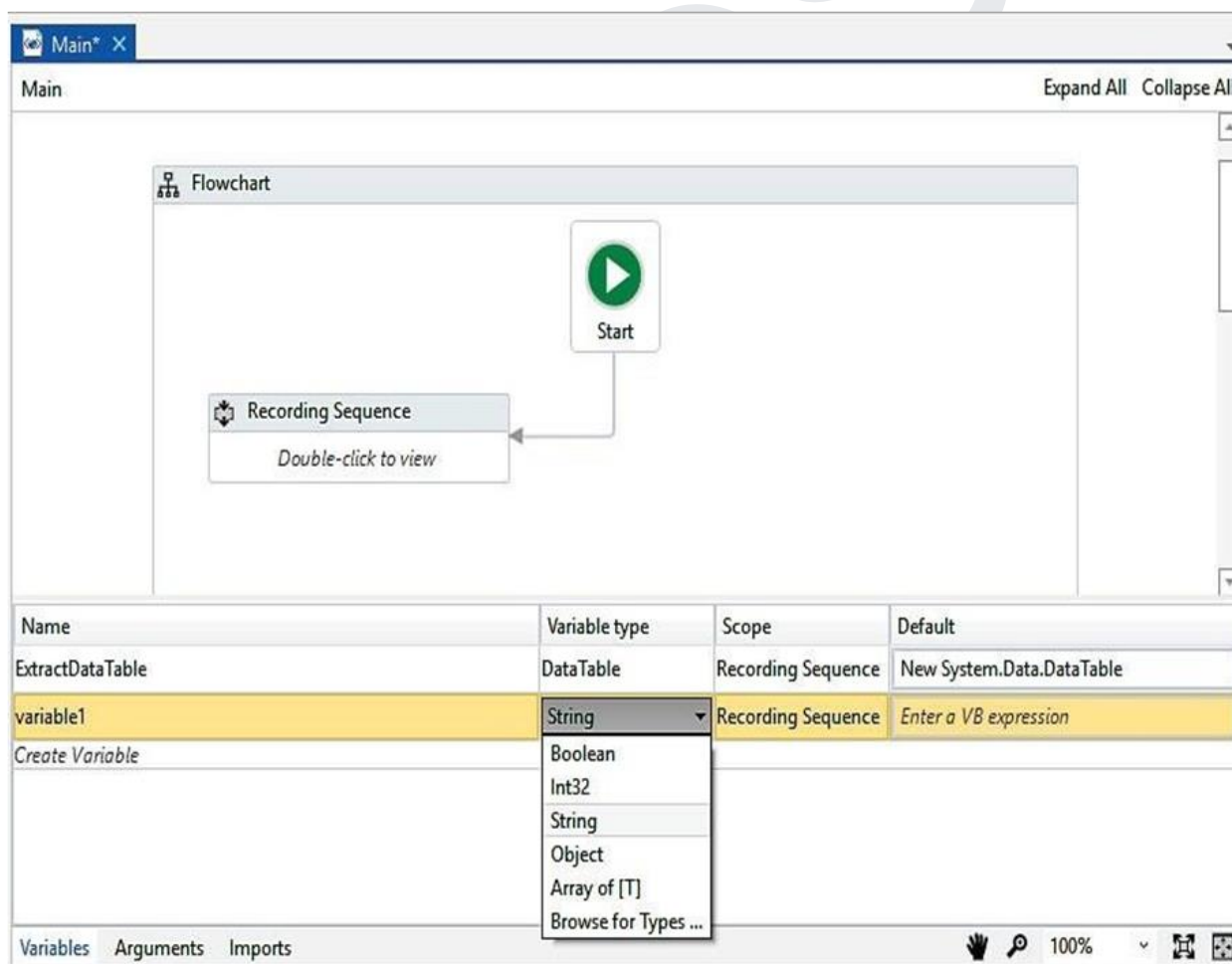
A particular type of variable can hold only that type of data.

If there is a mismatch between the data and the variable type, then an error occurs.

The following table shows the type a of variable available with UiPath:

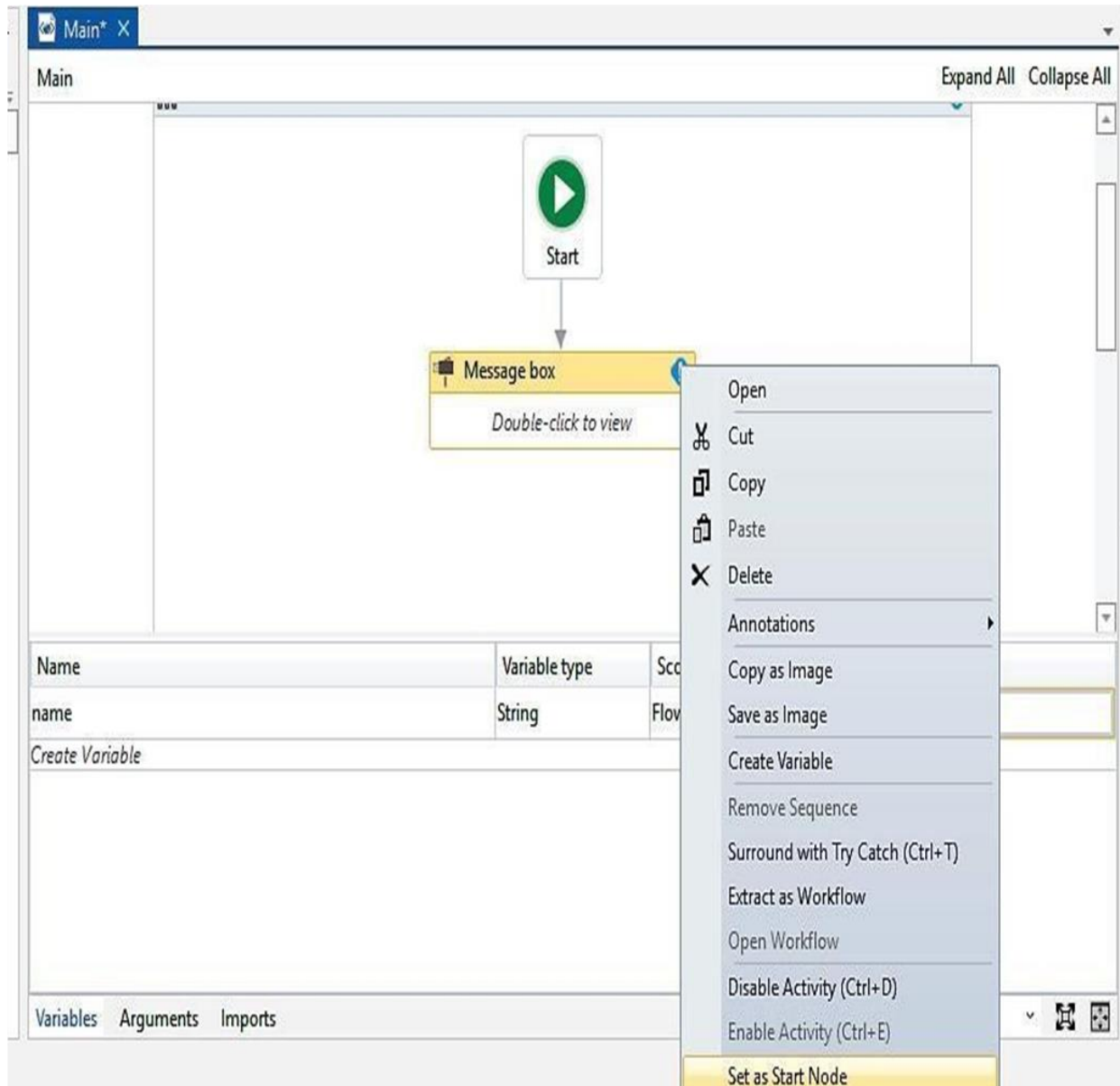
Type	Content
Integer	Whole numbers
String	Text of any kind: "The Quick Fox @4596"
Boolean	True or false
Generic	Anything

- In UiPath, we can declare a variable in the Variables section. Just give it a meaningful name and select the appropriate type from the drop-down list.
- We can also specify the scope of a variable. The Scope is the region under which the data has its effect or availability.
- You can choose the Scope of the variable according to your requirements; try to limit it as far as possible.

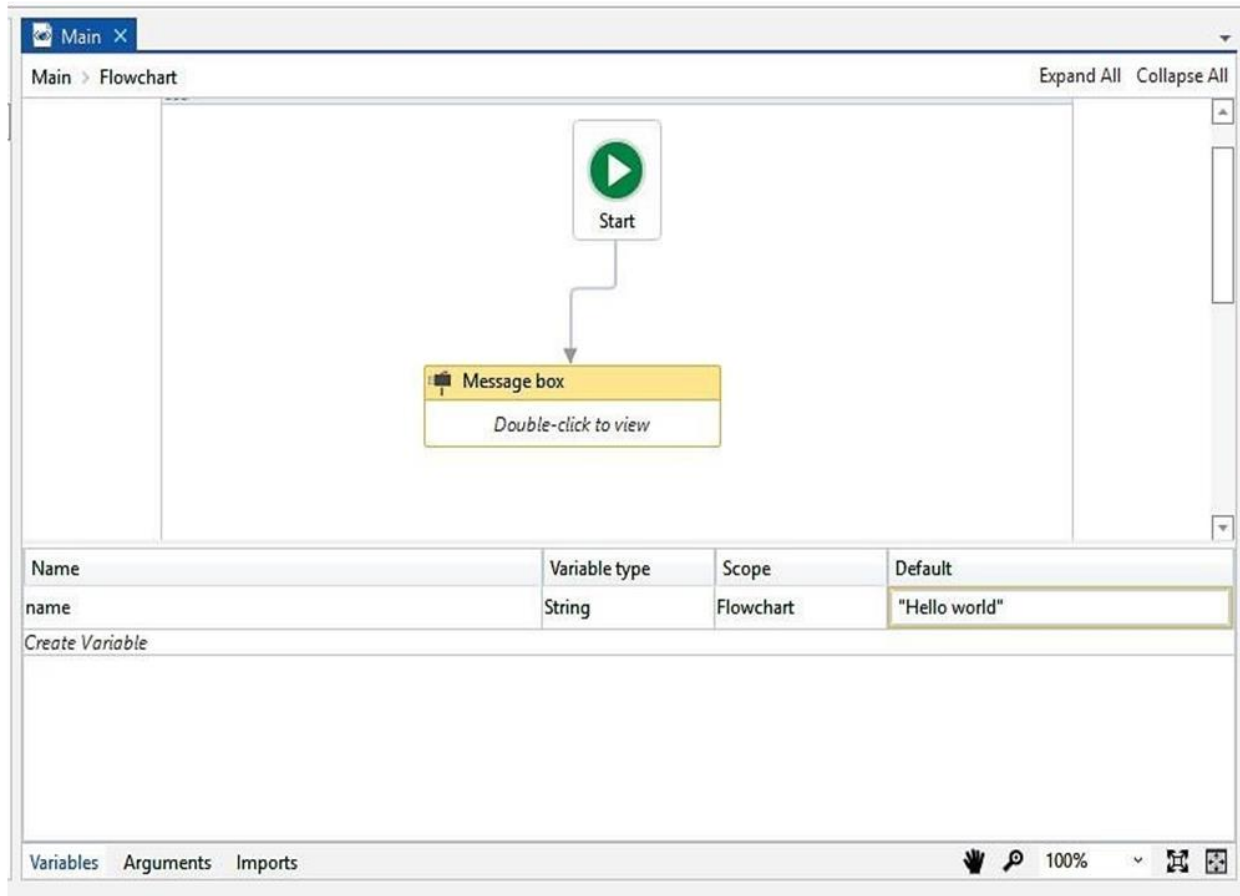


Let us take an example of creating a variable and then displaying a Message box using that variable:

1. We have declared a variable as name in the Variables section and set its Default value to "Hello World". By default, the type of the variable is String (we can change its type according to our needs).
2. Search for Message box in the Activities panel. Drag and drop that Message box template into a Flowchart.
3. Right-click on the message template and select Set as Start node:



4. Double-click on the Message box template and specify the variable name that we Created earlier. At this stage, we are ready to run our application by simply clicking on the Run button:



A dialogue box will pop up with the “Hello World” text displayed on it.

Collections

There are different types of variables. Variables can be classified into three categories:

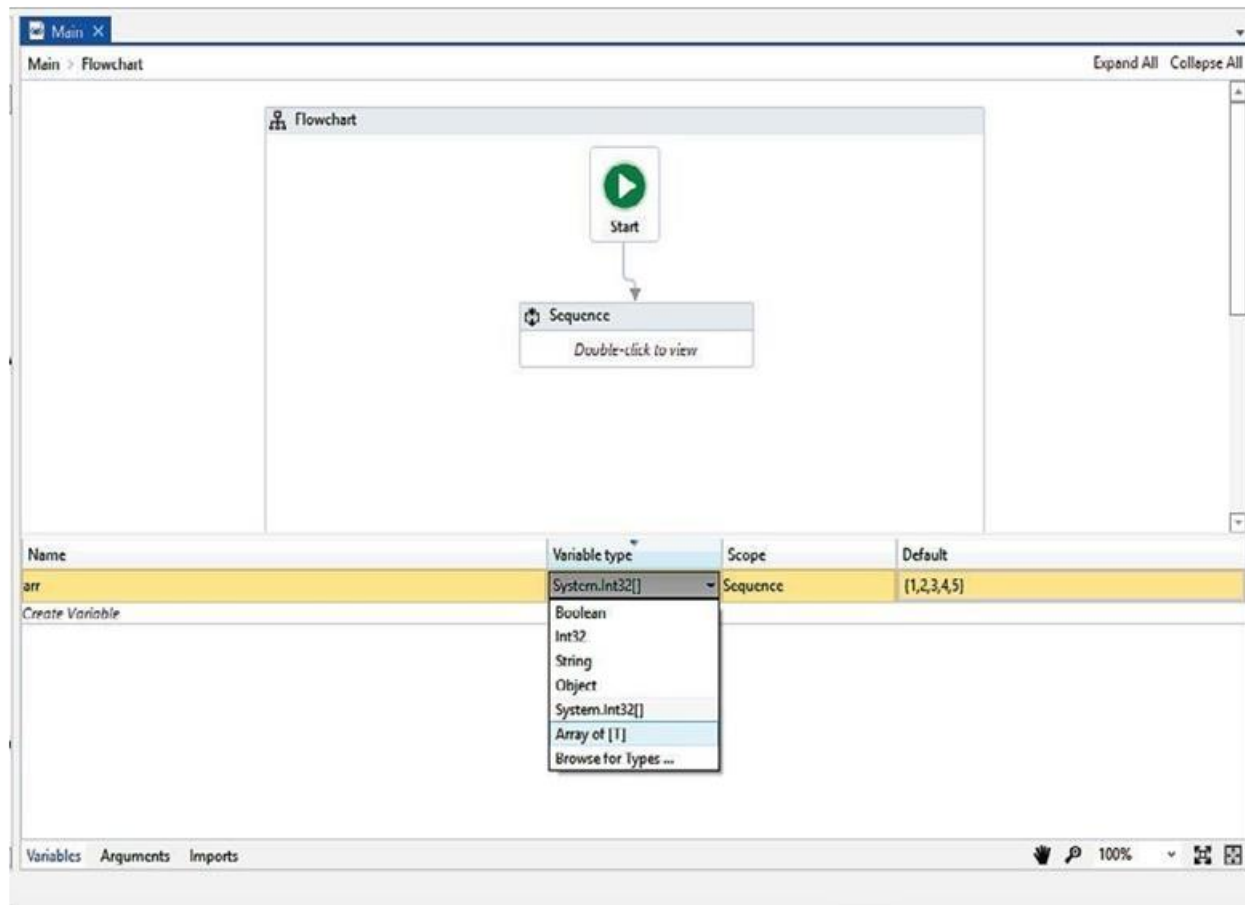
- **Scalar:** These are variables that can only hold a single data point of a particular data type, for example; Character, Integer, Double, and so on.
- **Collections:** These are variables that can hold one or more data point of a particular data type. For example; array, list, dictionary, and so on.
- **Tables:** These are a tabular form of the data structure which consists of rows and columns.

In a collection, we can store one or more data points, but all the data must be the same.

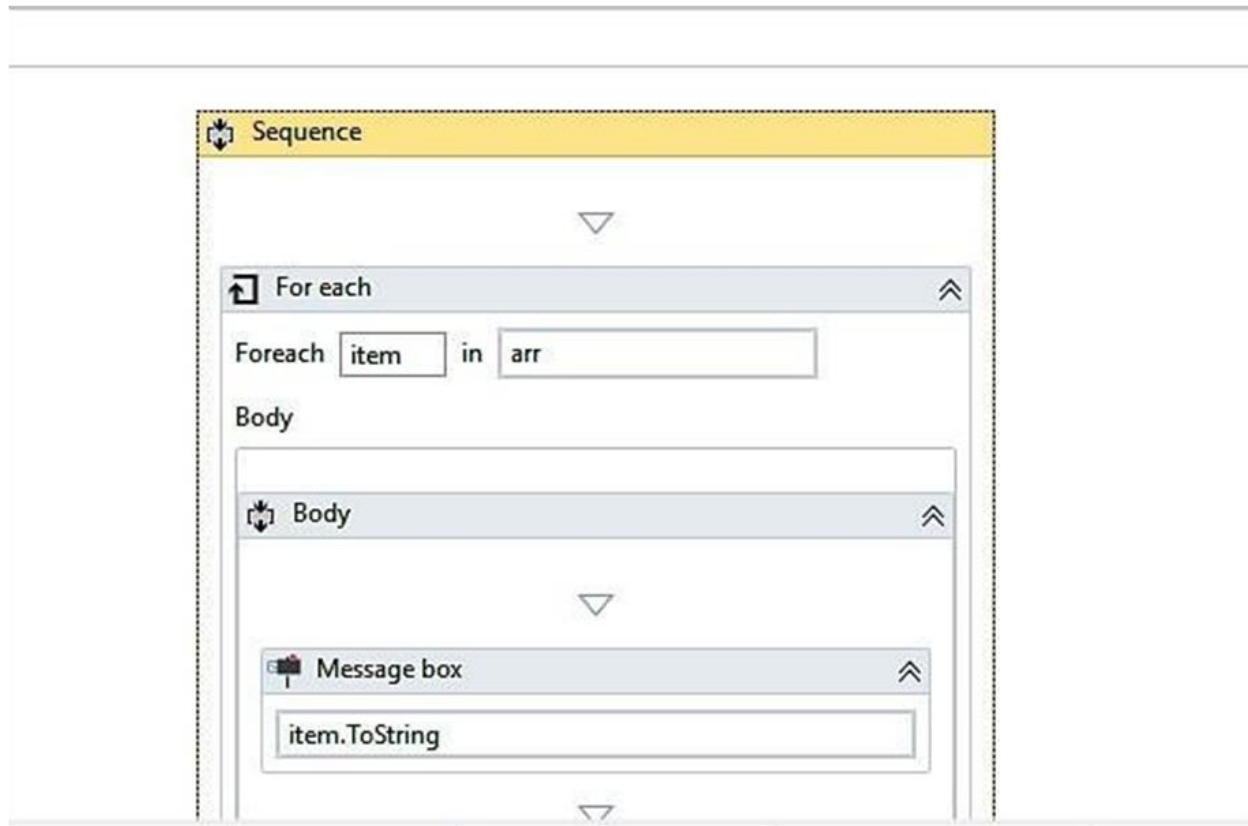
In this example, we are going to take an array of integers, initialize it, and then iterate through all the elements of the array:

1. Drag and drop a Flowchart activity onto the main Designer panel, and drag and drop a Sequence activity inside the Flowchart. Set the sequence as Start node.

2. Create a variable in the Variables panel and give it a meaningful name (in this example, we have created a variable named arr, which is an array of integers). Choose the data type as an array of integers.
3. We have initialized the array as {1, 2, 3, 4, 5} in the Default section. You can initialize it with the int32 data type:



4. Drag and drop a For each activity from the Activities panel inside the Sequence, and drag and drop a Message box activity inside the For each activity.
5. Specify the array name in the expression text box of the For each activity.
6. Specify the item variable that is auto-generated by the For each activity, inside the Message box activity. hold on, we have to convert the item variable into the String type because the Message box activity is expecting the string data type in the text box. Just press the dot (.) along with the item variable and choose the ToString method:



Hit the Run button to see the result. All the values will pop up at once

Data table

A data table is a tabular form of data structure. It contains rows and each row has columns, for example:

Student name	Roll number	Class
Andrew Jose	1	3
Jorge Martinez	2	3
Stephen Cripps	3	2

- The preceding illustration is an example of a data table that has three rows and three columns.
- For example, you have to build a table dynamically. You can use a data table as your preferred choice.
- A data table is also extensively used to store tabular data structures. In data scraping, data tables are widely used.

- Data scraping is a method in which we can dynamically create tabular data records of search items on the web.

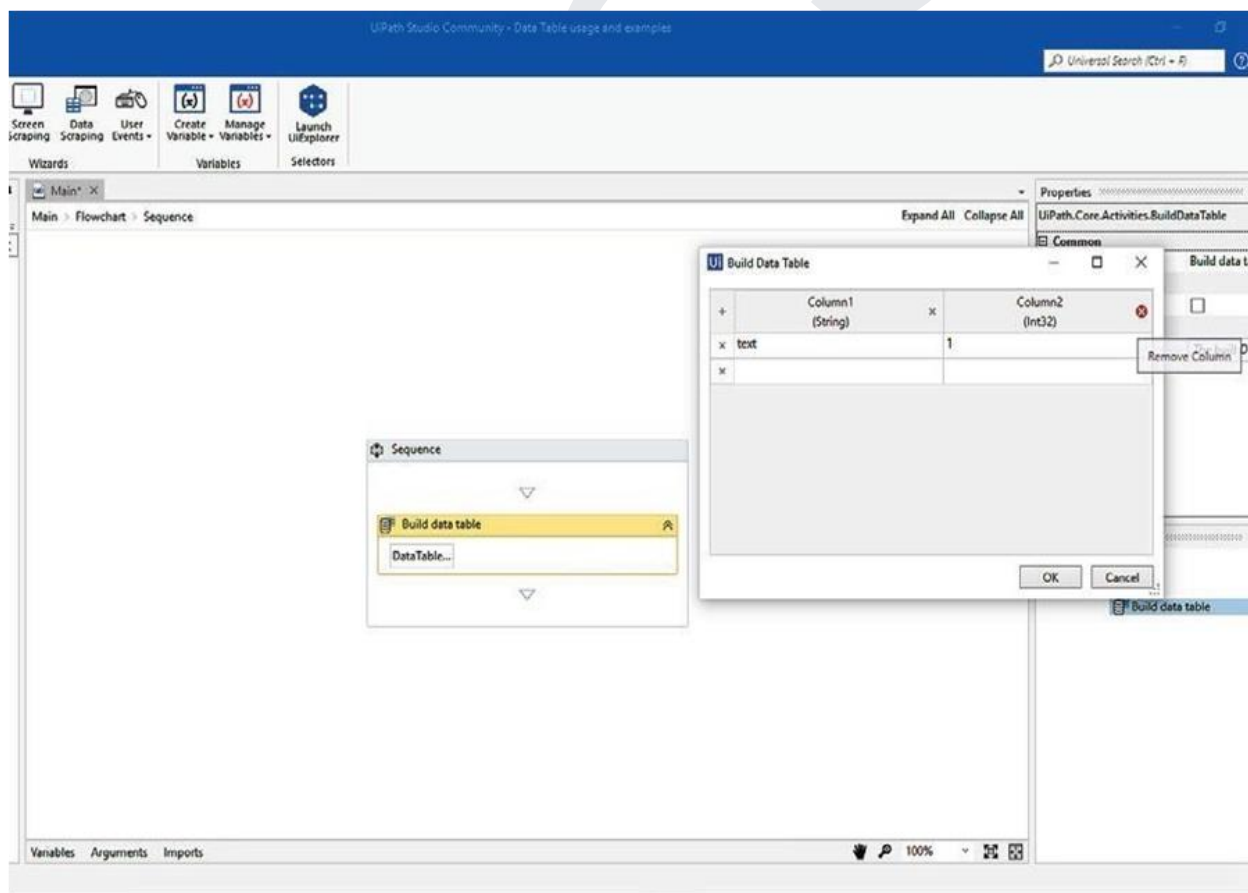
- We shall build two projects in which we will use a data table:

Building a data table.

Building a data table using data scraping (dynamically).

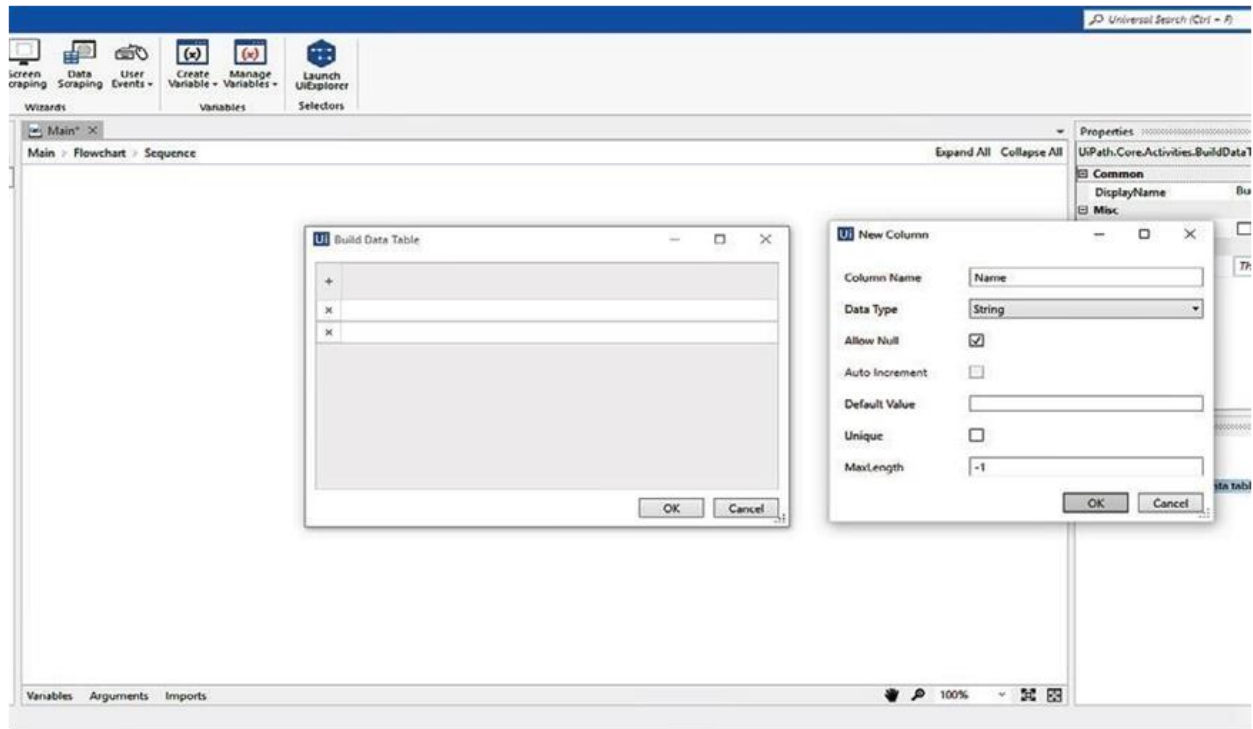
Building a data table

- Create an empty project. Give it a proper name:
 1. Drag and drop a Flowchart activity on the Designer panel. Then, drag and drop a Sequence activity and set it as the Start node.
 2. Double click on the Sequence and drag and drop the Build Data Table activity inside the Sequence activity.
 3. Click on the Data Table button. A pop-up window will appear on the screen. Remove both the columns (auto generated by the Build Data Table activity) by clicking on the Remove Column icon:

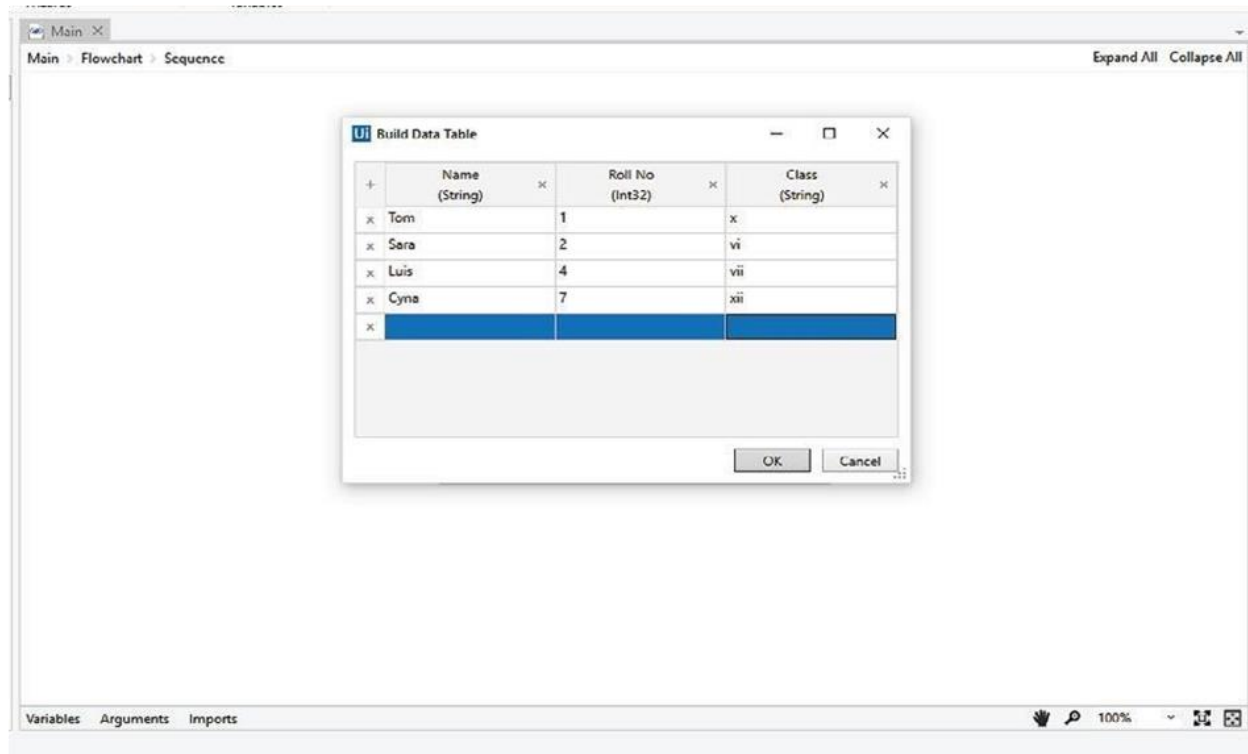


- Now, we will add three columns by simply clicking on the + symbol. Specify the Column names and select the appropriate data types from the drop-down list.

Click on the OK button. We will add column Name of String Data Type, ROLL_NO of Int32 type and finally Class of string type:

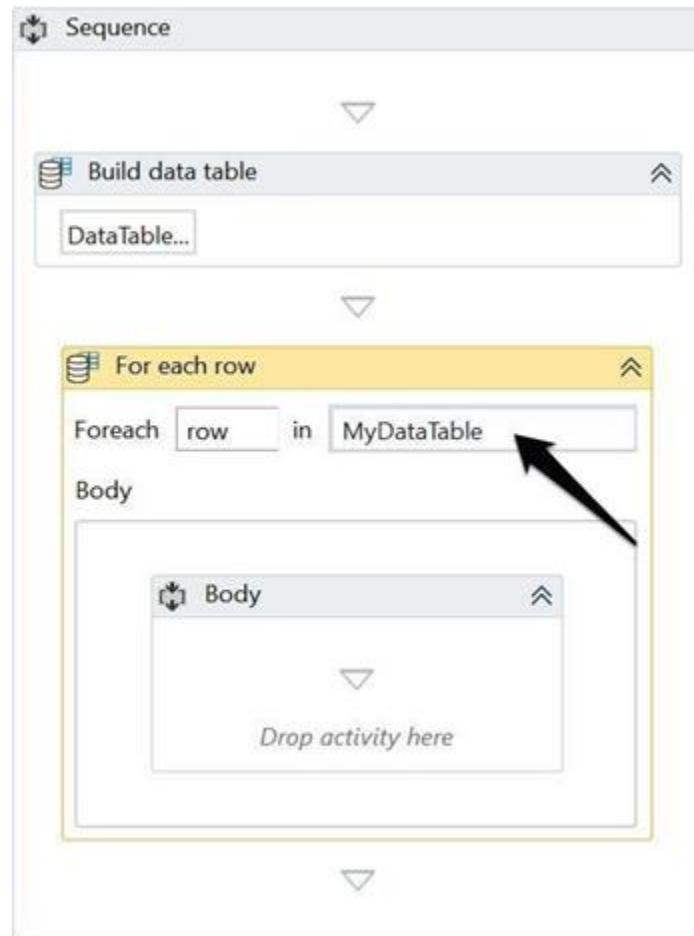


Now enter some random values just to insert the data into the rows:

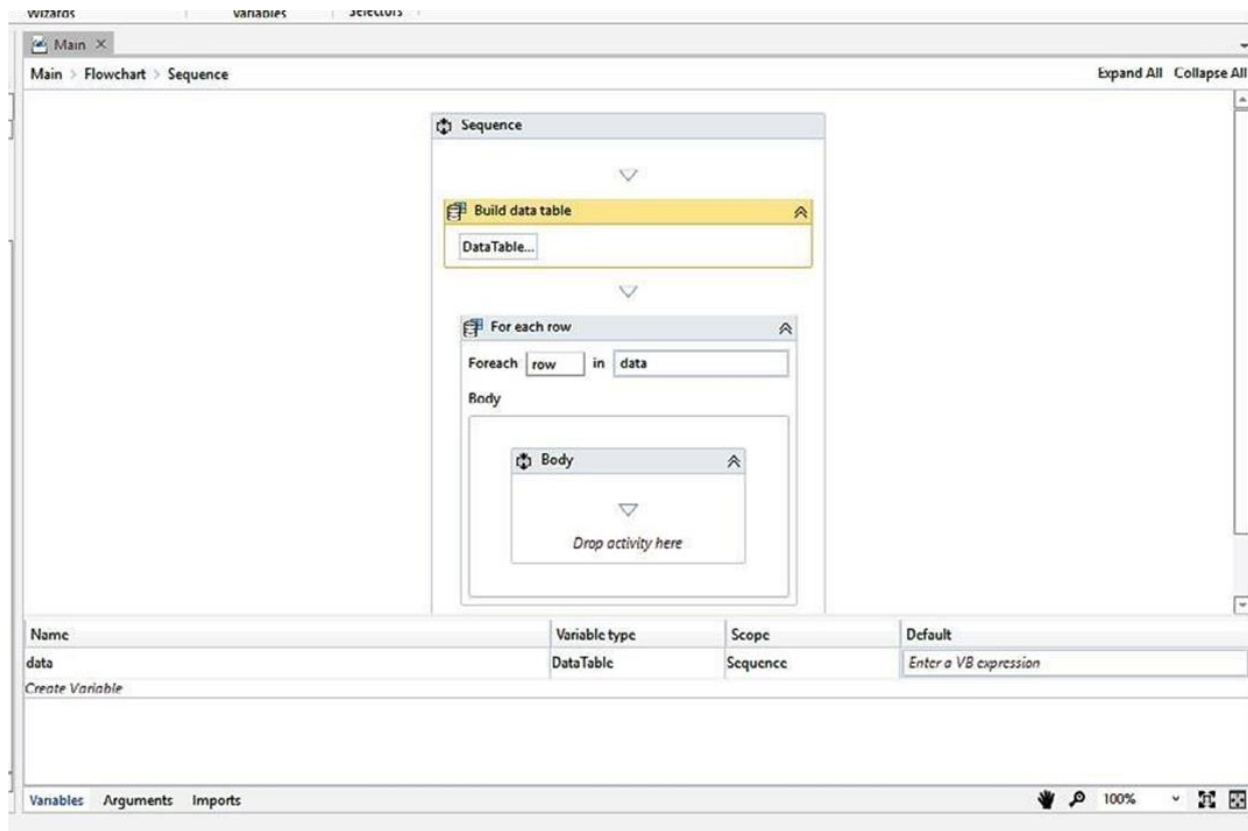


Click on the OK button and our data table is ready. We must iterate over the data table's rows to ensure everything works correctly.

5. To store the Data Table created by the Build Data Table activity, we have to create a data table variable MyDataTable of Data Table type and store the result of the data table that we have dynamically built. Also, specify and assign the Output property of the Build Data Table activity with this variable. Specify the data table variable's name there.
6. After our data table is ready, we will iterate the data table's rows to make sure everything works correctly. Drag and drop the For each row activity from the Activities panel inside the Sequence activity. Specify the data table variable's name (MyDataTable) in the expression text box of the For each row activity:



7. Drag and drop the For each row activity from the Activities panel inside the Sequence activity. Specify the data table variable's name in the expression text box of the For each row activity:



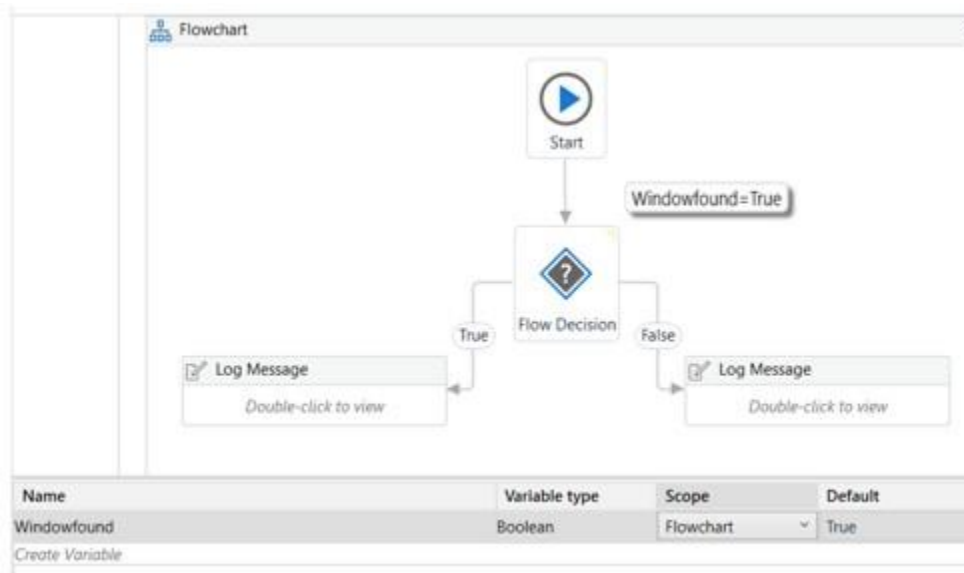
Data Manipulation

Data Manipulation includes dealing with various variables and performing operations with them. UiPath Studio offers a wide range of variables to support automation: –

- **Text variables:** – This type of variable can store only string data type and is enclosed in double quotes. You can define the type as “System.String” to declare this kind of variable.

Name	Variable type	Scope	Default
EmployeeName	String	Sequence	"ABC"
Create Variable			

- **True or False variables:** – This type of variable stores only two possible values – true or false and supports Boolean data type. You can define the types as “System.Boolean” to declare this kind of variable. Make use of this data type when working with flow decisions in Studio.

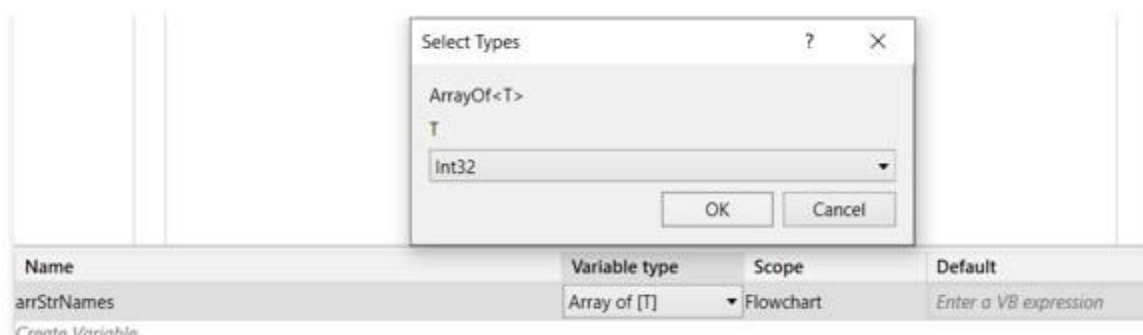


Number variables: – This type of variable stores numeric data and supports Integer values. You can define the type as “Int32” to declare this kind of variable. Use the “.ToString” method to convert the integer variable to a string and display as such.

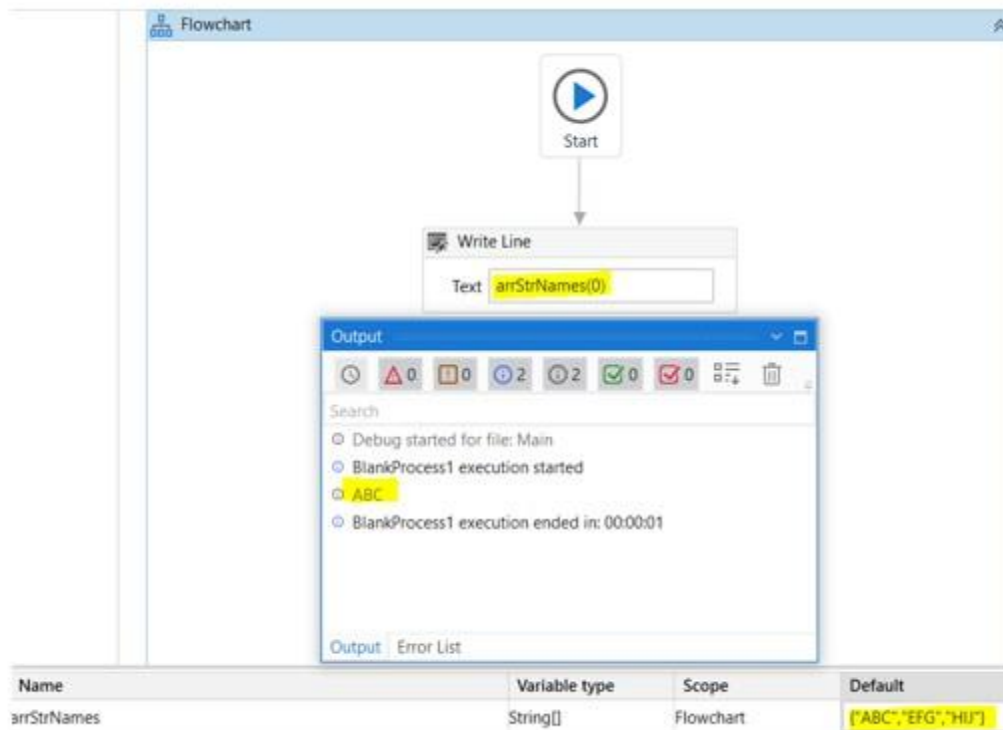
Name	Variable type	Scope	Default
EmployeeID	Int32	Flowchart	35

Create Variable

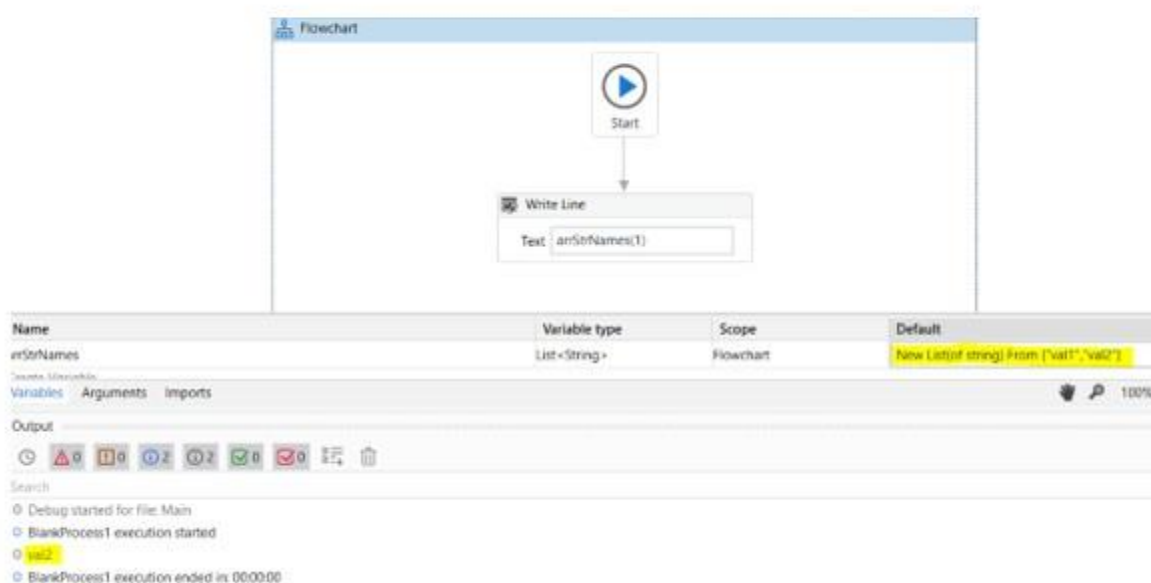
- **Array variables:** – This type of variable will allow you to store a fixed number of multiple values of the same data type, i.e. you can declare an array of number variables (System.Int32[]), an array of string variables (System.String[]), an array of Boolean variables (System.Boolean[]) and so on. Use the Select Types window to declare an array of your choice.



To access all the values you store in an array, give a reference of index in the output. For example, to retrieve the first name from the arrStrNames variable, write line as arrStrNames(0)

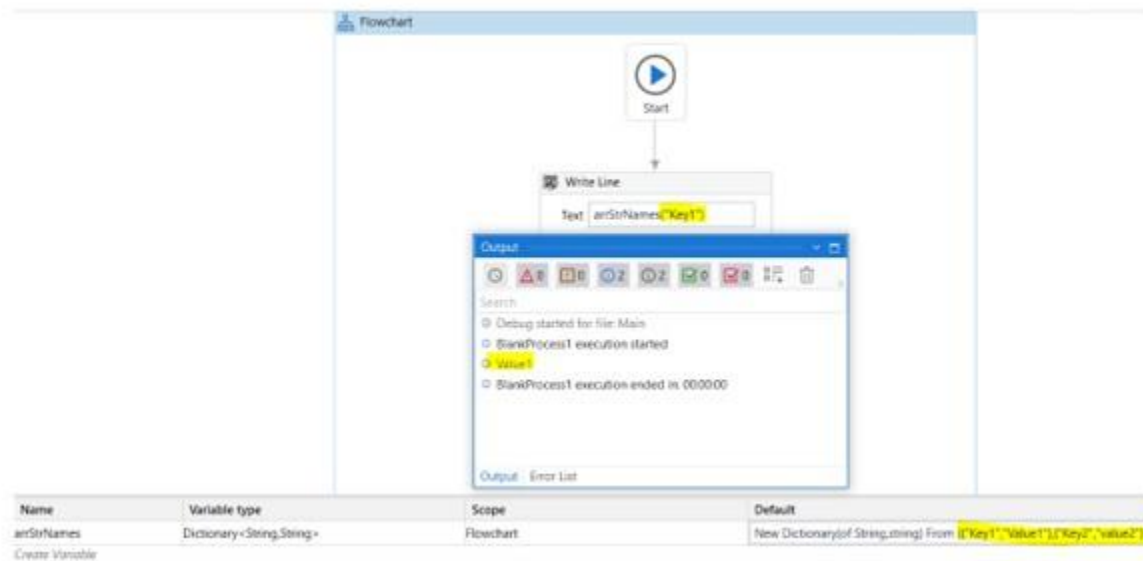


List variables: – This type of variable will allow you to store a varying number of multiple values of the same data type. For example, you can declare a list of strings as – `System.Collections.Generic.List<System.String>`



Dictionary variables: – A dictionary is an array which can hold a varying number of elements associated with a key-value format. The value is retrieved from the defined key in the variable. To

declare a dictionary variable, use “System.Collections.Generic.Dictionary<System.String,System.String>” as the data type.

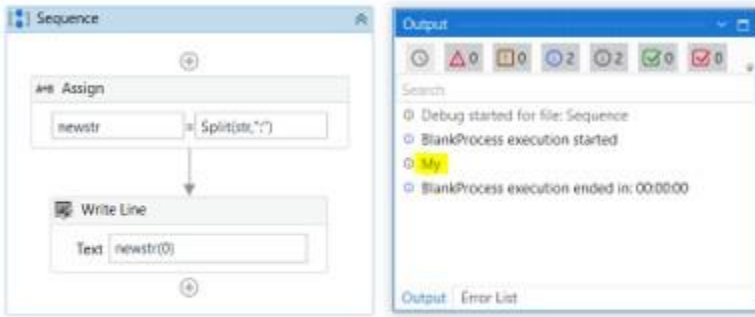


- **Date and Time variables:** – This type of variable stores information about any date and time. Declaration is done by using “System.DateTime” as the data type. The default format used is “day.hh:mm:ss”. You can modify the format to only retrieve a particular component.
- **Data table variables:** – This variable can store information in a database or a simple spreadsheet consisting of rows and columns. Declare this kind of variable by using “System.Data.DataTable”. You can make use of this variable when dealing with excel/workbook automation.
- **Generic-Value variables:** – This variable can store any kind of data such as numbers, text, dates, arrays, etc. Use this data type when unsure of the exact structure of data you are dealing with. Declare using “UiPath.Core.GenericValue” data type.

String Manipulation in UiPath

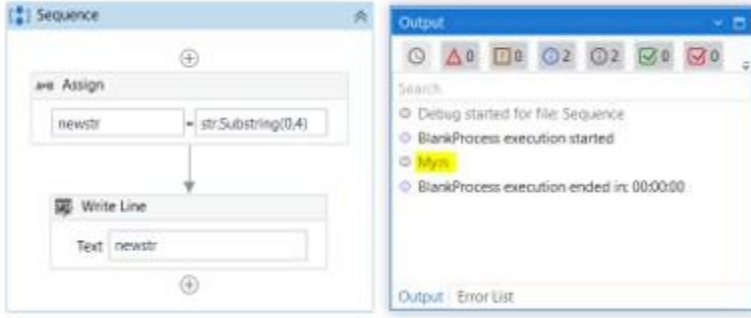
UiPath offers a wide range of functions that can be used to perform operations on strings. Details to each are as below:

- **Split:** This function will divide the string into two parts based on the split identifier mentioned. For an example, take a string “str” as “My:name:is” and define the split function as – “Split(str, “:”)”. This will split the string in three different sections based on “:” and to retrieve any one section, you can refer to the output by an index as done in an array.



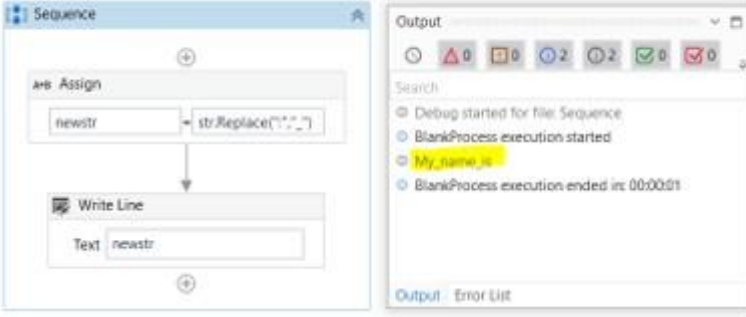
Name	Variable type	Scope	Default
str	GenericValue	Sequence	"Myname:is"
newstr	String[]	Sequence	Enter a VB expression

- **Substring:** This function lets you customize your division of the entire string.



Name	Variable type	Scope	Default
str	GenericValue	Sequence	"Myname:is"
newstr	String	Sequence	Enter a VB expression

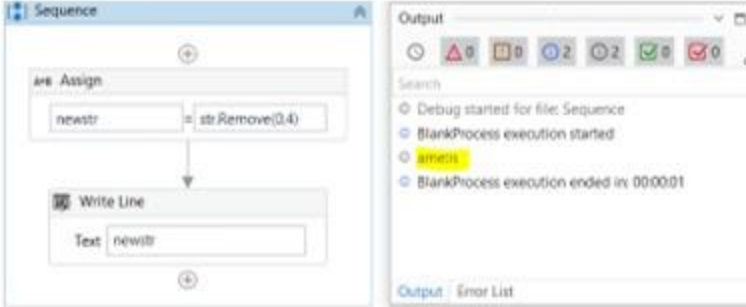
- **Trim:** This function removes the leading spaces (Ltrim), trailing spaces(RTrim) or leading and trailing spaces (Trim) from a string. Use this function as – Trim(yourstring), LTrim(yourstring), RTrim(yourstring).
- **Replace:** This function allows you to replace a character with another character specified. Here “:” gets replaced with “_”



The screenshot shows a sequence editor with two steps: 'Assign' and 'Write Line'. The 'Assign' step uses the 'str.Replace' function to replace spaces with underscores in the variable 'newstr'. The 'Write Line' step outputs the value of 'newstr'. The output window shows the execution log, including 'Debug started for file: Sequence', 'BlankProcess execution started', 'My_name_is', and 'BlankProcess execution ended in: 00:00:01'.

Name	Variable type	Scope	Default
str	Generic Value	Sequence	"Mynameis"
newstr	String	Sequence	Enter a VB expression

Remove: This function removes the text specified. It can be understood as an inverse operation to substring.

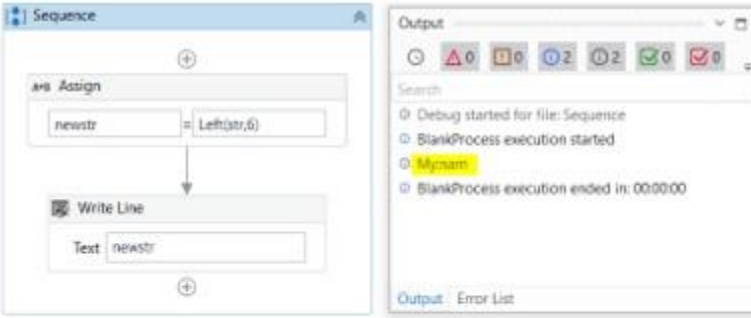


The screenshot shows a sequence editor with two steps: 'Assign' and 'Write Line'. The 'Assign' step uses the 'str.Remove' function to remove the first four characters (0,4) from the variable 'str'. The 'Write Line' step outputs the value of 'newstr'. The output window shows the execution log, including 'Debug started for file: Sequence', 'BlankProcess execution started', 'ameis', and 'BlankProcess execution ended in: 00:00:01'.

Name	Variable type	Scope	Default
str	String	Sequence	"Mynameis"
newstr	String	Sequence	Enter a VB expression

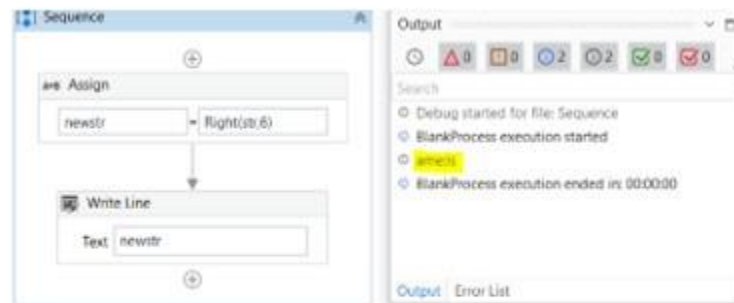
- **Left:** This function outputs the text from the left side of the variable value.





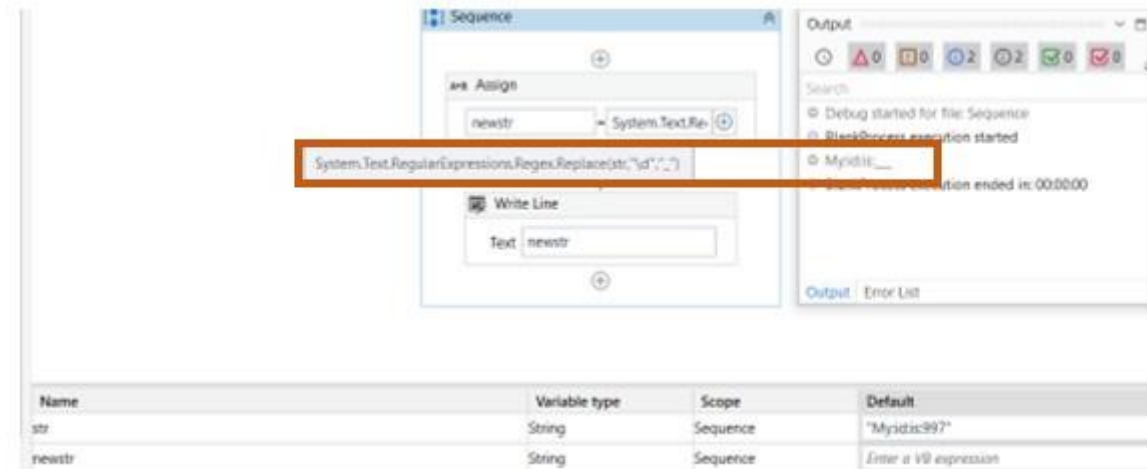
Name	Variable type	Scope	Default
str	String	Sequence	"Myname"
newstr	String	Sequence	Enter a VB expression

- **Right:** This function outputs the text from the right side of the variable value.



Name	Variable type	Scope	Default
str	String	Sequence	"Myname"
newstr	String	Sequence	Enter a VB expression

- **Regex.Replace:** This function will replace an expression combination according to the regex specified with a character. Here, “\d” is used as a regular expression to find digit combinations only and then they are replaced by an underscore “_”.



Gathering and Assembling Data

Data sources: The first step is to identify the data sources that you need to gather. This could include databases, spreadsheets, websites, and other applications.

Data formats: Once you know the data sources, you need to determine the format of the data. This could be structured data, unstructured data, or semi-structured data.

Data extraction: The next step is to extract the data from the data sources. This can be done using a variety of techniques, such as APIs, web scraping, and file parsing.

Data cleansing: Once the data is extracted, it needs to be cleansed. This means removing errors, inconsistencies, and duplicate records.

Data formatting: The data may also need to be formatted. This means changing the way that the data is displayed, such as changing the date format or the currency format.

Data aggregation: The data may also need to be aggregated. This means combining data from multiple sources into a single dataset.

Data storage: The data needs to be stored in a secure location. This could be a database, a spreadsheet, or a cloud storage service.

Gathering and assembling data is an important part of RPA. By understanding these concepts, you can create RPA bots that can gather and assemble data more efficiently and effectively.

Here are some additional tips for gathering and assembling data in RPA:

- Use the right tools for the job. There are many different tools that can be used to gather and assemble data. Choose the tools that are right for your needs.
- Be careful about data security. Data is a valuable asset, so it is important to protect it. Take steps to ensure that your data is secure.
- Be mindful of data privacy. When gathering and assembling data, it is important to be mindful of data privacy. Only gather and assemble data that you have permission to gather and assemble.

By following these tips, you can gather and assemble data in RPA in a way that is efficient, effective, and secure.

VIT CSE