LAB/PRACTICE EXERCISE MANUAL

List of Exercises

Table of Contents

Lesson 2 – Introduction to UiPath

2.1 Downloading and Installing UiPath Academic Alliance Edition 2020.10

About UiPath Academic Alliance Edition

UiPath Learning offers its partner organizations the opportunity to use the UiPath Studio software at no charge through a dedicated Learning license, which is free to students and educators alike, for teaching, learning, and research.

- The license entitles the user to access UiPath Studio and UiPath StudioX, the automation development software application, along with the UiPath Development Robot to allow for testing and execution of the automation.
- The Academic Alliance Edition of UiPath Studio is a locked version of the software aligned to the current Studio course resources provided by the UiPath Academic Alliance Partners and UiPath Learning Partners. Moreover, the current UiPath Certified Professional exams are based on this version.
- One license key allows for the activation of the UiPath Studio software on 2 different machines.
- The license type granted is Named User, to be applied solely to User Mode installs. Instructions for activating the Academic Alliance Edition license will be provided accordingly.
- Each Academic Alliance Edition license expires after 365 days upon its activation, and it is subject to being extended as long as the Academic Alliance partnership is maintained.

Using the Academic Alliance Edition ensures that you have the appropriate software version installed which aligns with the curricula and hands-on lab materials. This will avoid any disruptions to your learning experience. This will also ensure that projects can be shared between students and educators without software version differences.

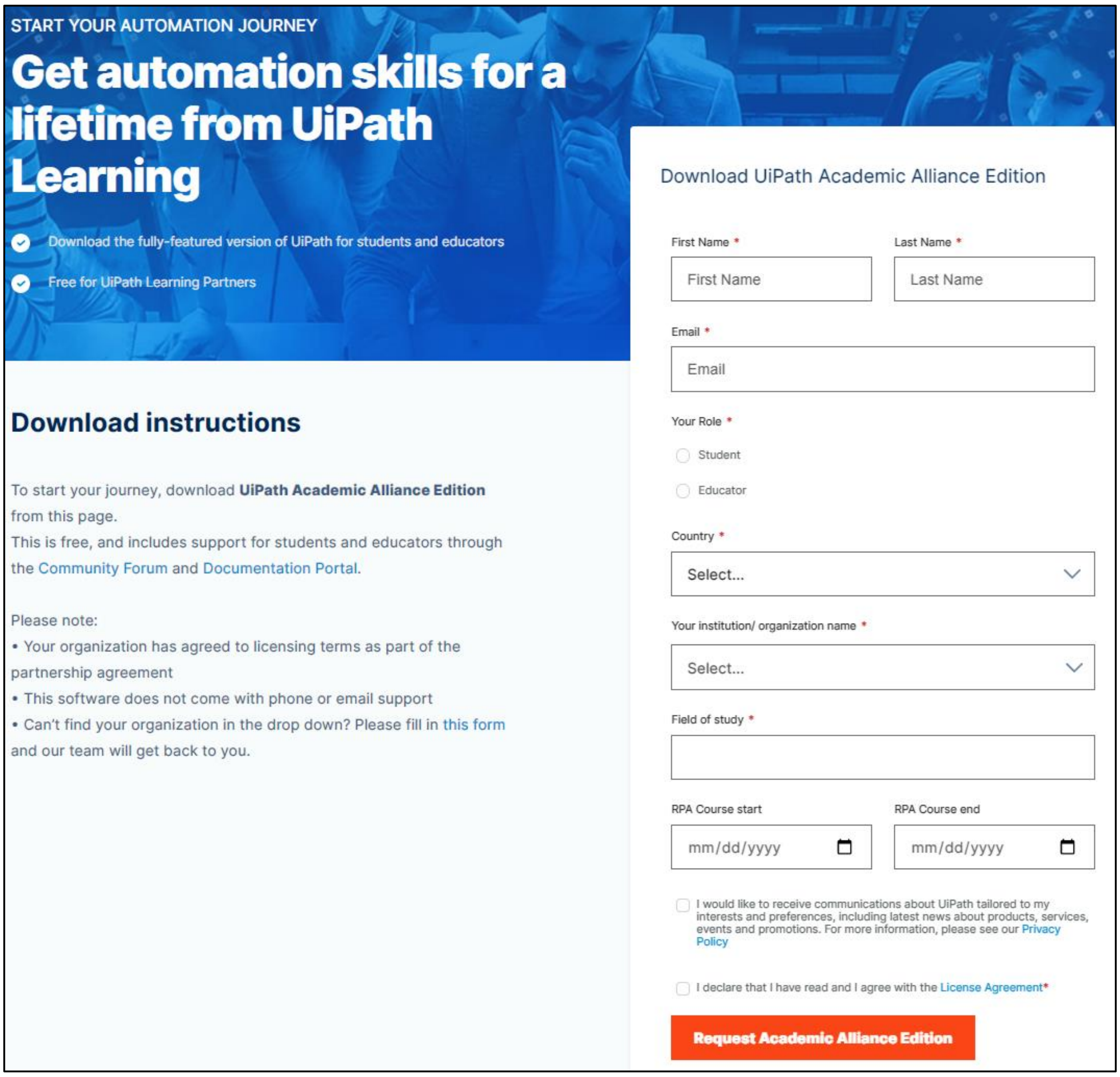
If you have installed UiPath Studio Community Edition previously, please uninstall it first before installing the UiPath Studio Academic Alliance Edition. Failure to do this may result in both editions of the software not working correctly.

If you have previous versions of the Academic Alliance Edition (19.4 or 19.10), please install the new one above it (i.e. no need to uninstall).

Download and Install UiPath Studio Academic Alliance Edition

Step 1: To download your free copy of the UiPath Studio software, open the URL: https://www.uipath.com/landing/academic-studio-download. It is a private form designated only for the students and educators who are affiliated with the UiPath Academic Alliance & UiPath Learning Partner institutions.

Step 2: Fill in your contact details and choose the institution you belong to.



START YOUR AUTOMATION JOURNEY

Get automation skills for a lifetime from UiPath Learning

- Download the fully-featured version of UiPath for students and educators
- Free for UiPath Learning Partners

Download instructions

To start your journey, download **UiPath Academic Alliance Edition** from this page.
This is free, and includes support for students and educators through the Community Forum and Documentation Portal.

Please note:
• Your organization has agreed to licensing terms as part of the partnership agreement
• This software does not come with phone or email support
• Can't find your organization in the drop down? Please fill in this form and our team will get back to you.

Download UiPath Academic Alliance Edition

First Name * | Last Name *
Email *
Your Role * ( ) Student ( ) Educator
Country * Select...
Your institution/ organization name * Select...
Field of study *
RPA Course start mm/dd/yyyy | RPA Course end mm/dd/yyyy

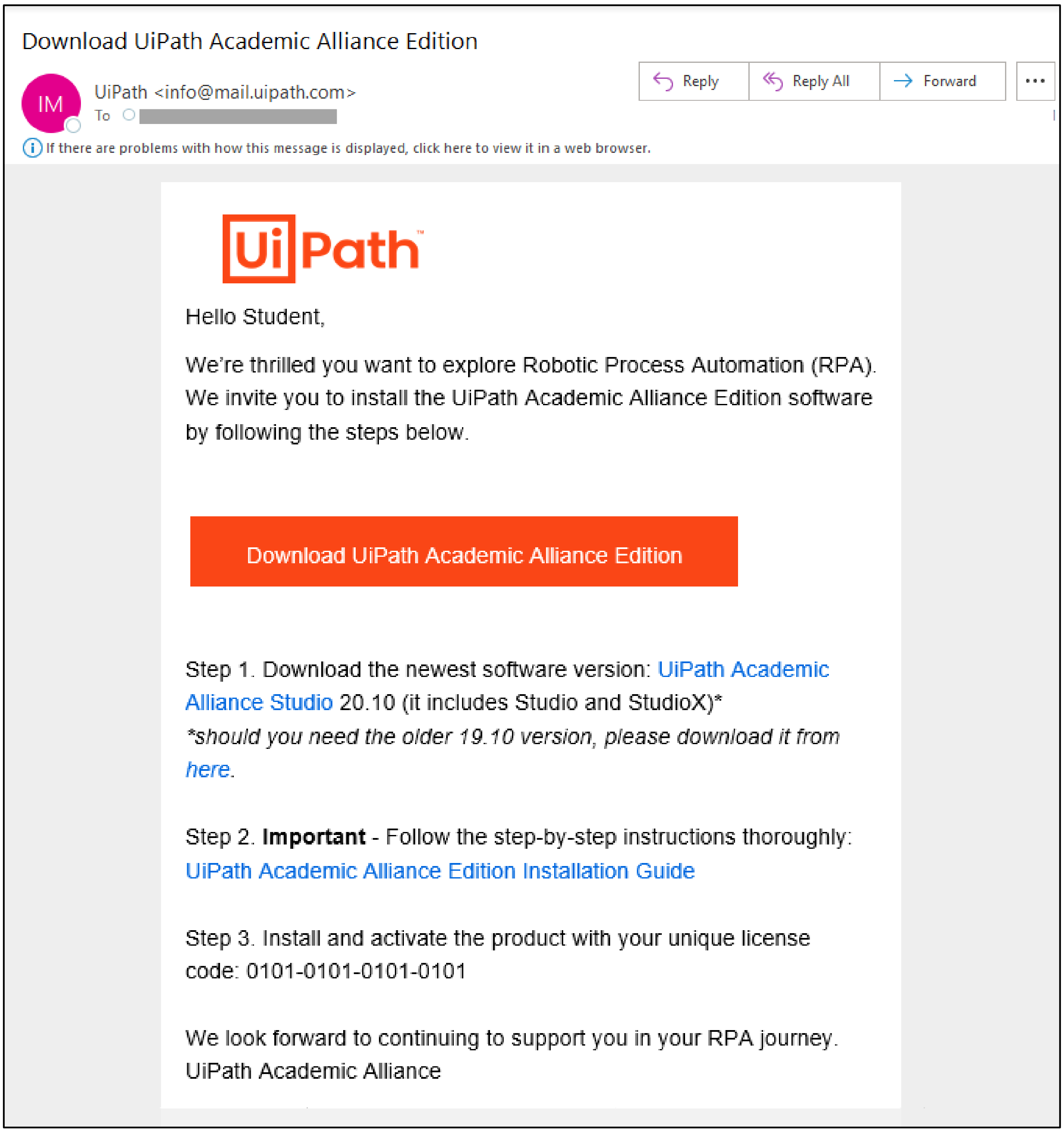
☐ I would like to receive communications about UiPath tailored to my interests and preferences, including latest news about products, services, events and promotions. For more information, please see our Privacy Policy

☐ I declare that I have read and I agree with the License Agreement*

Request Academic Alliance Edition

Step 3: Click **Request Academic Alliance Edition**. Then, check your email in a couple of minutes. You will receive a link to download the software by email. The email you receive contains the installation guide, the link to download the *UiPathStudio.msi* file along your dedicated license key.

Make sure you follow the steps in the email and carefully follow this guide.



Download UiPath Academic Alliance Edition

UiPath <info@mail.uipath.com>
To [redacted]

If there are problems with how this message is displayed, click here to view it in a web browser.

Hello Student,

We're thrilled you want to explore Robotic Process Automation (RPA). We invite you to install the UiPath Academic Alliance Edition software by following the steps below.

Download UiPath Academic Alliance Edition

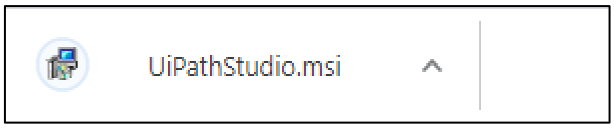
Step 1. Download the newest software version: UiPath Academic Alliance Studio 20.10 (it includes Studio and StudioX)*
should you need the older 19.10 version, please download it from here.

Step 2. **Important** - Follow the step-by-step instructions thoroughly: UiPath Academic Alliance Edition Installation Guide

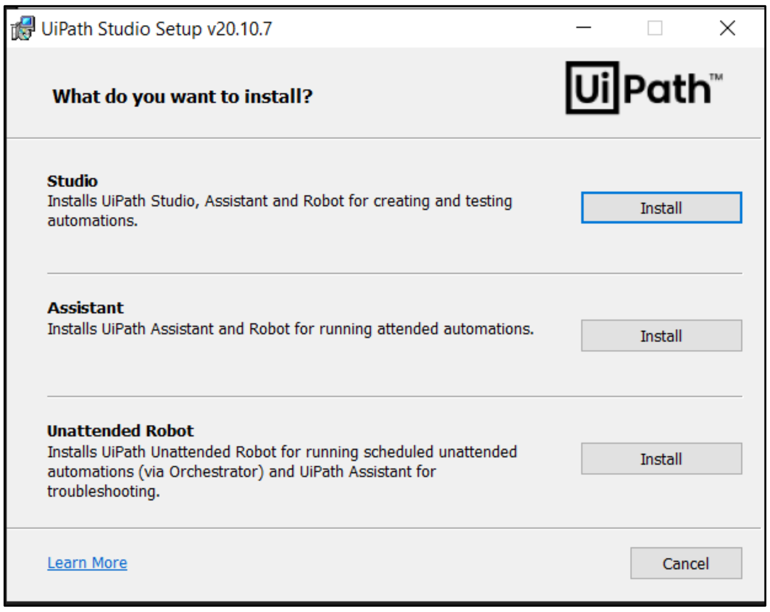
Step 3. Install and activate the product with your unique license code: 0101-0101-0101-0101

We look forward to continuing to support you in your RPA journey.
UiPath Academic Alliance

The file will begin downloading. Click to open the **UiPathStudio.msi** once it is downloaded.



Step 4: Click on the *Install* button right beside *Studio*.



UiPath Studio Setup v20.10.7

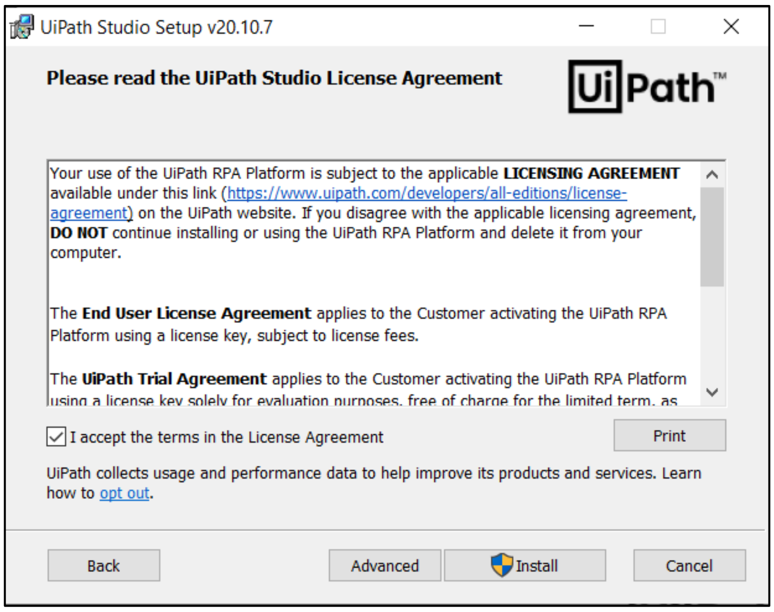
What do you want to install?

Studio
Installs UiPath Studio, Assistant and Robot for creating and testing automations.

Assistant
Installs UiPath Assistant and Robot for running attended automations.

Unattended Robot
Installs UiPath Unattended Robot for running scheduled unattended automations (via Orchestrator) and UiPath Assistant for troubleshooting.

Step 5: Accept the terms in the License Agreement and click on *Install*.



UiPath Studio Setup v20.10.7

Please read the UiPath Studio License Agreement

Your use of the UiPath RPA Platform is subject to the applicable **LICENSING AGREEMENT** available under this link (https://www.uipath.com/developers/all-editions/license-agreement) on the UiPath website. If you disagree with the applicable licensing agreement, **DO NOT** continue installing or using the UiPath RPA Platform and delete it from your computer.

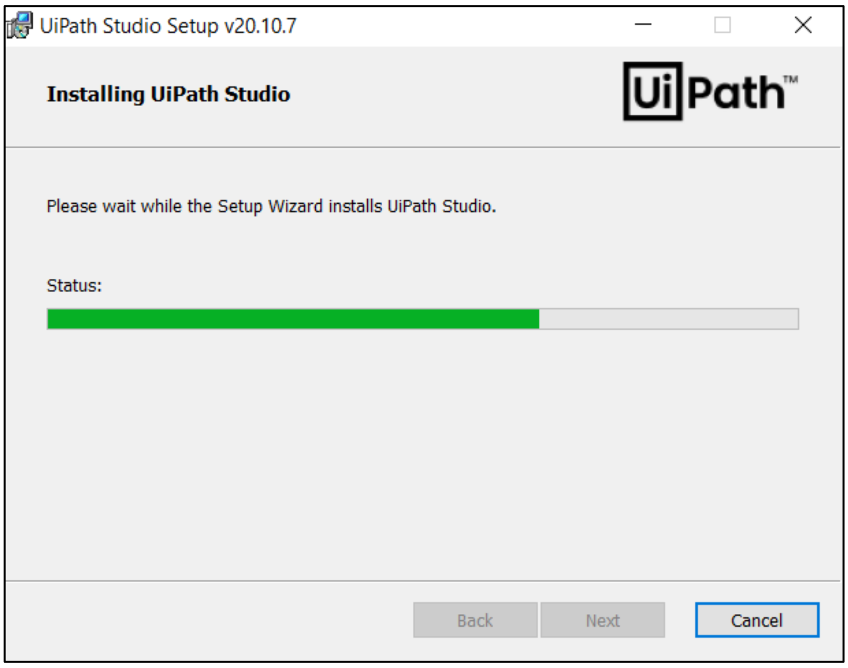
The **End User License Agreement** applies to the Customer activating the UiPath RPA Platform using a license key, subject to license fees.

The **UiPath Trial Agreement** applies to the Customer activating the UiPath RPA Platform using a license key solely for evaluation purposes, free of charge for the limited term, as

☑ I accept the terms in the License Agreement

UiPath collects usage and performance data to help improve its products and services. Learn how to opt out.

Step 6: The software will begin its installation.



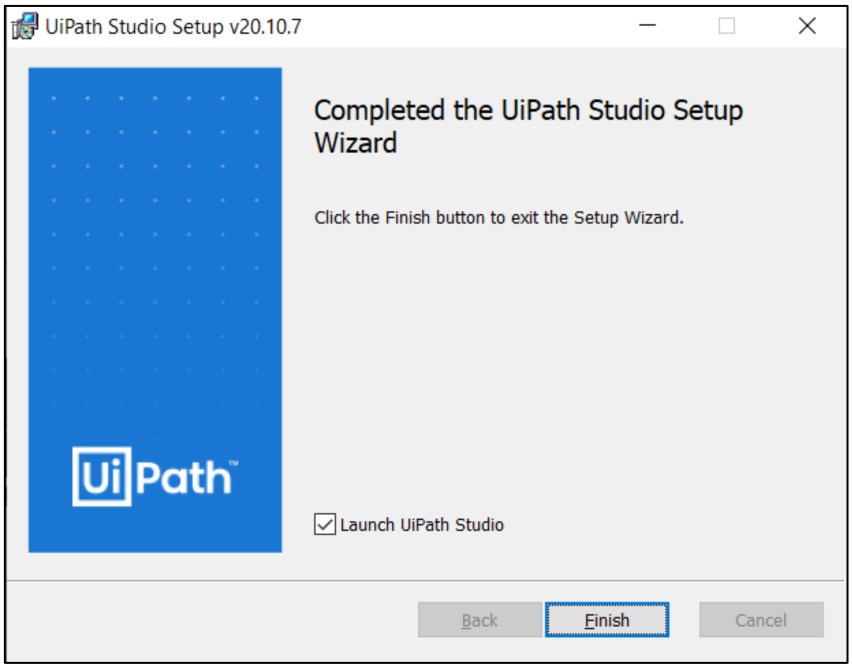
UiPath Studio Setup v20.10.7

Installing UiPath Studio

Please wait while the Setup Wizard installs UiPath Studio.

Status:

Step 7: Click Finish.



UiPath Studio Setup v20.10.7

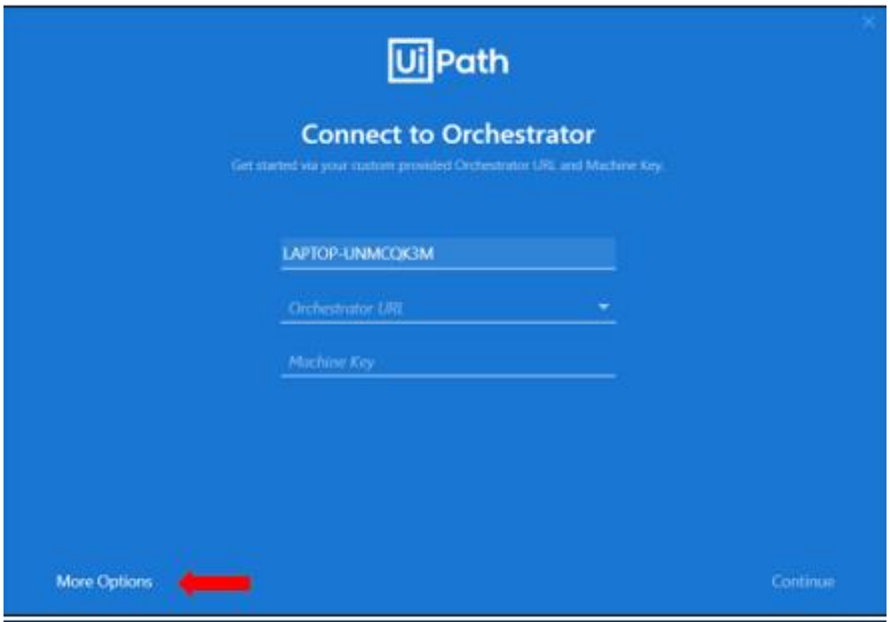
Completed the UiPath Studio Setup Wizard

Click the Finish button to exit the Setup Wizard.

☑ Launch UiPath Studio

If "Launch UiPath Studio" was checked, then UiPath Studio is launched. If not, you can search in your Windows explorer for UiPath Studio to open it.

Step 8: In the “Welcome to UiPath Studio” window, click on *More Options* and then *Standalone Options* to activate your dedicated Academic Alliance Edition of Studio with your license key.



Connect to Orchestrator

Get started via your custom provided Orchestrator URL and Machine Key.

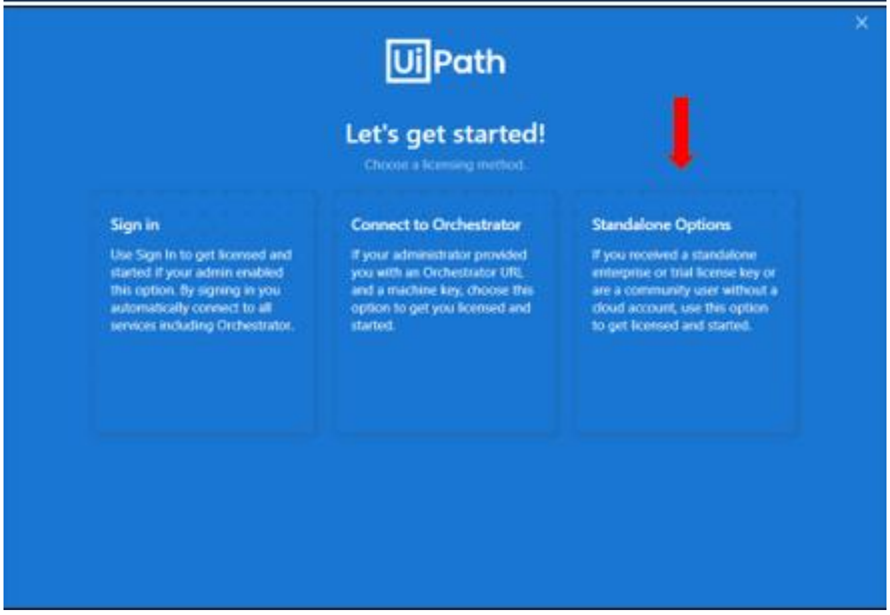
LAPTOP-UNMCQK3M

Orchestrator URL

Machine Key

More Options

Continue



Let's get started!

Choose a licensing method.

Sign in

Use Sign In to get licensed and started if your admin enabled this option. By signing in you automatically connect to all services including Orchestrator.

Connect to Orchestrator

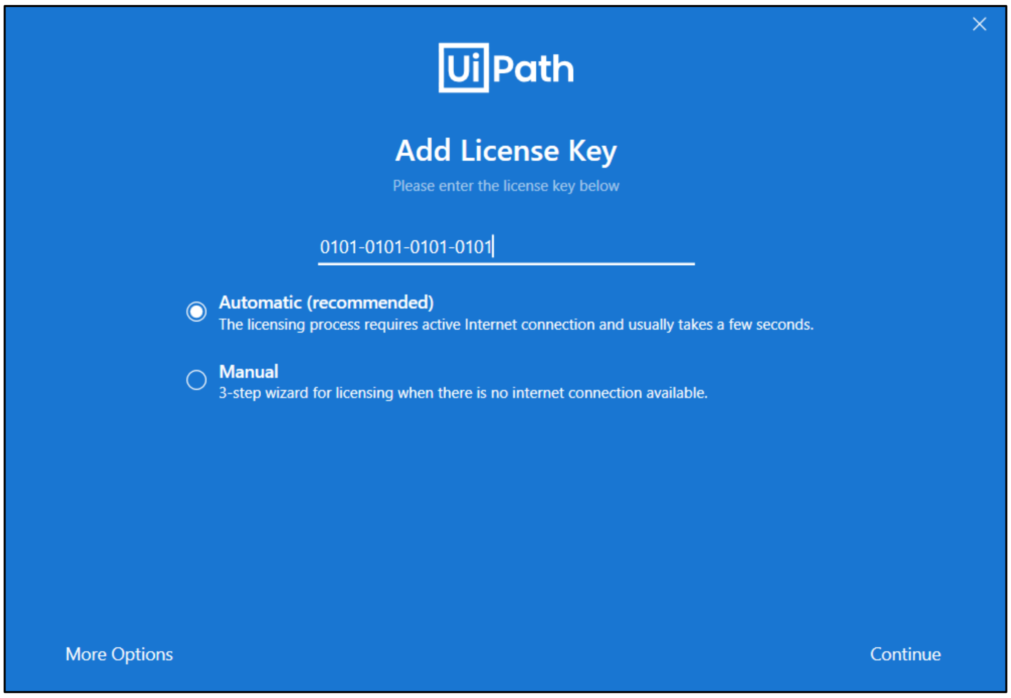
If your administrator provided you with an Orchestrator URL and a machine key, choose this option to get you licensed and started.

Standalone Options

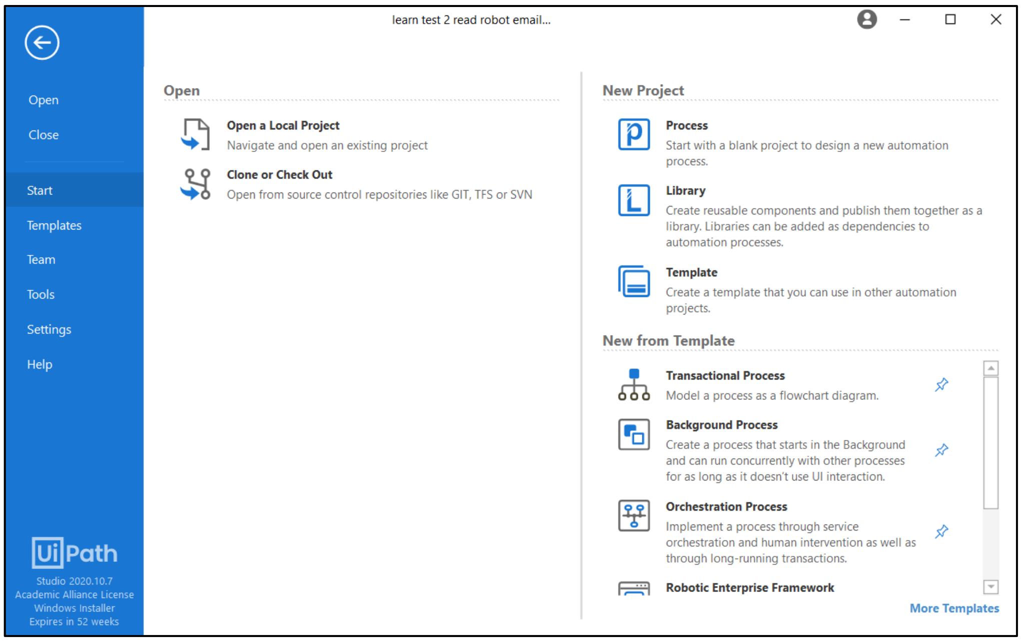
If you received a standalone enterprise or trial license key or are a community user without a cloud account, use this option to get licensed and started.

Step 9: The UiPath Registration window is displayed

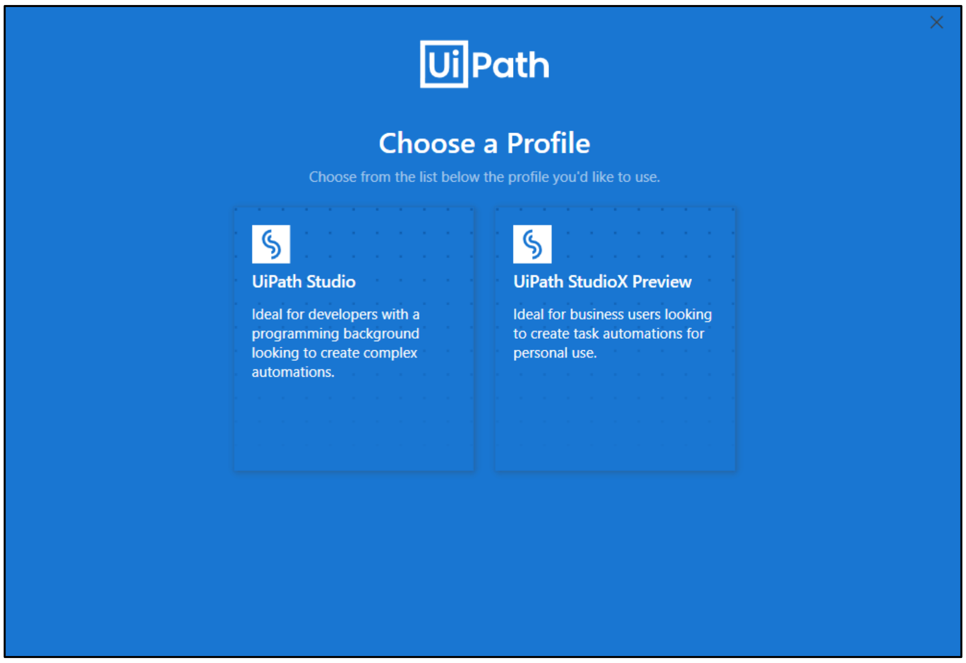
a. Enter the **License Key** you received by email, including the dashes.
b. Select **Automatic** activation, but ensure you have an internet connection.
c. Click **Continue**. Your UiPath license is now activated.


UiPath
Add License Key
Please enter the license key below
0101-0101-0101-0101
Automatic (recommended)
The licensing process requires active Internet connection and usually takes a few seconds.
Manual
3-step wizard for licensing when there is no internet connection available.
More Options
Continue

Step 10: Now, you can start using UiPath Studio and StudioX software to create automation workflows.


learn test 2 read robot email...
Open
Close
Start
Templates
Team
Tools
Settings
Help
UiPath
Studio 2020.10.7
Academic Alliance License
Windows Installer
Expires in 52 weeks
Open
Open a Local Project
Navigate and open an existing project
Clone or Check Out
Open from source control repositories like GIT, TFS or SVN
New Project
Process
Start with a blank project to design a new automation process.
Library
Create reusable components and publish them together as a library. Libraries can be added as dependencies to automation processes.
Template
Create a template that you can use in other automation projects.
New from Template
Transactional Process
Model a process as a flowchart diagram.
Background Process
Create a process that starts in the Background and can run concurrently with other processes for as long as it doesn't use UI interaction.
Orchestration Process
Implement a process through service orchestration and human intervention as well as through long-running transactions.
Robotic Enterprise Framework
More Templates

Step 11: You can switch between the two profiles (Studio or StudioX) at any time by going to Settings – License and Profile – View or Change Profile. Choose whether you want to use UiPath Studio or StudioX.



Choose a Profile

Choose from the list below the profile you'd like to use.

UiPath Studio

Ideal for developers with a programming background looking to create complex automations.

UiPath StudioX Preview

Ideal for business users looking to create task automations for personal use.

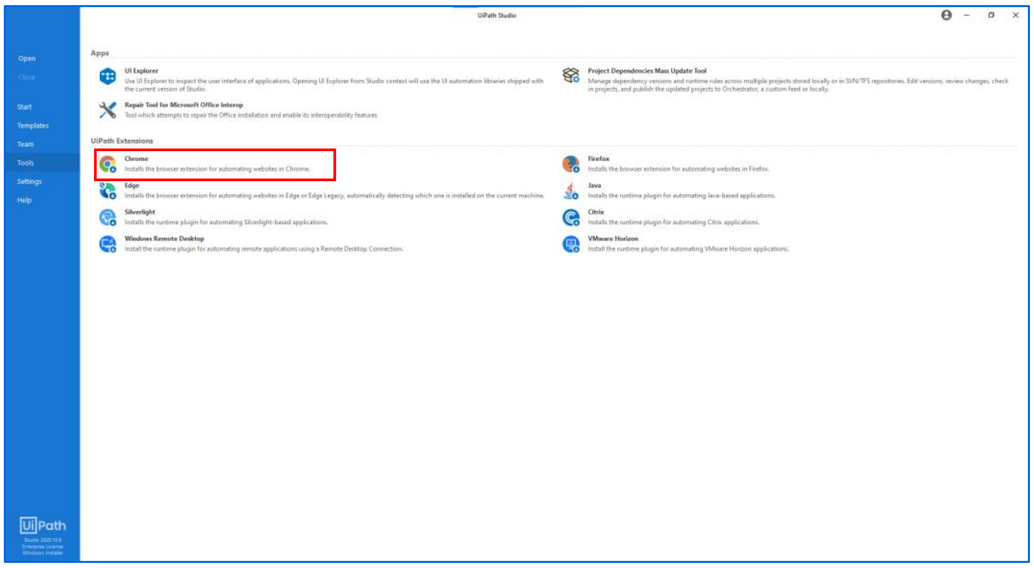
2.2 Installing UiPath Extension in Browsers

Follow the steps mentioned in the sections below to install browser extension of UiPath Studio in Chrome and Firefox browser.

2.2.1 Chrome Browser

Follow the below steps to install UiPath Extension on Chrome Browser:

Step 1: Click the *Home* button, and go to the *Tools* tab from the backend screen. Click on the "Chrome Extension" button. After that, you need to perform two more steps to activate the extension.



Apps

UI Explorer
Use UI Explorer to inspect the user interface of applications. Opening UI Explorer from Studio context will use the UI automation libraries shipped with the current version of Studio.

Repair Tool for Microsoft Office Interop
Tool which attempts to repair the Office installation and enable its interoperability features

Project Dependencies Mass Update Tool
Manage dependency versions and runtime rules across multiple projects stored locally or in SVN/TFS repositories. Edit versions, review changes, check in projects, and publish the updated projects to Orchestrator, a custom feed or locally.

UiPath Extensions

Chrome
Installs the browser extension for automating websites in Chrome.

Edge
Installs the browser extension for automating websites in Edge or Edge Legacy, automatically detecting which one is installed on the current machine.

Silverlight
Installs the runtime plugin for automating Silverlight-based applications.

Windows Remote Desktop
Install the runtime plugin for automating remote applications using a Remote Desktop Connection.

Firefox
Installs the browser extension for automating websites in Firefox.

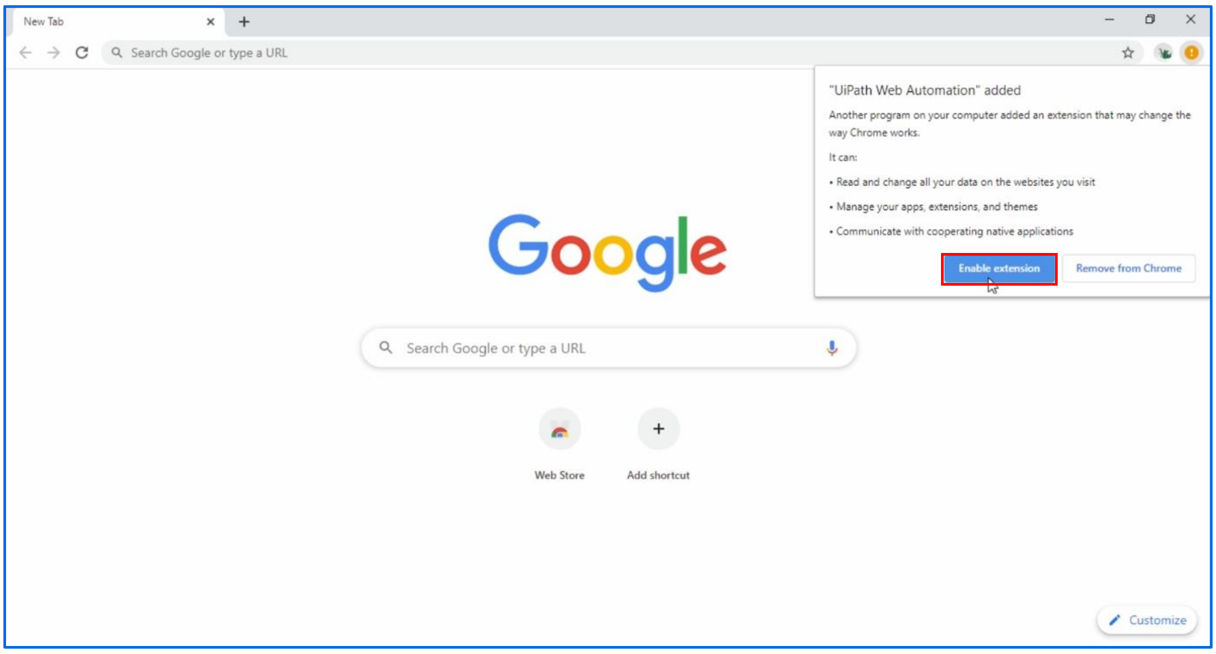
Java
Installs the runtime plugin for automating Java-based applications.

Citrix
Installs the runtime plugin for automating Citrix applications.

VMware Horizon
Install the runtime plugin for automating VMware Horizon applications.

Step 2: Open the Chrome browser and click on the notification, as shown on the screen below. This will enable the UiPath Web Automation extension.

Step 3: You will get a prompt as shown in the screenshot below, click on *Enable Extension* button and you are done.


"UiPath Web Automation" added

Another program on your computer added an extension that may change the way Chrome works.

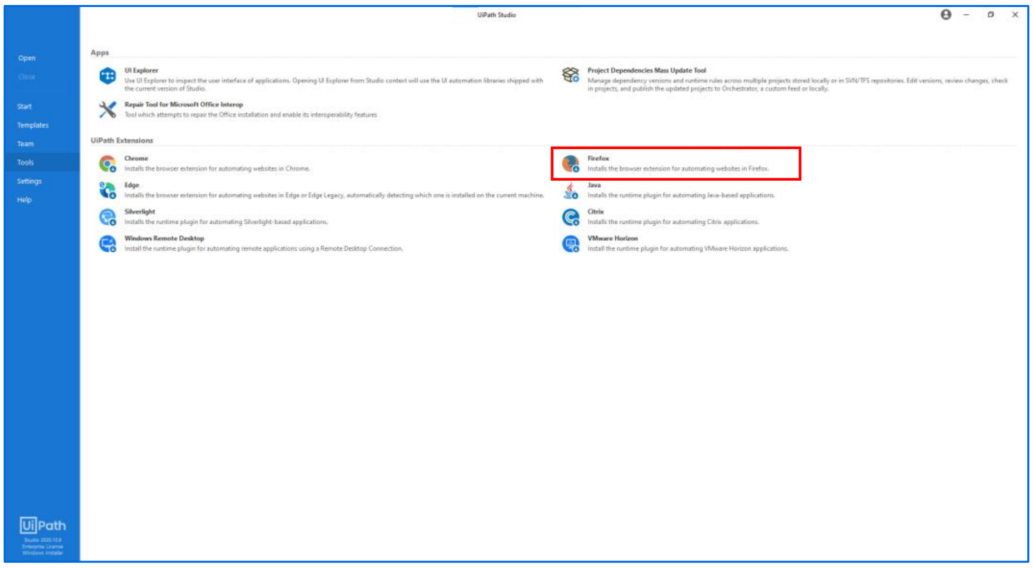
It can:

- Read and change all your data on the websites you visit
- Manage your apps, extensions, and themes
- Communicate with cooperating native applications

2.2.2 Firefox Browser

Follow the below steps to install UiPath Extension on Firefox Browser:

Step 1: Click the *Home* button, and go to the *Tools* tab from the backend screen. Click on the “Firefox Extension” button as shown in the image below:


Apps

UI Explorer – Use UI Explorer to inspect the user interface of applications. Opening UI Explorer from Studio context will use the UI automation libraries shipped with the current version of Studio.

Repair Tool for Microsoft Office Interop – Tool which attempts to repair the Office installation and enable its interoperability features

Project Dependencies Mass Update Tool – Manage dependency versions and runtime rules across multiple projects stored locally or in SVN/TFS repositories. Edit versions, review changes, check in projects, and publish the updated projects to Orchestrator, a custom feed or locally.

UiPath Extensions

Chrome – Installs the browser extension for automating websites in Chrome.

Edge – Installs the browser extension for automating websites in Edge or Edge Legacy, automatically detecting which one is installed on the current machine.

Silverlight – Installs the runtime plugin for automating Silverlight-based applications.

Windows Remote Desktop – Install the runtime plugin for automating remote applications using a Remote Desktop Connection.

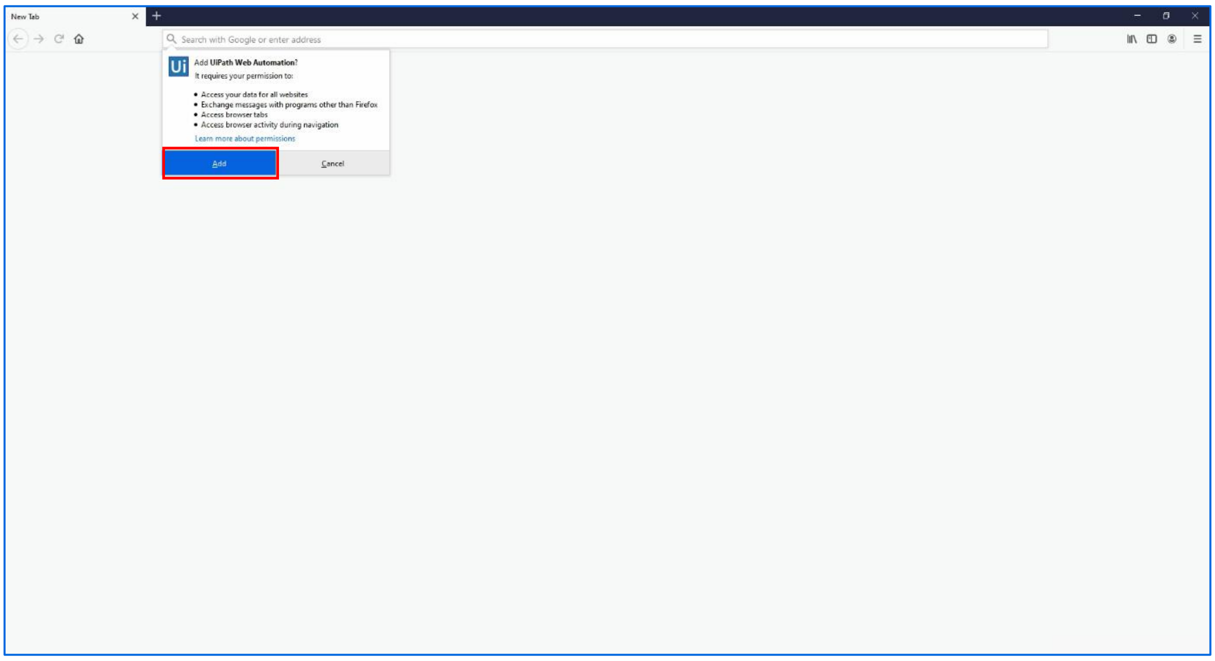
Firefox – Installs the browser extension for automating websites in Firefox.

Java – Installs the runtime plugin for automating Java-based applications.

Citrix – Installs the runtime plugin for automating Citrix applications.

VMware Horizon – Install the runtime plugin for automating VMware Horizon applications.

Step 2: Now open the Firefox browser and click on the *Ok* button. It will redirect you to a new page in Firefox. Click on the *Add* button to finish the installation. Now the extension is enabled for the Firefox browser.


Add **UiPath Web Automation?**
It requires your permission to:

- Access your data for all websites
- Exchange messages with programs other than Firefox
- Access browser tabs
- Access browser activity during navigation

Learn more about permissions

2.2.3 Internet Explorer

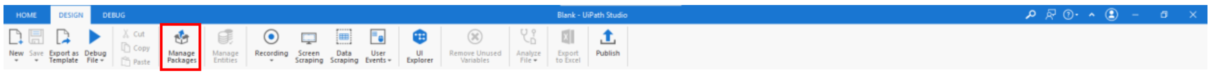
UiPath Studio is compatible with Internet Explorer by default. Thus, no additional browser extension is required to automate the workflow if you are using this browser.

2.3 Installing Activity Packages in UiPath Studio

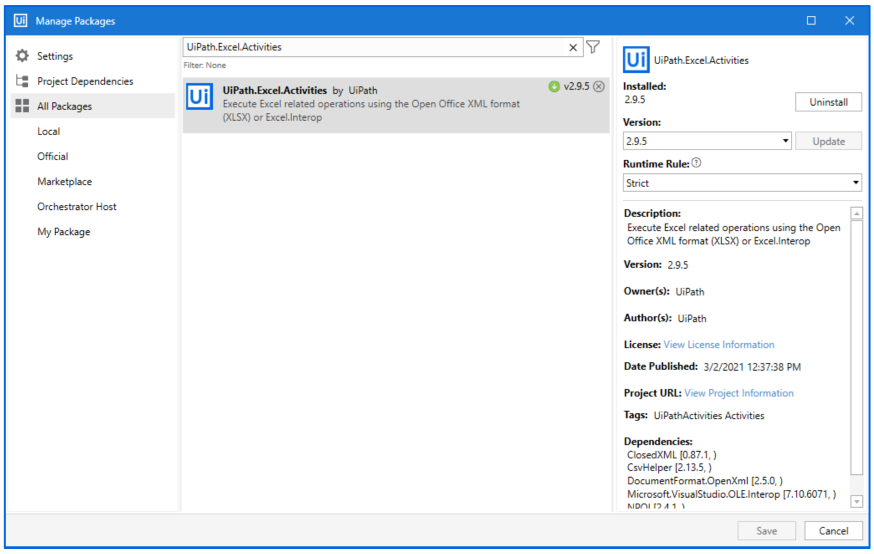
Follow the steps mentioned in the sections below to install activity packages in UiPath Studio for Excel, Email, and PDF. You can follow similar steps to install other packages as well.

2.3.1 Excel Activity Package Installation

Step 1: Click on the Manage Packages button from the Design ribbon.



Step 2: Search for **UiPath.Excel.Activities** in the Manage Packages window


Manage Packages

Settings
Project Dependencies
All Packages
Local
Official
Marketplace
Orchestrator Host
My Package

UiPath.Excel.Activities
Filter: None

UiPath.Excel.Activities by UiPath v2.9.5
Execute Excel related operations using the Open Office XML format (XLSX) or Excel.Interop

UiPath.Excel.Activities
Installed: 2.9.5 Uninstall
Version: 2.9.5 Update
Runtime Rule: Strict
Description: Execute Excel related operations using the Open Office XML format (XLSX) or Excel.Interop
Version: 2.9.5
Owner(s): UiPath
Author(s): UiPath
License: View License Information
Date Published: 3/2/2021 12:37:38 PM
Project URL: View Project Information
Tags: UiPathActivities Activities
Dependencies:
ClosedXML [0.87.1,)
CsvHelper [2.13.5,)
DocumentFormat.OpenXml [2.5.0,)
Microsoft.VisualStudio.OLE.Interop [7.10.6071,)

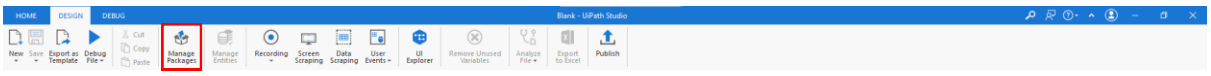
Save Cancel

Step 3: Select the version number from the drop-down menu. Choose the latest version if you are not sure which version to pick. After selecting the version number, click Install.

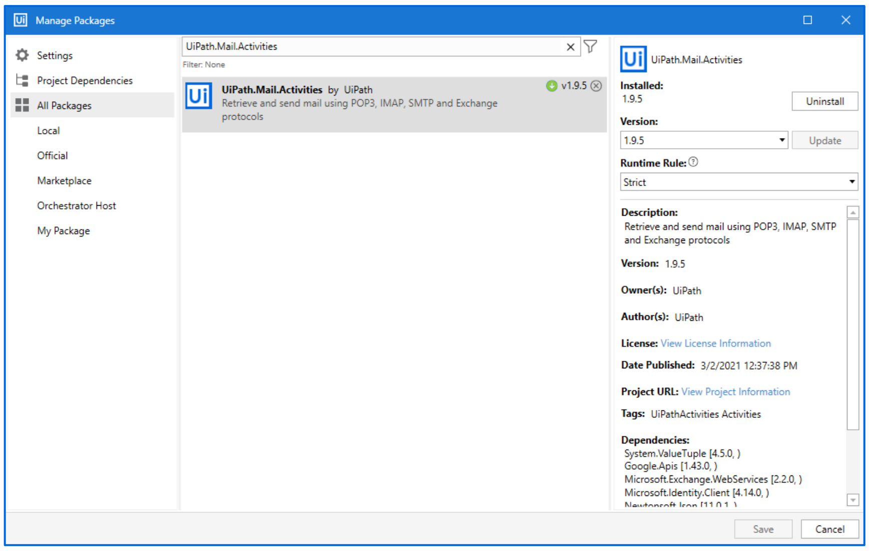
Step 4: Save the packages by clicking on the “Save” button.

2.3.2 Email Activity Package installation

Step 1: Click on the Manage Packages button from the Design ribbon.



Step 2: Search for **UiPath.Mail.Activities** in the Manage Packages window


Manage Packages

Settings
Project Dependencies
All Packages
Local
Official
Marketplace
Orchestrator Host
My Package

UiPath.Mail.Activities
Filter: None

UiPath.Mail.Activities by UiPath v1.9.5
Retrieve and send mail using POP3, IMAP, SMTP and Exchange protocols

UiPath.Mail.Activities
Installed: 1.9.5 Uninstall
Version: 1.9.5 Update
Runtime Rule: Strict
Description: Retrieve and send mail using POP3, IMAP, SMTP and Exchange protocols
Version: 1.9.5
Owner(s): UiPath
Author(s): UiPath
License: View License Information
Date Published: 3/2/2021 12:37:38 PM
Project URL: View Project Information
Tags: UiPathActivities Activities
Dependencies:
System.ValueTuple [4.5.0,)
Google.Apis [1.43.0,)
Microsoft.Exchange.WebServices [2.2.0,)
Microsoft.Identity.Client [4.14.0,)

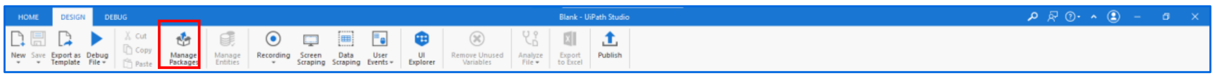
Save Cancel

Step 3: Select the version number from the drop-down menu. Choose the latest version if you are not sure which version to pick. After selecting the version number, click Install.

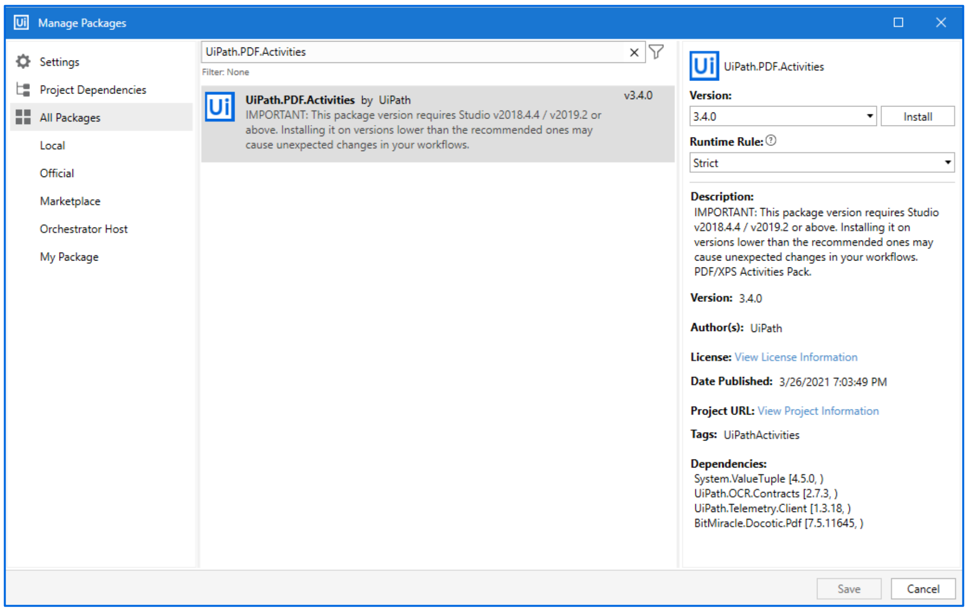
Step 4: Save the packages by clicking on the "Save" button.

2.3.3 PDF Activity Package installation

Step 1: Click on the Manage Packages button from the Design ribbon.



Step 2: Search for **UiPath.PDF.Activities** in the Manage Packages window


Manage Packages

Settings
Project Dependencies
All Packages
Local
Official
Marketplace
Orchestrator Host
My Package

UiPath.PDF.Activities
Filter: None

UiPath.PDF.Activities by UiPath v3.4.0
IMPORTANT: This package version requires Studio v2018.4.4 / v2019.2 or above. Installing it on versions lower than the recommended ones may cause unexpected changes in your workflows.

UiPath.PDF.Activities

Version: 3.4.0 Install

Runtime Rule: Strict

Description:
IMPORTANT: This package version requires Studio v2018.4.4 / v2019.2 or above. Installing it on versions lower than the recommended ones may cause unexpected changes in your workflows.
PDF/XPS Activities Pack.

Version: 3.4.0

Author(s): UiPath

License: View License Information

Date Published: 3/26/2021 7:03:49 PM

Project URL: View Project Information

Tags: UiPathActivities

Dependencies:
System.ValueTuple [4.5.0,)
UiPath.OCR.Contracts [2.7.3,)
UiPath.Telemetry.Client [1.3.18,)
BitMiracle.Docotic.Pdf [7.5.11645,)

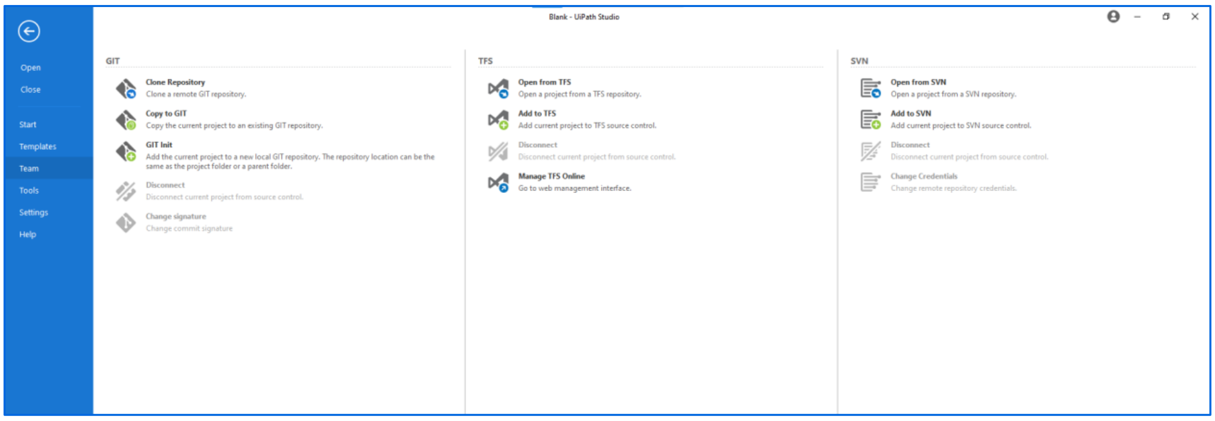
Save Cancel

Step 3: Select the version number from the drop-down menu. Choose the latest version if you are not sure which version to pick. After selecting the version number, click Install.

Step 4: Save the packages by clicking on the "Save" button.

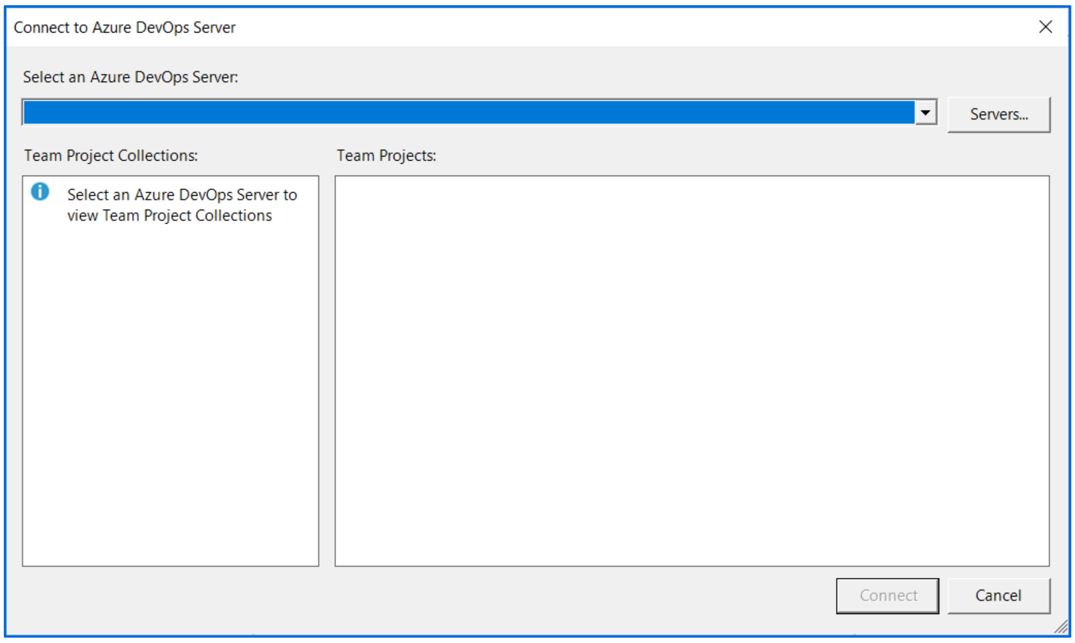
2.4 Version Control using TFS

In UiPath Studio, you can manage and control versions using three options. They are GIT, TFS, and SVN. Select the 'Settings' tab in the Backstage view, then select 'Team' option, and ensure that the corresponding 'Source control plugins' for TFS are enabled.


Blank - UiPath Studio
Open
Close
Start
Templates
Team
Tools
Settings
Help
GIT
Clone Repository
Clone a remote GIT repository.
Copy to GIT
Copy the current project to an existing GIT repository.
GIT Init
Add the current project to a new local GIT repository. The repository location can be the same as the project folder or a parent folder.
Disconnect
Disconnect current project from source control.
Change signature
Change commit signature
TFS
Open from TFS
Open a project from a TFS repository.
Add to TFS
Add current project to TFS source control.
Disconnect
Disconnect current project from source control.
Manage TFS Online
Go to web management interface.
SVN
Open from SVN
Open a project from a SVN repository.
Add to SVN
Add current project to SVN source control.
Disconnect
Disconnect current project from source control.
Change Credentials
Change remote repository credentials.

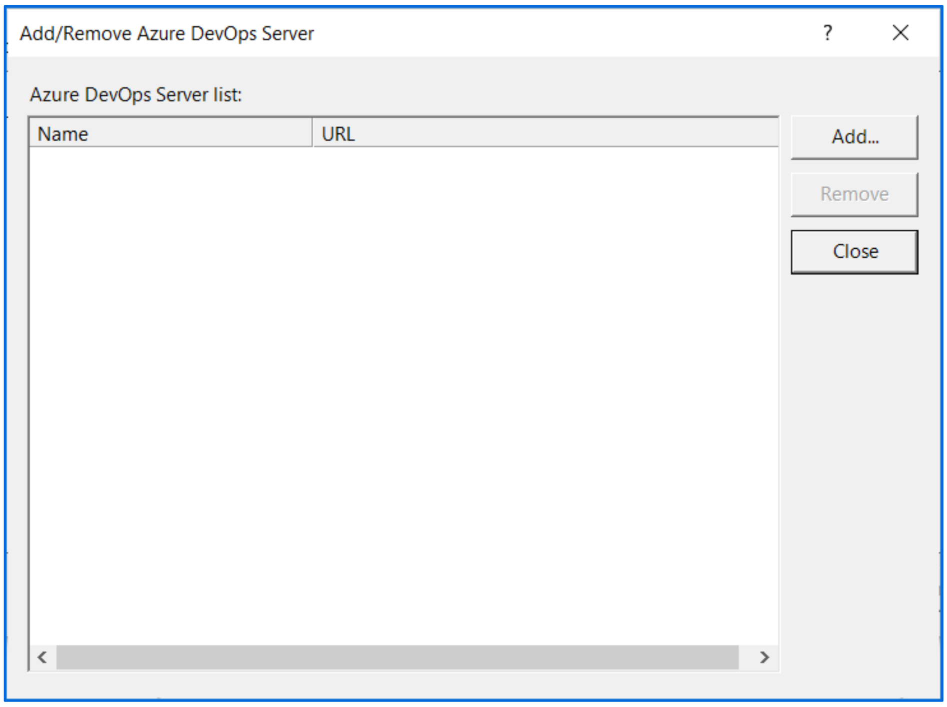
You can set up TFS in UiPath Studio by following the steps mentioned below:

Step 1: In the **Team** tab, click **Open from TFS** or **Add to TFS**. The **Connect to Azure DevOps Server** window is displayed.

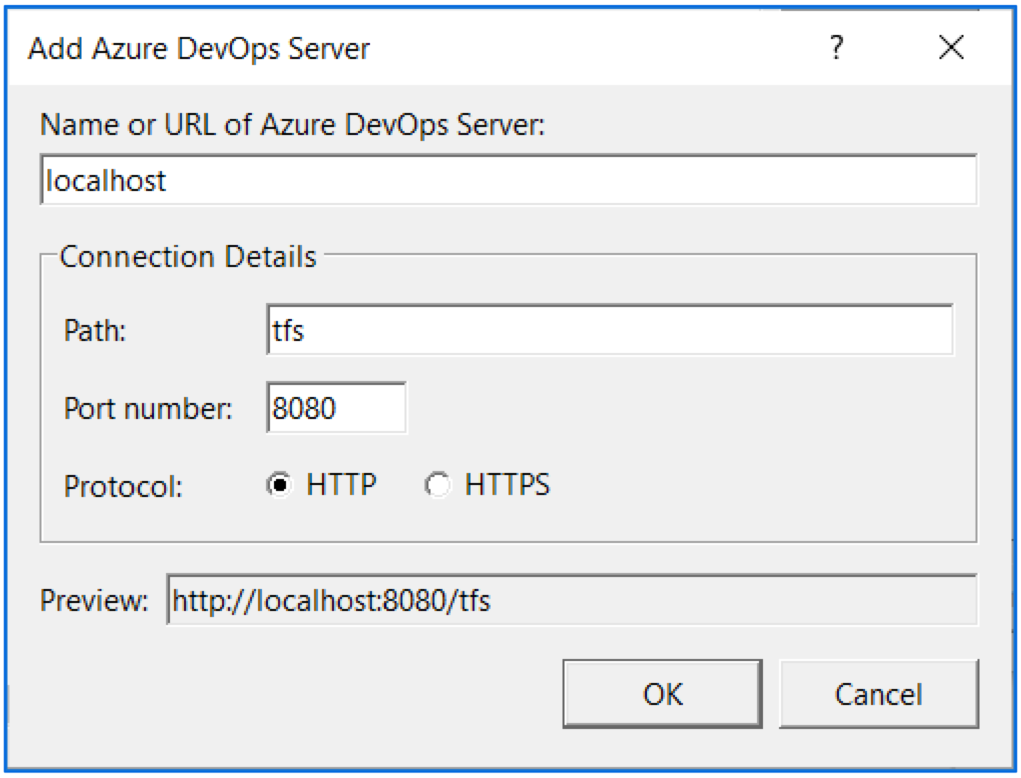

Connect to Azure DevOps Server
Select an Azure DevOps Server:
Servers...
Team Project Collections:
Team Projects:
Connect
Cancel

Select an Azure DevOps Server to view Team Project Collections

Step 2: Click **Servers**. The **Add/Remove Azure DevOps Server** window is displayed.



Step 3: Click the **Add** button. The **Add Azure DevOps Server** window is displayed.



Add Azure DevOps Server

Name or URL of Azure DevOps Server: localhost

Connection Details

Path: tfs

Port number: 8080

Protocol: [✓ HTTP] HTTPS

Preview: http://localhost:8080/tfs

Step 4: Fill in the details of your **TFS** repository and click **OK**. Your team's collections and projects are available in the **Connect to Azure DevOps Server** window.

The **Manage TFS Online** button connects you to the web management interface. If the TFS server is online (e.g. https://<account>.visualstudio.com), UiPath Studio requests authentication with a Microsoft account.

2.5 Hello World

2.5.1 Objective

Build a workflow that prints "Hello World" in a message box.

2.5.2 Process Overview

- START
- Open UiPath Studio
- Add a Sequence activity
- Add a Message Box activity
- Enter the text "Hello World"
- STOP

2.5.3 Step by Step Process

Step 1: Open UiPath Studio.

Step 2: Create a new process and name it as "Hello World"

Step 3: Drag a Sequence activity from the Activities panel and drop it in the Designer panel.

Step 4: Name the Sequence activity as "Sequence – Hello World"

Step 5: Right click on the Sequence activity container and select *Annotations* from the context menu.

Step 6: Enter the annotation 'This code will create a workflow to print "Hello World" in a message box.'

Step 7: Insert a Comment activity from the Activities panel within the Sequence activity.

Step 8: Enter the comment "'Message box' activity displays a message box with the specified text."

Step 9: Insert a Message Box activity below the Comment activity and name it as "Message Box – Hello World". Add the annotation "This activity displays data in a message box."

Step 10: In the text box of the Message Box activity, enter the text "Hello World".

Step 11: Save and run the workflow.

Lesson 3 – Variables and Arguments

3.1 Variable Swapping

3.1.1 Objective

Build a workflow that swaps two numbers using a third variable.

- Ask the user to input two numeric values and store them in two variables.
- Swap values of both the variables using a third variable.
- Display initial and swapped values of both the variables in the Output panel.

3.1.2 Process Overview

- START
- Use an Input Dialog activity to receive two numeric values from the user.
- Store the received values in two integer variables called **First_Input_Value**, and **Second_Input_Value**
- Declare a third integer variable called **Swapping_Support_Variable**
- Use Assign activity to assign the value of **First_Input_Value** to **Swapping_Support_Variable**
- Use second Assign activity to assign the value of **First_Input_Value** to **Second_Input_Value**
- Use third Assign activity to assign the value of **Second_Input_Value** to **Swapping_Support_Variable**
- Use a Write Line activity to display initial and final values of **First_Input_Value** and **Second_Input_Value** in the Output panel.
- STOP

3.1.3 Step by Step Process

Step 1: Open UiPath Studio.

Step 2: Create a process and name it as “Variable Swapping”

Step 3: Drag a Sequence activity from the Activities panel and drop in the Designer panel.

Step 4: Name the Sequence activity as "Sequence – 'This code is for swapping two numbers using a third variable'"

Step 5: Insert a Comment activity from the Activities panel within the Sequence activity.

Step 6: Add comment "Taking input of two numbers from the user and swap them by using a third variable."

Step 7: Drag another Sequence activity from the Activities panel and insert it below the Comment activity.

Step 8: Name the Sequence activity as "Sequence – 'For prompting the user to give the input'".

Step 9: Right-click on the Sequence activity container and select *Annotations* from the context menu.

Step 10: Enter an annotation "This code is for swapping two numbers by using a third variable."

Step 11: Insert an Input Dialog activity within the second Sequence activity and name it as "Input Dialog – 'First Variable by User'".

Step 12: Right-click on the Input Dialog activity container and select Annotations from the context menu. Add an annotation : "Taking User input and storing the value in "First_Input"".

Step 13: In the Input Dialog activity, enter values as shown below:

Title	Label
"First Value"	"Please enter the first numeric value: "

Step 14: In the Variables panel, create a variable for the above Input Dialog activity as shown below:

Name	Variable type	Scope	Default
First_Input_Value	Double	Sequence – 'This code is for swapping two numbers by using a third variable'	

Step 15: Go to the Properties panel of the Input Dialog activity and insert **First_Input_Value** in its Output property.

Step 16: Insert a second Input Dialog activity below the previous Input Dialog activity, and name it as "Input Dialog – 'Second variable by User'".

Step 17: Right-click on the Input Dialog activity container and select *Annotations* from the context menu. Add an annotation : "Taking User input and storing the value in "Second_Input_Value".

Step 18: In the second Input Dialog activity, enter values as shown below:

Title	Label
"Second Value"	"Please enter the second numeric value: "

Step 19: In the Variables panel, create a variable for the second Input Dialog activity as shown below:

Name	Variable type	Scope	Default
Second_Input_Value	Double	Sequence – 'This code is for swapping two numbers by using a third variable'	

Step 20: Go to the Properties panel of the Input Dialog activity and insert the variable **Second_Input_Value** in its Output property.

Step 21: Insert a Write Line activity from the Activities panel after the second Sequence activity, and name it as "Write Line – 'Value entered before swapping'".

Step 22: Right-click on the Write Line activity container and select *Annotations* from the context menu. Add an annotation : "Enter the text to get the result in the Output Panel".

Step 23: In the text box of the Write Line activity, enter the expression: **"First Value is: " + First_Input_Value.ToString + Environment.NewLine + "Second Value is: " + Second_Input_Value.ToString**

Step 24: Insert another Sequence activity from the Activities panel below the Write Line activity, name it as "Sequence – 'Swapping of numbers'" and annotate it as "This block of code will swap the values of the numbers entered".

Step 25: In the Variables panel, create a new variable as shown below:

Name	Variable type	Scope	Default
Swapping_Support_Variable	Double	Sequence – 'This code is for swapping two numbers by using a third variable'	

Step 26: Insert an Assign activity in the third Sequence activity, name it as "Assign – 'Move the First_Input_Value to Swapping_Support_Variable'" and enter the annotation : "Swap Swapping_Support with First_Input_Value".

Step 27: In the Assign activity, enter values as shown below:

To	Value
Swapping_Support_Variable	First_Input_Value

Step 28: Insert a second Assign activity below the previous Assign activity, name it as "Assign – 'Move the Second_Input_Value to First_Input_Value'" and Enter the annotation "Swap First_Input_Value with Second_Input_Value".

Step 29: In the second Assign activity, enter values as shown below:

To	Value
First_Input_Value	Second_Input_Value

Step 30: Insert a third Assign activity below the second Assign activity, name it as "Assign – 'To swap Swapping_Support_Variable with Second_Input_Value'" and enter annotation: "Swap Second_Input_Value with Swapping_Support".

Step 31: In the third Assign activity, enter values as shown below:

To	Value
Second_Input_Value	Swapping_Support_Variable

Step 32: Insert a Write Line activity below the third Sequence activity, name it as "Write Line – 'Swapped Result'" and enter annotation: "Enter the text to get the result in Output Panel".

Step 33: In the text box of the Write Line activity, enter the expression: **"First Value after swapping is: " + First_Input_Value.ToString + Environment.NewLine + "Second Value after swapping is: " + Second_Input_Value.ToString"**

Step 34: Save and run the workflow.

Lesson 4 – Selectors

4.1 Input Methods

4.1.1 Objective

Build a workflow that uses different Input Methods to input data in a Notepad.

- Open a Notepad file and type "Automation makes life easier".
- Minimize the Notepad file using the 'SimulateClick' method.
- Type "Welcome to the new world of Automation" using the 'SendWindowMessages' method.
- Change the font type to Times New Roman, the font style to Italic, and increase the font size by 5.
- Close the Font window by clicking Enter.

4.1.2 Process Overview

- START
- Use Open Application activity to indicate a Notepad file.
- Use Type Into activity to enter "Automation makes life easier".
- Minimize the Notepad window using the 'Simulate Click' method in the Click activity
- Use Type Into activity to enter "Welcome to the new world of automation" using the 'Send Window Messages' method.
- Use Send Hotkey activity to send "Ctrl + A".
- Use Click, Attach Window, and Type Into activities to change the font type to Times New Roman, font style to Italic and increase the font size by 5.
- Use Send Hotkey activity to close the Font window of the Notepad window.
- STOP

4.1.3 Step by Step Process

Step 1: Open a new Notepad file.

Step 2: Open UiPath Studio.

Step 3: Create a new process and name it as “Input Methods”.

Step 4: Drag and Drop a Sequence activity from the Activities panel into the Designer panel.

Step 5: Name the Sequence activity as “Sequence – ‘This is the code to test the input methods in UiPath’”

Step 6: Right-click on the Sequence activity container and select Annotations from the context menu. Add an annotation "This block of code demonstrates a workflow that uses different Input Methods to input data in a Notepad."

Step 7: Now, manually open a Notepad window in the background.

Step 8: In UiPath Studio, Drag and Drop an Open Application activity within the Sequence activity, rename it to “Open Application - Open Notepad” and Enter annotation: “Open application is used to open the blank notepad file.”.

Step 9: Click on the “Indicate element on screen” link and select the Notepad window.

Step 10: Click the hamburger button and select Edit Selector. In the bottom panel of the Selector Editor, rename the title of the notepad to ‘* - Notepad’. Click OK to save the changes.

Step 11: In the Do section of the Open Application activity, drag and drop a Type Into activity from the Activities panel, and rename it to “Type First Text” and Enter annotation: “Types the first text in notepad file.”.

Step 12: Click on the “Indicate element on screen” link of the Type Into activity and select the editor area of the Notepad.

Step 13: In the text box of the Type Into activity, enter the text “Automation makes life easier”.

Step 14: Drag and drop a Click activity from the Activities panel below the Type Into activity, and rename it to “Click Minimize - Simulate Click”.

Step 15: Right-click on the Click activity container and select *Annotations* from the context menu.

Step 16: Add an annotation “Minimizes the notepad window using SimulateClick.”

Step 17: Click on the “Indicate element on screen” of the Click activity, and select the Minimize button of the Notepad.

Step 18: In the Properties panel of the Click activity, check the box of the SimulateClick property.

Step 19: Drag and Drop another Type Into activity below the Click activity, rename it to "Type Second Text- Send window Messages" and Enter annotation: "Types the second text in Notepad file."

Step 20: Click on the "Indicate element on screen" link of the Type Into activity and select the editor area of the Notepad file.

Step 21: In the Properties panel of the second Type Into activity, check the box of SendWindowMessages property.

Step 22: In the text area of the second Type Into activity, enter the text "Welcome to the new world of automation."

Step 23: Insert a Send Hotkey activity below the second Type Into activity, rename it to "Send Hotkey - Select all text" and enter annotation: "Ctrl+A selects all the text in the notepad file."

Step 24: Click on the "Indicate element on screen" link and select the Notepad editor area.

Step 25: Check the box below **Ctrl** option in the Send Hotkey activity. Choose "A" from the dropdown menu below *Key* option.

Step 26: Insert a Click activity from the Activities panel, rename it as "Click Format Button." and enter annotation: "Click Format button from the menu section of the notepad file."

Step 27: Click on the "Indicate element on screen" link, select the *Format* button.

Step 28: Insert an Attach Window activity below the Click activity and rename it to "Attach Window - Format Menu Window". Right-click on the Attach Window activity and select *Annotations* from the context menu.

Step 29: Add an annotation "Performs action on the format menu window"

Step 30: Click on the "Indicate Element on screen" link and Press F2 to add a delay. Now manually open the Format menu. After the timeout, select the *Format* menu from the Notepad window.

Step 31: Inside the Do container of the Attach Window activity, insert a Click activity and name it as "Click Font". Enter the annotation "Click Font button from the menu section of Format."

Step 32: Click on Indicate Element on Screen link to select the *Font* button from the Format menu of Notepad window. The Font button is available in the drop-down. To indicate the Font button, press F2 to add a delay. It will help to first select the format button manually and then indicate the Font button.

Step 33: In UiPath Studio, insert another Attach Window activity after the previous Attach Window activity, and rename it to “Attach Window - Font window”. Add an annotation “Performs Action on Font window.”

Step 34: Select the *Font* window by using “Indicate element on screen” link of the Attach Window activity.

Step 35: Insert a Type Into activity within the second Attach Window activity, name it as “Type into - Font type” and enter annotation: “Types "Times New Roman" as Font Type.”

Step 36: Select the input field of the Font names section in the Font window.

Step 37: Type “Times New Roman” in the text field of the Type Into activity.

Step 38: Drag and drop another Type Into activity, name it as “Type into – Font Style”, Enter annotation: “Types "Italic" as Font Style.” and select the input field of Font style section in the Font window.

Step 39: Type “Italic” in the text field of the Type Into activity.

Step 40: Insert a Get Text activity below the Type Into activity, name it as “Get Text – Font Size” and Enter annotation: “Text gets stored in a variable named "FontSize"”.

Step 41: Select the input field of the Font Size section in the Font window.

Step 42: In the Properties section of the Get Text activity, save the result into a variable named as **FontSize**.

Step 43: Insert a Type Into activity, name it as “Type Into – New Font Size”, Enter annotation: “Incrementing Font size with '5'.” and select the input field of Font Size from the Font window.

Step 44: In the text area of the Type Into activity, enter the expression: (**cint(FontSize) + 5).ToString**.

Step 45: Insert a Send hotkey activity below the Type Into activity, name it as “Send Hotkey - Enter” and enter annotation: “Close the Font Window of Notepad Window”.

Step 46: Select *enter* from the dropdown menu of Key option.

Step 47: Save and run the workflow.

4.2 Starting Browser

4.2.1 Objective

Build a workflow that opens a browser and then opens UiPath's website.

- Open a browser.
- Open the URL – www.uipath.com.
- Display "Success" in a message box.

4.2.2 Process Overview

- START
- Use an Open Browser activity and enter the website URL – "www.uipath.com".
- Use a Message Box activity and enter text "Success".
- STOP

4.2.3 Step by Step Process

Step 1: Open UiPath Studio.

Step 2: Create a new process and name it as "Starting Browser".

Step 3: Insert a Sequence activity from the Activities panel into the Designer panel.

Step 4: Name the Sequence activity as "Sequence – This code is to understand about Open Browser activity".

Step 5: Right-click on the Sequence activity container and select Annotations from the context menu. Add an annotation "This block of code demonstrates a workflow that opens a browser and then opens UiPath's website."

Step 6: Insert an Open Browser activity from the Activities panel within the Sequence activity, name it as "Open Browser – Opens www.uipath.com" and enter the annotation "Open browser and redirects to the URL "www.uipath.com"".

Step 7: In the text box of the Open Browser activity, enter "www.uipath.com" (in double quotes).

Step 8: Insert a Message Box activity within the Do container of the Open Browser activity, name it as "Message Box – Success" and enter annotation: "Prints "Success"".

Step 9: In the text box of the Message Box activity, enter the text "Success"

Step 10: Save and run the workflow.

4.3 Web Recording

4.3.1 Objective

Build a workflow using Web Recorder in UiPath Studio to Sign in to UiPath's website.

- Ask the user's email address and password.
- Open the sign in page of UiPath's Website.
- Sign in to the website using the user's credentials.

4.3.2 Process Overview

- START
- Use two Input Dialog activities and take the email address and password from the user.
- Store received input in two variables.
- Use an Open Browser activity and open URL – "www.uipath.com".
- Use the Web Recorder in UiPath Studio to:
  - Click the *Try UiPath Free* button.
  - Click the *Sign in* the link on the next page.
  - Type in the user's email address and password.
  - Click on the *Sign in* button.
- STOP

4.3.3 Step by Step Process

Step 1: Open UiPath Studio.

Step 2: Create a new process and name it as "Web Recording".

Step 3: Drag a Sequence activity from the Activities panel and drop in the Designer panel.

Step 4: Name the Sequence activity as "Sequence – UiPath Sign in using Web Recorder".

Step 5: Right-click on the Sequence activity container and select *Annotations* from the context menu.

Step 6: Enter the annotation "This block of code takes the credentials from the user and signs in to UiPath's website using Web Recorder."

Step 7: Insert an Input Dialog activity within the Sequence activity, name it as "Input Dialog – Email Address" and add an annotation "Takes Email Address from User."

Step 8: In the Input Dialog activity, enter values as shown below:

Title	Label
"Email Address"	"Enter your Email Address"

Step 9: In the Variables panel, create a variable for the above Input Dialog activity as shown below:

Name	Variable type	Scope	Default
EmailAddress	String	Sequence – 'UiPath Sign in using Web Recorder'	

Step 10: Go to the Properties panel of the Input Dialog activity and insert **EmailAddress** in its Output property.

Step 11: Insert another Input Dialog activity below the previous Input Dialog activity, name it as "Input Dialog – Password" and enter annotation: "Takes Password from User".

Step 12: In the Input Dialog activity, enter values as shown below:

Title	Label
"Password"	"Enter your Password"

Step 13: In the Variables panel, create a variable for the above Input Dialog activity as shown below:

Name	Variable type	Scope	Default
Password	String	Sequence – 'UiPath Sign in using Web Recorder'	

Step 14: Go to the Properties panel of the Input Dialog activity and insert **Password** in its Output property.

Step 15: Insert an Open Browser activity below the Input Dialog activity, name it as “Open Browser - Opens "www.uipath.com" webpage” and add an annotation: “Open UiPath webpage.”

Step 16: In the text area of the Open Browser activity, and enter URL “www.uipath.com”.

Step 17: Run the program at this stage to ensure that the browser opens the URL. Keep the website opened to follow further steps.

Step 18: In the Design ribbon of UiPath Studio, click on the Recording button, and select *Web* from the dropdown menu.

Step 19: Select the Click option from the Web Recording toolbar, and indicate the *Try UiPath Free* button on UiPath’s Website. It opens another webpage.

Step 20: Again select the Click option from the Web Recording toolbar, and indicate the *Sign in* link. It opens another webpage containing the Sign in form.

Step 21: Again select the Click option from the Web Recording toolbar, and indicate the *Sign in with Email* link.

Step 22: Select Type option from the Web Recording toolbar, and indicate the input area of the Email Address of the sign in form. Enter the variable **EmailAddress** into it.

Step 23: Again select the Type option from the Web Recording toolbar, and indicate the input area of the Password of the sign in form. Enter the variable **Password** into it.

Step 24: Select the Click option from the Web Recording toolbar, and indicate the *Sign in* button of the sign in form.

Step 25: Click the Save and Exit option from the Web Recording toolbar.

Step 26: Drag the Web container from outside of the Open Browser activity container, and drop it into the Do container of the Open Browser activity.

Step 27: Save and run the workflow.

4.4 Anchors

4.4.1 Objective

Build a workflow that fills the form on RPAChallenge.com website with organized data from an excel file.

- Download the practice excel file from RPAChallenge.com.
- Take CompanyName, RoleInCompany, Address, Email, FirstName, LastName, PhoneNumber from the excel file.
- Submit the data in the RPAChallenge.com form.
- Use anchor-based selectors for this exercise.

4.4.2 Prerequisites

✓ Challenge.xlsx Excel file from RPAChallenge.com.
✓ UiPath.Excel.Activities package.

4.4.3 Process Overview

- START
- Use a Read Range activity within an Excel Application Scope activity to read data from an Excel file challenge.xlsx and store it in a data table.
- Use an Open Browser activity to open the URL – "www.rpachallenge.com".
- Use a Click activity to click on the Start button on the website.
- Use a For Each Row activity to loop through rows in the data table.
- Use Anchor Base activities within the For Each Row activity to locate anchors for each input area of the RPAChallenge.com submission form.
- Use Type Into activities within Anchor Base activities to input data from the data table.
- Use string manipulation methods called Substring and IndexOf to manipulate the names of the users.
- Use a Click activity to submit the form.
- STOP

4.4.4 Step by Step Process

Step 1: Download the 'Challenge.xlsx' excel file from www.rpachallenge.com and place it in a project folder.

Step 2: Open the URL www.rpachallenge.com in a browser.

Step 3: Open UiPath Studio.

Step 4: Create a new process and name it as "Anchors"

Step 5: Drag a Sequence activity from the Activities panel and drop it in the Designer panel.

Step 6: Name the Sequence activity as Sequence – 'RPA Challenge'

Step 7: Right-click on the Sequence activity container and select *Annotations* from the context menu.

Step 8: Add an annotation "Creating a workflow that fills the form on 'rpachallenge.com' with organized data that will be taken from an Excel file, The first and last name is obtained from the full name column of the excel file."

Step 9: Insert an Excel Application Scope activity in the Sequence activity.

Step 10: Right-click on the Excel Application Scope activity container and select *Annotations* from the context menu.

Step 11: Add an annotation "Reading an Excel File that contains the following data: CompanyName, RoleInCompany, Address, Email, FirstName, LastName, PhoneNumber."

Step 12: In the Variables panel, create a variable as shown below:

Name	Variable type	Scope	Default
dt_DataTable	DataTable	Sequence – 'RPA Challenge	

Step 13: Drag a Read Range activity from the activities panel under the "Excel" category and drop it into the Excel Application Scope activity. Name the activity as "Read Range - Challenge.xlsx" and Add an annotation: "Reads "Sheet1" of challenge.xlsx".

Step 14: In the Properties panel of the Read Range activity, enter **dt_DataTable** in the Output property.

Step 15: Click on the File path button of the Excel Application Scope and select the excel file already stored into the project folder.

Step 16: Insert an Open Browser activity and enter the URL - “www.rpachallenge.com”

Step 17: In the Do section of the Open Browser activity, insert a Click activity. Name it as “Click 'START BUTTON'”, and add an annotation “Clicks the START button”.

Step 18: Click on the “Indicate element on screen” link, and select the Start button of the RPA challenge website.

Step 19: Insert a For Each Row activity after the Click activity

Step 20: Right-click on the For Each Row activity select *Annotations* from the context menu.

Step 21: Add an Annotation “It will iterate through each row of the DataTable.”

Step 22: Insert “Row” in the first section and “dt_DataTable” as the *Vb Expression* in the second section of the For Each Row activity. Name the activity as “For Each Row of dt_DataTable variable”.

Step 23: In the Body section of For Each Row activity, insert seven Anchor Base activities one after another, and name them in this order:

- “Anchor Base – Address”
- “Anchor Base – Role in Company”
- “Anchor Base – Phone Number”
- “Anchor Base – Last Name”
- “Anchor Base – Company Name”
- “Anchor Base – First Name”
- “Anchor Base – Email”.

Step 24: In the **Anchor** sections of each of Anchor Base activities, insert Find Element activities, name and Annotate them as shown below:

Container Name	**Rename Find Element to:**	**Annotate Find Element to:**
Anchor Base – Address	Find Element – Address	The Find Element activity identifies the Address field.
Anchor Base – Role in Company	Find Element – Role in Company	The Find Element activity identifies the Role in Company field.
Anchor Base – Phone Number	Find Element – Phone Number	The Find Element activity identifies the Phone Number field.
Anchor Base – Last Name	Find Element – Last Name	The Find Element activity identifies the Last Name field.

Anchor Base – Company Name	Find Element – Company Name	The Find Element activity identifies the Company Name field.
Anchor Base – First Name	Find Element – First Name	The Find Element activity identifies the First Name field.
Anchor Base – Email	Find Element – Email	The Find Element activity identifies the Email field.

Step 25: Insert a Type Into activity in the **Action Activity** section of each of the Anchor Base activities, and name and annotate them as shown below:

Container Name	**Rename Find Element to:**	**Annotate Type Into to:**
Anchor Base – Address	Type Into – Address	The Type Into activity writes in the Address field
Anchor Base – Role in Company	Type Into – Role in Company	The Type Into activity writes in the Role in Company field
Anchor Base – Phone Number	Type Into – Phone Number	The Type Into activity writes in the Phone Number field
Anchor Base – Last Name	Type Into – Last Name	The Type Into activity writes in the Last Name field
Anchor Base – Company Name	Type Into – Company Name	The Type Into activity writes in the Company Name field
Anchor Base – First Name	Type Into – First Name	The Type Into activity writes in the First Name field
Anchor Base – Email	Type Into – Email	The Type Into activity writes in the Email field

Step 26: Click the "Indicate element on screen" link of each of the Find Element activities and indicate the respective labels from the RPAChallenge.com form field as shown below:

Find Element Name	**Indicate the form label to:**
Find Element – Address	Address
Find Element – Role in Company	Role in Company
Find Element – Phone Number	Phone Number
Find Element – Last Name	Last Name
Find Element – Company Name	Company Name

Find Element – First Name	First Name
Find Element – Email	Email

Step 27: Click the "Indicate element on screen" link of each of the Type Into activities and select the input fields of the respective labels as shown below:

Find Element Name	**Indicate the form input area of:**
Type Into – Address	Address
Type Into – Role in Company	Role in Company
Type Into – Phone Number	Phone Number
Type Into – Last Name	Last Name
Type Into – Company Name	Company Name
Type Into – First Name	First Name
Type Into – Email	Email

Step 28: In the text area of each of the Type Into activities, enter respective expressions as shown below:

Find Element Name	**Indicate the form input area of:**
Type Into – Address	row(4).ToString
Type Into – Role in Company	row(3).ToString
Type Into – Phone Number	row(6).ToString
Type Into – Last Name	row(1).ToString
Type Into – Company Name	row(2).ToString
Type Into – First Name	row(0).ToString
Type Into – Email	row(5).ToString

Step 29: Insert a Click activity after "Anchor Base – Email" container, name it as "Click - Submit" and Add an annotation: "Clicks the Submit button to end the challenge".

Step 30: Click the "Click element inside browser" link and select the Submit button of the form.

Step 31: Save and run the workflow.

4.5 Debugging Selectors

4.5.1 Objective

Build a workflow that replaces double spaces with single spaces from a text stored in multiple Notepad files with different names.

- Open a notepad file containing text with double space
- Replace double spaces with single spaces
- Debug the selector to make the workflow work for Notepad files with different names

4.5.2 Prerequisites

✓ Before beginning this practice exercise, create three Notepad files with names "File_123421.txt", "File_43153.txt", and "File_34689.txt" that contain some text with double spaces between few words.

4.5.3 Process Overview

- START
- Use an Attach window activity and indicate one Notepad file (File_123421.txt)
- Use the wildcard (*) in the Notepad file name using the Selector Editor to make the workflow work for a Notepad with a different name.
- Use Click activities and indicate the Replace option within the Edit option from the Notepad file's toolbar.
- Use Type Into activities to enter double space in the *Find what* text box and single space in the *Replace with* text box of the Replace popup window.
- Use Click activities to click the Replace All button and to close the Replace pop up window.
- Run the program to replace the double spaces of "File_123421.txt" file.
- STOP

4.5.4 Step by Step Process

Step 1: Open "File_123421.txt" file.

Step 2: Open UiPath Studio.

Step 3: Create a new process and name it as "Debugging Selectors".

Step 4: Drag a Sequence activity from the Activities panel and drop it in the Designer panel.

Step 5: Name the Sequence activity as “Sequence - Debugging Selectors”.

Step 6: Right-click on the Sequence activity container and select *Annotations* from the context menu.

Step 7: Add an annotation “This block of code demonstrates a workflow that replaces double spaces with single spaces from a text stored in multiple Notepad files with different names.”

Step 8: Now, Manually open the “File_123421.txt” Notepad file in the background.

Step 9: Drag and Drop an Attach Window activity within the Sequence activity, name it as “Attach Window- Notepad” and Add an annotation “Attach Notepad Window”.

Step 10: Click on the “Indicate element on screen” link, and select the notepad editor window of “File_123421.txt”

Step 11: Insert a Click activity in the Do section of the Attach Window activity and name it as “Click - ‘Edit’”. Add an annotation “Click Edit option form the notepad menu”.

Step 12: Click on the “Indicate element on screen” link of the Click activity, select the *Edit* option from the toolbar of the Notepad File.

Step 13: Insert another Click activity in the Do section of the Attach window activity, name it as “Click - ‘menu item - Replace’” and add an annotation “Click Replace option from the Edit Menu”.

Step 14: Click on the “Indicate element on screen” link. The *Replace* button is available in the drop-down menu of the *Edit* option. To indicate the *Replace* button, press F2 to add a delay. It will help to first select the *Edit* button manually and then indicate the *Replace* button.

Step 15: Insert a Type Into activity and name it as “Type into - ‘Find What’”. Enter annotation: “Types in 'Find What' section of Replace window”.

Step 16: Click on the “Indicate element on screen” link and select the text area of the *Find what* section.

Step 17: In the text area of the Type Into activity, enter double space within double quotes – “ ”.

Step 18: Insert a Send Hotkey activity below the previous Click activity. Click on the "Indicate element inside window" link and select the notepad window. Check the box below *Ctrl* and select 'A' from the drop-down menu below *Key*.

Step 19: Insert another Type Into activity, name it as "Type into - 'Replace With'" and Enter the annotation "Types in 'Replace with' section of the Replace window".

Step 20: Click on the "Indicate element on screen" link and select the text area of *Replace with* section.

Step 21: In the text area of the Type Into activity, enter single space within double quotes – " ".

Step 22: Insert a Click activity under the Type Into activity, name it as "Click - 'Replace All'" and Add an annotation "Click Replace All button."

Step 23: Click on the "Indicate element on screen" link and select Replace All button from the Replace pop-up window.

Step 24: Insert another Click activity and name it as "Click - 'Close'". Add an annotation "Click 'Close" button."

Step 25: Click the "Indicate element on screen" link and select the Close button of the Replace pop-up window.

Step 26: Save and run the workflow to remove all double spaces in the "File_123421.txt"

Step 27: Close the "File_123421.txt" and open the "File_43153.txt".

Step 28: Run the workflow. It will give an error called 'Selector Not Found.' Debug this error using wildcard.

Step 29: In UiPath Studio, click on the hamburger icon of the Attach Window activity, and select *Edit Selector* from the context menu.

Step 30: In the Edit Selector panel of the Selector Editor window, rename the title of the Notepad file from "File_123421.txt" to "File_*.txt" in the Edit Selector section.

Step 31: Save and run the workflow for all three text files – "File_123421.txt", "File_43153.txt" and "File_34689.txt".

Lesson 5 – Control Flow

5.1 If Statement

5.1.1 Objective

Build a workflow using an If statement, which asks a user, whether the user will get the second Marshmallow or not.

- Ask the user, "Do you want to eat your first Marshmallow now or after 5 minutes?"
- If the user answers "Now", respond with "Oops! You will not get the second Marshmallow."
- If the user answers "After 5 minutes", respond with "Congrats! You will also get the second Marshmallow."
- If the answer is other than "Now" or "After 5 minutes", respond with "Invalid Input".

5.1.2 Process Overview

- START
- Use an Input Dialog activity to ask the user "Do you want to eat your first Marshmallow now or after 5 minutes?"
- Store user response in a string variable.
- Use an If activity to check the user response
    - If the answer is "Now", use a Message Box activity to display "Oops! You will not get the second Marshmallow."
    - If the answer is "After 5 minutes", use a Message Box activity to display "Congrats! You will also get the second Marshmallow."
    - If the answer is other than "Now" or "After 5 minutes", use a Message Box activity to display "Invalid Input".
- STOP

5.1.3 Step by Step Process

Step 1: Open UiPath Studio.

Step 2: Create a new process and name it as "If Statement".

Step 3: Drag a Sequence activity from the Activities panel and drop it in the Designer panel.

Step 4: Name the Sequence activity as “Sequence – ‘Marshmallow Game’”.

Step 5: Right-click on the Sequence activity container and select *Annotations* from the context menu.

Step 6: Enter the annotation “This code is to ask the user whether he wants a second Marshmallow.”

Step 7: Insert an Input Dialog activity within the Sequence activity and name it as “Input Dialog – ‘Question’”. Enter the annotation “Question to User”.

Step 8: In the Input Dialog activity, enter values as shown below:

Title	Label
“Question”	“Do you want to eat your first Marshmallow? Choose among the following options: ” + Environment.NewLine + “1. Now” + Environment.NewLine + “2. After 5 minutes”

Step 9: In the Variables panel, create a variable for the above Input Dialog activity as shown below:

Name	Variable type	Scope	Default
UserInput	String	Sequence – ‘Marshmallow Game’	

Step 10: Go to the Properties panel of the Input Dialog activity and insert **UserInput** in its Output property.

Step 11: Insert an If activity below the Input Dialog activity and name it as “If - To check if the user input is ‘Now’”. Enter annotation: “This activity judges the User Input whether it is "Now", "After 5 minutes" or "Invalid"”.

Step 12: In the condition input area of If activity, enter the expression: **UserInput = “Now”**.

Step 13: Insert a Message Box activity in the **Then** section of the If activity and name it as “Message Box - Failed”. Enter annotation: “Prints Fail message”.

Step 14: In the Message Box activity, enter the text “Oops! You will not get the second Marshmallow.”

Step 15: Insert a second If activity, name it as “If - To check User input is 'After 5 minutes'”, add an annotation “Check whether the input contains ‘After 5 minutes’ or Invalid input” within the **Else** section of the first If activity.

Step 16: In the condition input area of second If activity, enter the expression: **UserInput = “After 5 minutes”**.

Step 17: Insert a Message Box activity in the **Then** section of the second If activity and name it as “Message Box - Success”. Add an annotation: “Prints Success message”.

Step 18: In the Message Box activity, enter the text “Congrats! You will get the second Marshmallow.”

Step 19: Insert another Message Box activity in the **Else** section of the second If activity and name it as “Message Box – Invalid Input”. Add an annotation: “Prints Invalid Input message”.

Step 20: In the Message Box activity, enter the text “Invalid Input”

Step 21: Save and run the workflow.

5.2 Switch Activity

5.2.1 Objective

Build a workflow using Switch activity that asks users' their eye color and display their personality in a message box.

- Ask the user for their eye color.
- If the user enters "Blue", respond with "You must be very Brave!"
- If the user enters "Green", respond with "You must be Generous!"
- If the user enters "Gray", respond with "You must be very Wise!"
- If the user enters "Black", respond with "You must be very Bold!"

5.2.2 Process Overview

- START
- Use an Input Dialog activity to get the eye color input of the user.
- Use a Switch activity to compare the input with four different cases – Blue, Green, Gray, and Black.
- Use Message Box activities to display the result of each case
  - For "Blue", display "You must be Brave!"
  - For "Green", display "You must be Generous!"
  - For "Gray", display "You must be very Wise!"
  - For "Black", display "You must be very Bold!"
- STOP

5.2.3 Step by Step Process

Step 1: Open UiPath Studio.

Step 2: Create a new process and name it as "Switch Activity"

Step 3: Drag a Sequence activity from the Activities panel and drop it in the Designer panel.

Step 4: Name the Sequence activity as "Sequence – 'Create a Robot that asks user their eye color'"

Step 5: Right-click on the Sequence activity container and select *Annotations* from the context menu.

Step 6: Enter the annotation “This block of code is executed using Switch Activity.”

Step 7: Insert an Input Dialog activity, name it as “Input Dialog – ‘Question’”, Add an annotation “Question to User” and enter the values as shown below:

Title	Label
“Question”	“Enter the color of your eye:”

Step 8: In the Variables panel, create a variable for the above Input Dialog activity as shown below:

Name	Variable type	Scope	Default
EyeColor	String	Sequence – ‘Create a Robot that asks user their eye color’	

Step 9: Go to the Properties panel of the Input Dialog activity and insert **EyeColor** in its Output property.

Step 10: Drag and drop a Switch activity below the Input Dialog activity, name it as “Switch - Eye Color” and Add an annotation “Switch activity compares input with four different cases – Blue, Green, Gray, and Black.”

Step 11: In the Properties panel of the Switch activity ensure that the TypeArgument property is **String**.

Step 12: In the Expression text area of the Switch activity, enter the variable **EyeColor**.

Step 13: In the Default section of the Switch activity, insert a Message Box activity and name it as “Message Box - Default”. Add an annotation: “Prints default message”

Step 14: In the text area of the Message Box activity, enter the text “Can’t recognize that color!”

Step 15: Add a new case in the Switch activity by clicking ‘Add new case’ option available. Enter “Blue” in the text area of ‘Case value’.

Step 16: In the ‘Case Blue’ section, add a Message Box activity, and name it as “Message Box - Case Blue”. Add an annotation “Prints Case Blue message” and in the text area enter “You must be very Brave!”

Step 17: Add a new case and enter “Green” in the text area of the ‘Case value’.

Step 18: In the ‘Case Green’ section, add a Message Box activity, name it as “Message Box - Case Green”. Add an annotation “Prints Case Green message” and in the text area enter “You must be very Generous!”

Step 19: Add a new case and enter “Gray” in the text area of the ‘Case value’.

Step 20: In the ‘Case Gray’ section, add a Message Box activity, name it as “Message Box - Case Gray”. Add an annotation “Prints Case Gray message” and in the text area enter “You must be very Wise!”

Step 21: Add a new case and enter “Black” in the text area of the ‘Case value’.

Step 22: In the ‘Case Black’ section, add a Message Box activity, and name it as “Message Box - Case Black”. Add an annotation “Prints Case Black message”.

Step 23: In the text area, enter “You must be very Bold!”

Step 24: Save and Run the project.

5.3 Do While Loop

5.3.1 Objective

Build a workflow for a 'Guessing Game' with the following conditions:

- Generate a random number and prompt the user to input a number.
- In case of a wrong input, a message is displayed to the user stating, 'Please enter a lesser/greater number'.
- The loop keeps on running until the input number equals to the generated number.

5.3.2 Process Overview

- START
- Use an Input Dialog activity within a Do While activity to get the guessed number from the user.
- For Do While activity, set the condition to check guessed number is not equal to the actual number.
- Use a Message Box activity to display “You Guessed it correct” for the correct match.
- Use an If activity within the Do While loop to check if the guessed number is equal to the actual number.
    - If correct, use a Message Box activity to display “You Guessed it correct” for the correct match.
    - Use another If activity within the Else section to check if the guessed number is greater than the actual number.
        - If correct, use a Message Box activity to display “Please try a smaller number”.
        - If incorrect, use a Message Box activity to display “Please try a greater number”.
- STOP

5.3.3 Step by Step Process

Step 1: Open UiPath Studio.

Step 2: Create a new process and name it as “Do While Loop”.

Step 3: Drag a Sequence activity from the Activities panel and drop it in the Designer panel.

Step 4: Name the Sequence activity as "Sequence – 'Guessing Game".

Step 5: Right-click on the Sequence activity container and select *Annotations* from the context menu.

Step 6: Enter the annotation :

"This block of code demonstrates a workflow using a Do While statement for creating a 'Guessing Game' with the following conditions:

1. Generate a random number and prompt the user to input a number.
2. In case of a wrong input, a message is displayed to the user stating, 'Please enter a lesser/greater number'.
3. The loop keeps on running until the input number equals the generated number."

Step 7: Create variables using Variables panel as shown below:

Name	Variable type	Scope	Default
intRandomNo	Int32	Sequence – Guessing Game	25
intGuessedNo	Int32	Sequence – Guessing Game	

Step 8: Insert a Do While activity within the Sequence activity, name it as "Do While - Guessed Number <> Random Number", add an annotation "The loop iterates until it reaches the given condition".

Step 9: Set its condition to **int intGuessedNo<> intRandomNo**

Step 10: Insert an Input Dialog activity within the Do While activity, name it as "Input Dialog - Guessed Number", add an annotation: "Take Guessed Number as User input" and enter values as shown below:

Title	Label
"Number"	"Guess a number"

Step 11: In the Properties panel of the Input Dialog activity, enter **intGuessedNo** in the Output property.

Step 12: Insert an If activity below the Input Dialog activity, name it as “If - User input equals Random Number”, add an annotation: “This activity checks whether the User input is equal to the Random Number or not” and enter the condition **intGuessedNo= intRandomNo**

Step 13: In the Then section, insert a Message Box activity and name it as “Message Box - Correct Guess”. Add an annotation: “Prints Correct Guess message”.

Step 14: Enter the text “You Guessed it correct”.

Step 15: Insert another If activity, in the **Else** section of the first If activity, and enter condition **intGuessedNo> intRandomNo.** Name it as “If- Guessed number is greater or smaller than Random Number”, add an annotation: “This activity checks whether the user input is greater or smaller than the Random number.”

Step 16: In the Then section, insert a Message Box activity, name it as “Message Box - Try Smaller Number”, add an annotation: “Prints Smaller Number message” and enter the text “Please try a smaller number”.

Step 17: In the Else section, insert a Message Box activity, name it as “Message Box - Try Greater Number”, add an annotation: “Prints Greater Number message” and enter the text “Please try a greater number”.

Step 18: Save and run the workflow.

5.4 While Loop

5.4.1 Objective

Build a workflow using a While loop that tells the user if the input is a prime number or not.

- Ask the user to input a number.
- Check if it is a prime number.
- If the input number is prime, then display "It is a prime number" in a message box.
- If the input number is not prime, then display "It is not a prime number" in a message box.

5.4.2 Process Overview

- START
- Use an Input Dialog activity and ask for any number from the user and store in a variable called **intNumber**.
- Create two more variables **intRandom** and **c** with Variable Type as **Int32** and Default value as **2** and **0** respectively in the variables panel.
- Use a While activity and set the condition to **intRandom<Number**.
- Use an If activity within the While activity and set the condition to **intNumber mod intRandom=0**.
- Use an Assign activity within the **Then** section and increment value of **intCount** by **1**.
- Use an Assign activity after/below the If activity, and increment value of **intRandom** by **1**.
- Use another If activity after/below the While activity and enter condition **intCount>0**.
- Use a Message Box activity within the Then section to display “It is not a prime number”.
- Use a Message Box activity within the Else section to display “It is a prime number”.
- STOP

5.4.3 Step by Step Process

Step 1: Open UiPath Studio.

Step 2: Create a new process and name it as “While Activity”.

Step 3: Drag a Sequence activity from the Activities panel and drop it in the Designer panel.

Step 4: Name the Sequence activity as “Sequence – ‘This is the code to test whether the input is a prime number or not.’”

Step 5: Right-click on the Sequence activity container and select *Annotations* from the context menu.

Step 6: Enter the annotation : “This block of code demonstrates a workflow using While loop that tells the user if the input is a prime number or not.”

Step 7: Insert an Input Dialog activity within the Sequence activity, name it as “Input Dialog – ‘To take the input from the user’” and add an annotation “Take User input as a Number”.

Step 8: In the Input Dialog activity, enter values as shown below:

Title	Label
“Number”	“Enter a number”

Step 9: In the Variables panel, create three variables as shown below:

Name	Variable type	Scope	Default
intNumber	Int32	Sequence – ‘This is the code to test whether the input is a prime number or not.’	
intRandom	Int32	Sequence – ‘This is the code to test whether the input is a prime number or not.’	2
intCount	Int32	Sequence – ‘This is the code to test whether the input is a prime number or not.’	0

Step 10: Go to the Properties panel of the Input Dialog activity and insert **intNumber** in its Output property.

Step 11: Insert a While activity below the Input Dialog activity and name it as “While – ‘To check if the number is a prime number or not’”.

Step 12: Right-click on the While activity container, and select *Annotations* from the context menu.

Step 13: Add an annotation “This block of code will check whether the number is prime. If it is, it will increment the value of ' intCount’.”

Step 14: Inside the While activity, enter the condition as **intRandom < intNumber**

Step 15: In the **Body** section of the While activity drag and drop a Sequence activity.

Step 16: Rename the Sequence activity to “Sequence – ‘Check the number using ‘If’”.

Step 17: Right-click on the Sequence activity container and select *Annotations* from the context menu.

Step 18: Add an annotation “In this sequence using 'If' activity, the 'Number' is divided by ' **intRandom** ' until **intRandom** = **intNumber**.”

Step 19: Insert an If activity inside the Sequence activity.

Step 20: Inside the If activity, enter the condition as **intNumber Mod intRandom = 0**.

Step 21: Inside the **Then** section of the If activity, insert an Assign activity, and enter values as shown below:

To	Value
intCount	intCount + 1

Step 22: Change the Assign activity name to “Assign – ‘Increment the value of intCount’”.

Step 23: Right-click on the Assign activity container and select *Annotations* from the context menu.

Step 24: Add an annotation “Incrementing the value of ‘intCount’ when ‘**intNumber** is found to be a prime number.”

Step 25: Below the If activity, insert another Assign activity and rename it to “Assign-Incrementing the value of ‘**intRandom**’.

Step 26: In the Assign activity, enter the values as shown below:

To	Value
intRandom	intRandom +1

Step 27: Right-click on the Assign activity container, and select *Annotations* from the context menu.

Step 28: Add an annotations "Incrementing the value of ' **intRandom** ' whenever the loop iterates".

Step 29: Below the While activity, insert an If activity and name it as "If – Print the message".

Step 30: Right-click on the If activity container, and select *Annotations* from the context menu.

Step 31: Add an annotation "This block of code will print the message in a message box whether the input is Prime or not."

Step 32: Inside the If activity, enter the condition **intCount >0**.

Step 33: In the **Then** section, insert a Message Box activity and name it as "Message Box - Not a prime number". Add an annotation "Displays that the number is not a prime."

Step 34: Enter the text "It is not a prime number."

Step 35: In the **Else** section, insert another Message Box activity and name it as "Message Box - Is a prime number". Add an annotation "Displays that the number is not a prime."

Step 36: Enter the text "It is a prime number."

Step 37: Save and run the workflow.

5.5 For Each Loop

5.5.1 Objective

Build a workflow to display file names from a folder in the Output panel and also store names in an MS Word file.

- Locate and select a folder containing multiple files.
- List the directory path of all the files in the Output panel.
- Also, store the updated names in an MS Word file and save and close it.

5.5.2 Process Overview

- START
- Use a Select Folder activity to select a folder containing a few files.
- Use an Assign activity to store file names in an array.
- Use an Attach Window activity below the Assign activity and select MS Word window.
- Use a For Each activity to iterate through each file name in the array.
- Use a Write Line activity within the For Each activity to display file names in the Output panel.
- Use a Type Into activity below the Write Line activity to store file names in an MS Word file.
- Use Click and Send Hotkey activities to save and close the file.
- STOP

5.5.3 Step by Step Process

Step 1: Open a new MS Word file.

Step 2: Open UiPath Studio.

Step 3: Create a new process and name it as "For Each Activity"

Step 4: Drag a Sequence activity from the Activities panel and drop it in the Designer panel.

Step 5: Name the Sequence activity as "Sequence – 'This code is to write all the file names in MS Word present in a particular folder.'"

Step 6: Right-click on the Sequence activity container and select *Annotations* from the context menu.

Step 7: Enter the annotation shown below:

"1. Locate and select a folder containing multiple files.

2. List the directory path of all the files in the Output panel.

3. Store the updated names in MS Word file and save and close it."

Step 8: Insert a Select Folder activity in the Designer panel and add an annotation: "User selects a folder".

Step 9: In the Variables panel, define a new variable as shown below:

Name	Variable type	Scope	Default
FolderName	String	Sequence - 'This code is to write all the file names in MS Word present in a particular folder'	

Step 10: In the Properties panel of the Select Folder activity, enter **FolderName** in the Output property.

Step 11: Insert an Assign activity below the Select Folder activity and name it as "Assign - 'File List'". Add an annotation: "Get all the file names that the user selects and stores it in a list".

Step 12: In the Variables panel, define a new variable as shown below:

Name	Variable type	Scope	Default
FileList	System.String[]	Sequence - 'This code is to write all the file names in MS Word present in a particular folder'	

Step 13: In the Properties panel of the Assign activity, enter **FileList** in the Output property.

Step 14: In the Assign activity, enter the value as shown below:

To	Value
FileList	Directory.GetFiles.(FolderName)

Step 15: Drag and Drop an Attach window activity and name it as “Attach Window – MS Word”. Add an annotation: “Attach MS Word window”.

Step 16: Click on the “Indicate Element on Screen” link and select the MS Word window opened in the background.

Step 17: Click the hamburger button and select Edit Selector. In the bottom panel of the Selector Editor, rename the title of the MS Word to ‘*’. Click OK to save the changes.

Step 18: Drag and drop a For Each activity in the Do section of the Attach Window activity, insert **item** in the first text box, and the variable **FileList** in the second text box. Add an annotation: “Loop iterates for each item in the list”.

Step 19: Drag and Drop a Write Line activity in the Body section of the For Each activity. Enter the expression: **item.ToString.** Add an annotation: “The result will appear in the Output Panel”.

Step 20: Drag and Drop a Type Into activity in the Body section of the For Each activity and name it as “Type Into - 'File List into MS Word'”. Add an annotation: “Each item in the list results to be printed in the MS Word file.”

Step 21: Click on the “Indicate element on screen” link and select the editor area of MS Word.

Step 22: In the text area of Type Into activity, enter the expression: **item.ToString + “[k(enter)]”**

Step 23: Drag a Send Hotkey activity and drop it below the For Each activity.

Step 24: Name the “Send Hotkey” activity as “Send Hotkey - ' Open the Save As dialog box.'”. Add an annotation: “Pressing F12 key opens the 'Save As' Dialog box in the MS Word”.

Step 25: Click on the “Indicate element on screen” link and select the editor area of MS Word.

Step 26: In the Send Hotkey activity, select *F12* from the dropdown for Key option.

Step 27: Insert a Click activity. Click “Indicate element inside window” link and select the Save button of the ‘Save As’ dialog box.

Step 28: Name the Click activity as “Click ‘Save’ button”. Add an annotation: “Press Save button from Save As pop Dialog box.”

Step 29: Insert second Send Hotkey activity and name it “Send Hotkey” activity as Send Hotkey - 'To close the application’” Add an annotation: “Press 'Alt+F4' key to close the MS Word window.”

Step 30: Click on the “Indicate element on screen” link and select the editor area of MS Word.

Step 31: In the Send Hotkey activity, check the box below Alt option, and choose *F4* from the dropdown of Key option.

Step 32: Save and run the workflow.

5.6 Parallel Activity

5.6.1 Objective

Build a workflow using a Parallel activity to do the following:

- Perform the following activities in parallel:
  - Search UiPath website on Google, copy about UiPath from the search result.
  - Search for "What is automation" on Google, copy the definition from the search result.
  - Search for "Automation future" on Google, copy the first search result.
- Finally, store all copied text in an MS Word file.
- Save and close the MS Word file.

5.6.2 Process Overview

- START
- Use three Sequence activities in parallel within the Parallel activity.
- In the first Sequence activity:
  - Use an Open Browser activity to open the Google website “www.google.com”
  - Use a Type Into activity to search for UiPath website.
  - Set the SendWindowMessages property to True for this Type Into activity.
  - Copy about UiPath from the search result using the Get Text activity.
- In the second Sequence activity:
  - Use an Open Browser activity to open Google website “www.google.com”
  - Use a Type Into activity to search "What is automation".
  - Set the SendWindowMessages property to True of this Type Into activity.
  - Copy the search result using the Get Text activity.
- In the third Sequence activity:
  - Use an Open Browser activity to open Google website “www.google.com”
  - Use a Type Into activity to search for “Automation is Future”.
  - Set the SendWindowMessages property to True of this Type Into activity
  - Copy the search result using the Get Text activity.
- Use an Attach window and a Type Into activity to insert text from all three parallel Sequence activities into an MS Word file.

- Use Send Hotkey activities to Save and close the MS Word file.
- STOP

5.6.3 Step by Step Process

Step 1: Open UiPath Studio.

Step 2: Create a new process and name it as “Parallel Activity”.

Step 3: Drag a Sequence activity in the designer panel.

Step 4: Name the Sequence activity as “Sequence – ‘This code is to understand the working of parallel activity.’”

Step 5: Right-click on the Sequence activity container and select *Annotations* from the context menu.

Step 6: Enter the annotation shown below:

This code performs the following activities in parallel:

1. Search UiPath website on Google, copy about UiPath from the search result.

2. Search "What is automation" on Google, copy the definition from the search result.

3. Search "Automation future" on Google, copy the first search result.

4. Finally, store all copied text in an MS Word file.

5. Save and close the MS Word file.

Step 7: Drag and drop a Parallel activity in the sequence. Add an annotation: “This activity executes several sequences in parallel.” Add three Sequence activities in the parallel activity.

Step 8: Name the first Sequence activity as “Sequence - 'First Sequence'” and add an annotation “This block of code Search UiPath website on Google, copy about UiPath from the search result.”

Step 9: Drag and drop an Open Browser activity and name it as “Open Browser - 'Opens UiPath search result webpage'”. Add an annotation: “Opens Google homepage”.

Step 10: Enter the URL - “www.google.com”.

Step 11: Open Internet Explorer browser manually. In the Do section of the Open Browser activity, add a Type Into activity, and name it as "Type Into-'UiPath'". Add an annotation: "Types 'UiPath' in the search bar".

Step 12: Click on the "Indicate element on screen" link, and select the Search text area of the search engine.

Step 13: In the text area, enter text "UiPath" and set the SendWindowMessages property checkbox to 'True' in the Properties panel of the Type Into activity.

Step 14: Drag and drop a Send Hotkey activity and place it under the Type Into activity. Name it as "Send Hotkey - Hit Enter Key" and enter the annotation "Hit enter to search.". Select *enter* from the dropdown menu of the Key option.

Step 15: Click on the "Indicate element on screen" link and select the text area of Google Search.

Step 16: Drag and drop a Get Text activity and place it under the Send Hotkey activity. Name the Get Text activity as "Get Text - 'UiPath'". Enter annotation: "This activity copies about UiPath from the search result."

Step 17: Select about UiPath from the Google search result.

Step 18: Create a variable using the Variables panel as shown below:

Name	Variable type	Scope	Default
text1	String	Sequence - 'This code is to understand the working of parallel activity'	

Step 19: In the Properties panel of the Get Text activity, enter **text1** in the Output property.

Step 20: Name the second Sequence activity as "Sequence - 'Second Sequence'" and add an annotation: "This block of code Searches "What is automation" on Google, copy the definition from the search result."

Step 21: Drag and drop an Open Browser activity and name it as "Open Browser - 'Opens 'What is Automation' search result webpage'" add an annotation: "Opens Google homepage"

Step 22: Enter the URL – "www.google.com"

Step 23: In the Do section, add a Type Into activity and name it as “Type Into- ' What is Automation '”, add an annotation: “Types 'What is Automation' in the search bar”.

Step 24: Click on the “Indicate element on screen” link and select the search section of the search engine.

Step 25: In the text area of the Type Into activity, enter the text “What is Automation”.

Step 26: Set the SendWindowMessages property’s checkbox to‘True’ in the Properties panel of the Type Into activity.

Step 27: Drag and drop a Send Hotkey activity and place it under the Type Into activity. Name it as “Send Hotkey - Hit Enter Key”. Add an annotation: “Hit enter key to search”.

Step 28: Select *enter* from the dropdown menu of the Key option of the Send Hotkey activity.

Step 29: Click on the “Indicate element on screen” link and select the search box of the search engine.

Step 30: Drag and drop a Get Text activity and place it under the Send Hotkey activity. Name the Get Text activity as “Get Text - ' What is Automation '”. Enter the annotation “This activity copies the definition from the search result.”

Step 31: Select the definition of Automation appearing on the search result webpage.

Step 32: Create a variable using the Variables panel as shown below:

Name	Variable type	Scope	Default
text2	String	Sequence - ‘This code is to understand the working of parallel activity’	

Step 33: In the Properties panel of the Get Text activity, enter **text2** in the Output property.

Step 34: Name the third Sequence activity as “Sequence - 'Third Sequence'”, add an annotation “This block of code Searches "Automation future" on Google, copy the first search result.”

Step 35: Drag and drop an Open Browser activity and name it as "Open Browser - 'Opens 'Automation is Future' search result webpage'". Add an annotation: "Opens Google Homepage".

Step 36: Enter the URL – "www.google.com"

Step 37: In the Do section of the Open Browser activity, insert a Type Into activity and name it as "Type - ' Automation is Future'". Add an annotation: "Types 'Automation is Future' in the search bar."

Step 38: Click on the "Indicate element on screen" link and select the search box of the search engine.

Step 39: In the text area of the Type Into activity, enter the text "Automation is the Future"

Step 40: Set the SendWindowMessages property's checkbox to 'True' in the Properties panel of the Type Into activity.

Step 41: Drag and drop a Send Hotkey activity and place it under the Type Into activity. Name it as "Send Hotkey - Hit Enter Key". Add an annotation "Hit Enter key to search".

Step 42: Select *enter* from the dropdown menu of the Key option.

Step 43: Click on the "Indicate element on screen" link and select the search box of the search engine.

Step 44: Drag and Drop a Get Text activity and place it under the Send Hotkey activity. Name the Get Text activity as "Get Text - ' Automation is the Future'.". Add an annotation: "This activity copies the first search result."

Step 45: Select the first search result from the search results.

Step 46: Create a variable using the Variables panel as shown below:

Name	Variable type	Scope	Default
text3	String	Sequence - 'This code is to understand the working of parallel activity'	

Step 47: In the Properties panel of the Get Text activity, enter **text3** in the Output property.

Step 48: Insert an Attach Window activity outside the Parallel activity and name it as "Attach Window 'MS Word'". Add an annotation: "Attach MS Word window"

Step 49: Click on the "Indicate element on screen" link and select the MS Word Window.

Step 50: In the Do container of the Attach Window activity add a Type Into activity and name it as "Type Into - MS Word". Add an annotation: "Types each text variable in a new line."

Step 51: Click on the "Indicate element inside window" link and select the editor area of MS Word.

Step 52: In the text area of the Type Into activity, enter the expression: **text1+ "[k(enter)]" +text2+ "[k(enter)]" + text3**

Step 53: Insert a Send Hotkey activity after the Attach Window activity. Add an annotation: "Press F12 to open the 'Save As' dialog box in MS Word window".

Step 54: Name the Send Hotkey activity as "Send Hotkey - 'Open the Save As dialog box."

Step 55: Click on the "Indicate element on screen" link and select the editor area of the MS Word.

Step 56: In the Send Hotkey activity select *F12* from the Key option.

Step 57: Insert a Click activity. Click the "Indicate element inside window" link and select the Save button of Save As dialog box.

Step 58: Name the Click activity as "Click 'Save' button". Add an annotation: "Press Save button from Save As pop Dialog box."

Step 59: Insert another Send Hotkey activity and name it as "Send Hotkey - 'To close the application'". Add an annotation: "Press 'Alt+F4' to close the MS Word window."

Step 60: Click on the "Indicate element on screen" link and select the editor area of the MS Word.

Step 61: In the Send Hotkey activity, check the box below Alt option, and choose *F4* from the dropdown of Key option.

Step 62: Save and run the workflow.

5.7 Loop in Flowchart

5.7.1 Objective

Build a workflow that asks the user for their name and two-digit lottery number and displays if they are a winner.

- Ask the name of the user and two-digit lottery number.
- Give the user five chances to enter the correct lottery number.
- If entry is below 54 and above 64, display "Enter your lottery number. **X** chance remaining." Here, **X** is the number of remaining chances for the user.
- If entry is between 54 and 64, display, "Congratulations **User**! You won the lottery." Replace **User** with the name of the user.
- If chances end before correct entry, display "Sorry, you lost. No more chances are remaining."

5.7.2 Process Overview

- START
- Use an Input Dialog activity within a Flowchart activity to get the name of the user.
- Use another Input Dialog activity to take a lottery number from the user.
- Also, display remaining chances using the above Input Dialog activity in the format "Enter your lottery number. **X** chance remaining."
- Use a Flow Decision activity to check if the input is above 54.
    - If the input is below 54, use an Assign activity to increment the count of chances used by 1.
    - If the input is above 54, use another Flow Decision activity to check if the input is below 64.
        - If the input is above 64, use an Assign activity to increment the count of chances used by 1.
        - If the input is below 64, use a Message Box activity to display "Congratulations User! You won the lottery." Replace User with the name of the user.
- Use a Flow Decision activity to check if the count of chances used by the user reaches five.

  - Use a Message Box activity to display, “Sorry, you lost. No more chances are remaining”.
- STOP

5.7.3 Step by Step Process

Step 1: Open UiPath Studio.

Step 2: Create a new process and name it as “Loop in Flowchart”.

Step 3: Drag a Flowchart activity from the Activities panel and drop it in the Designer panel.

Step 4: Name the Flowchart activity as “Flowchart - Loop in Flowchart”.

Step 5: Right-click on the Flowchart activity container and select *Annotations* from the context menu.

Step 6: Enter the annotation “This code demonstrated a workflow that asks user for his name and two-digit lottery number and displays if he is a winner.”

Step 7: Under the Start node, insert an Input Dialog activity, name it as “Input Dialog - User Name”, add an annotation: “Take input from the user” and enter values as shown below:

Title	Label
“Name”	“Enter your name”

Step 8: In the

Step 9: panel, define a new variable as shown below:

Name	Variable type	Scope	Default
name	String	Flowchart - Loop in Flowchart	
intChance	Int32	Flowchart - Loop in Flowchart	

Step 10: In the Properties panel of Input Dialog activity, enter **name** in the Output property.

Step 11: Insert another Input Dialog activity, name it as “Input Dialog - Lottery Number”, add an annotation: “Takes Lottery number as Input from user” and enter values as shown below:

Title	Label
“Lottery Number”	“Enter your lottery number.” + (5-intChance).ToString + “chance remains”

Step 12: In the Variables panel, define a new variable as shown below:

Name	Variable type	Scope	Default
intEnteredNum	Int32	Flowchart - Loop in Flowchart	

Step 13: In the Properties panel of Input Dialog activity, enter **intEnteredNum** in the Output property.

Step 14: Connect a Flow Decision activity to the “Input Dialog – Lottery Number” activity, and name it as “Flow Decision - 1”. Add an annotation: “This judge the Input whether it is greater than User input or smaller”.

Step 15: Enter the condition **54 < intEnteredNum** in the “Flow Decision 1” activity.

Step 16: Connect another Flow Decision activity with the True node of “Flow Decision – 1” activity and name it as “Flow Decision - 2”. Add an annotation: “If the user input is smaller than 64 and greater than 64 he/she wins the lottery”

Step 17: Enter the condition **intEnteredNum<64** in the “Flow Decision - 2” activity.

Step 18: At the True node of “Flow Decision - 2”, insert a Message Box activity

Step 19: In the text area of Message Box activity, name it as “Message Box - Success” and add an annotation: “User wins the lottery”.

Step 20: Enter the expression: **"Congratulations!"+ name +" You won the lottery."**

Step 21: Insert an Assign activity, name it as “Assign - ' intChance' Increment” and connect it with False nodes of “Flow Decision – 1” and “Flow Decision - 2” activities. Add an annotation “As the user fails to guess the correct number, the intChance will get incremented by 1”.

Step 22: In the Assign activity, enter values as shown below:

To	Value
intChance	intChance + 1

Step 23: Insert another Flow Decision activity and connect to a node of the Assign activity and name it as “Flow Decision - 3”. Add an annotation: “If the intChance is greater or equal to 5, the process ends and displays No more Chances left message”.

Step 24: Enter the condition **intChance<5** in the “Flow Decision – 3" activity.

Step 25: Connect the True node of “Flow Decision – 3" activity with “Input Dialog – Lottery Number” activity.

Step 26: At the False node of “Flow Decision - 3”, insert a Message Box activity and name it as “Message Box - No more chances”. Add an annotation: “No more chances - Message”.

Step 27: In the text area of the Message Box activity, enter the text "Sorry, you lost. No more chances are remaining."

Step 28: Save and run the workflow.

5.8 Try Catch

5.8.1 Objective

Build a workflow using a Try Catch activity to do the following:

- Take Name, Gender, and Age as the user input.
- Subtract current year with Age value to get the Year of Birth.
- Handle an error that occurs due to a reckless user input of an incorrect age containing the 11-digit number.
- Continue the process to display the Name, Gender, and Year of Birth of the user in a message box.

5.8.2 Process Overview

- START
- Use three Input Dialog activities within the Try Catch activity to ask for the Name, Gender, and Age of the user.
- Use an Assign activity to subtract the age from the current year to get the year of birth of the user
- Use Exception Type: System.Exception in the Catches section of the Try Catch activity to handle reckless input from the user. Store error in a string variable.
- Use a Message Box activity to display the Name, Gender, and Year of Birth of the user along with the Error, if any.
- STOP

5.8.3 Step by Step Process

Step 1: Open UiPath Studio.

Step 2: Create a new process and name it as “Try Catch activity”

Step 3: Drag a Sequence activity from the Activities panel and drop it in the Designer panel.

Step 4: Name the Sequence activity as “Sequence – 'A workflow using Try Catch activity to catch certain errors'”.

Step 5: Right-click on the Sequence activity container and select *Annotations* from the context menu.

Step 6: Enter the annotation shown below:

This block of code demonstrates a workflow using the Try Catch activity to do the following:

1. Take the Name, Gender, and Age as user input.

2. Subtract current year with Age value to get the Year of Birth.

3. Handle an error that occurs due to a reckless user input of incorrect age containing an 11-digit number.

4. Continue the process to display the Name, Gender, and Year of Birth of the user in a message box.

Step 7: Drag and drop a Try Catch activity and name it as “Try Catch - 'User Inputs'”. Add an annotation “Take input and catch error.”

Step 8: In the Try Section, insert a Sequence activity and name it as “Sequence - ‘User Inputs’”. Add an annotation “Take inputs from user.”

Step 9: Insert an Input Dialog activity within the Sequence activity, name it as “Input Dialog - User Name”, add an annotation: “Name as User Input” and enter the values as shown below:

Title	Label
“Name”	“Enter your Name”

Step 10: Insert a second Input Dialog activity below the previous “Input Dialog - User Name” activity, and name it as “Input Dialog - User Gender”, add an annotation: “User Gender as User Input” and enter the values as shown below:

Title	Label
“Gender”	“Enter your Gender”

Step 11: Insert a third Input Dialog activity below the previous “Input Dialog - User Gender” activity, and name it as “Input Dialog - User Age”, add an annotation: “User Age as User input” and enter the values as shown below:

Title	Label
“Age”	“Enter your Age”

Step 12: In the Variables panel, define five new variables as shown shown below:

Name	Variable type	Scope	Default
userName	String	Sequence - ‘A workflow using Try Catch activity to catch certain errors’	
userGender	String	Sequence - ‘A workflow using Try Catch activity to catch certain errors’	
intUserAge	Int32	Sequence - ‘A workflow using Try Catch activity to catch certain errors’	
intYearOfBirth	Int32	Sequence - ‘A workflow using Try Catch activity to catch certain errors’	
inputError	String	Sequence - ‘A workflow using Try Catch activity to catch certain errors’	

Step 13: In the Properties panel of “Input Dialog – User Name” activity, enter **userName** in the Output property.

Step 14: In the Properties panel of “Input Dialog – User Gender” activity, enter **userGender** in the Output property.

Step 15: In the Properties panel of “Input Dialog – User Age” activity, enter **intUserAge** in the Output property.

Step 16: Insert an Assign activity below the “Input Dialog – User Age” activity and name it as “Assign - Year of Birth.” Add an annotation: “Present year - User's year of birth”, and enter values as shown below:

To	Value
intYearOfBirth	Date.Today.Year() - intUserAge

Step 17: In the Catches section of the Try Catch Activity, select “System.Exception” from the dropdown.

Step 18: Insert an Assign activity in the Catches section and name it as “Assign – InputError”. Add an annotation “Store error as a string in a variable.” Enter values as shown below:

To	Value
inputError	exception.Message

Step 19: Insert a Message Box activity below the Try Catch activity and name it as “Message Box - User Details”. Add an annotation: “Print all the details”

Step 20: In the text area of the Message Box activity, enter the expression: **"Name: " + userName + vbCr + "Gender: "+ userGender + vbCr + "Year of Birth: " + intYearOfBirth.ToString + vbCr + "Error: " + inputError**

Step 21: Save and run the Project

Lesson 6 – Data Manipulation

6.1 Data Conversion

6.1.1 Objective

Build a workflow using .ToString method that converts an integer to string.

- Ask the user to input their name and age.
- Subtract the age of the user with the current year to get the user's year of birth.
- Display the name and year of birth in a message box in string format.

6.1.2 Process Overview

- START
- Use an Input Dialog activity to ask the name and age from the user.
- Use an Assign activity and subtract the age of the user from the current year to get the year of birth of the user.
- Use a Message Box activity to display: "Hello *User*, you were born in *Year*."
- Replace the *User* and *Year* with name and the year of birth of the user, respectively.
- STOP

6.1.3 Step by Step Process

Step 1: Open UiPath Studio.

Step 2: Create a new process and name it as "Data Conversion".

Step 3: Drag a Sequence activity from the Activities panel and drop it in the Designer panel.

Step 4: Name the Sequence activity as "Sequence – 'Data Conversion'".

Step 5: Right-click on the Sequence activity container and select *Annotations* from the context menu.

Step 6: Enter the annotation "This block of code demonstrates a workflow using .ToString method that converts an integer to string."

Step 7: Drag and drop an Input Dialog activity, name it as “Input Dialog - User Name”, add an annotation: “Takes User name as Input” and enter the values as shown below:

Title	Label
“Name”	“Enter your Name”

Step 8: Insert another Input Dialog activity, name it as “Input Dialog - User Age”, add an annotation: “Takes User Age as Input” and enter the values as shown below:

Title	Label
“Age”	“Enter your Age”

Step 9: In the Variables panel, create three variables as shown below:

Name	Variable type	Scope	Default
userName	String	Sequence – ‘Data Conversion’	
intUserAge	Int32	Sequence – ‘Data Conversion’	
intYearOfBirth	Int32	Sequence – ‘Data Conversion’	

Step 10: In the Properties panel of “Input Dialog - User Name” activity, enter **userName** in the Output property.

Step 11: In the Properties panel of “Input Dialog - User Age” activity, enter **intUserAge** in the Output property.

Step 12: Insert an Assign activity in the Sequence activity and name it as “Assign - Year of Birth”. Add an annotation: “Present year - User Age = User's year of birth” and enter values as shown below:

To	Value
intYearOfBirth	Now.Year - intUserAge

Step 13: Insert a Message Box activity, name it as “Message Box - User Details”, add an annotation “Print user details” and in the text area enter the expression: **"Hello "+userName+", You were born in "+intYearOfBirth.ToString**

Step 14: Save and run the workflow.

6.2 String Manipulations

This section contains two practice exercises on string manipulations. You will use -

- Format, Join, IndexOf, Split, and Substring methods in the first practice exercise.
- Contains, Split and Trim methods in the second practice exercise.

6.2.1 String Manipulations – Practice 1

6.2.1.1 Objective

Build a workflow using Format, Join, IndexOf, Split, and Substring methods that extract key information from a text and prints in a different format.

- Use the text "You always wanted to study Automation Training. The materials are available in the following places: UiPath Blog, UiPath Academy.” for extraction.
- Extract “Automation Training” from the first sentence.
- Extract “UiPath Blog” and “UiPath Academy” from the second sentence.
- Display “get Automation Training from: UiPath Blog; UiPath Academy” in a message box.

6.2.1.2 Process Overview

- START
- Use an Assign activity for the initial value of the message string variable: "You always wanted to study Automation Training. The materials are available in the following places: UiPath Blog, UiPath Academy."
- Create a new String variable **study** and use a succession of String methods to assign the course from the query:
  message.Split("."c).First.ToString.Substring(message.LastIndexOf("study"))
  - **Split("."c).First.ToString** extracts the first sentence of the String and converts it to a String
  - **Substring(message.LastIndexOf("study"))** extracts the Substring starting from "get"
- Create a new List variable **places** and use a succession of String methods to assign the places from the query:
  message.Split("."c)(1).ToString.Split(":"c).Last.ToString.Split(","c).ToList

- **message.Split(".c)(1).ToString** extracts the second sentence of the String and converts it to a String.
- **Split(":"c).Last.ToString** splits the remaining string and keeps only the last part of it.
- **Split(","c).ToList** takes each string separated by comma and adds it as an element in the List variable.

- Use a Message Box activity to display output using this expression:
  String.Format("{0} from: {1}", study ,String.Join(";", places))
  - **String.Join** is used to extract each element in the "places" List variable and display them.
- STOP

6.2.1.3 Step by Step Process

Step 1: Open UiPath Studio.

Step 2: Create a new process and name it as "String Manipulations - Practice 1"

Step 3: Drag a Sequence activity from the Activities panel and drop it in the Designer panel.

Step 4: Name the Sequence activity as "Sequence – 'String Manipulation.'"

Step 5: Right-click on the Sequence activity container and select *Annotations* from the context menu.

Step 6: Enter the annotation : "This block of code demonstrates a workflow that performs string manipulation on a pre-defined Message."

Step 7: In the Variables panel, create three variables as shown below:

Name	Variable type	Scope	Default
message	String	Sequence - String Manipulation	
study	String	Sequence - String Manipulation	
places	List<String>	Sequence - String Manipulation	

Step 8: Drag and drop an Assign activity in the Sequence activity and name it as "Assign- Message". Add an annotation "Pre-defining the message" and enter values as shown below:

To	Value
message	"You always wanted to study Automation Training. The materials are available in the following places: UiPath Blog, UiPath Academy."

Step 9: Insert another Assign activity and name it as "Assign- Extracting (Automation Training)".

Step 10: Right-click on the Assign activity container, and select *Annotations* from the context menu.

Step 11: Add an annotation shown below:

"1. Split("."c).First.ToString extracts the first sentence of the String and converts it to a String

2. Substring(message.LastIndexOf("study")) extracts the Substring from "study""

Step 12: In the Assign activity, enter values as shown below:

To	Value
study	message.Split("."c).First.ToString.Substring(message.LastIndexOf("study"))

Step 13: Insert another Assign activity and name it as "Assign – Extracting (UiPath Blog and UiPath Academy)".

Step 14: Right-click on Assign activity container, and select *Annotations* from the context menu.

Step 15: Add an annotations as shown below:

"1. message.Split("."c)(1).ToString extracts the second sentence of the String and converts it to a String.

2. Split(":"c).Last.ToString splits the remaining String and keeps only the last part of it.

3. Split(","c).ToList takes each string separated by a comma and adds it as an element in the List variable".

Step 16: In the Assign activity, enter values as shown below:

To	Value
places	message.Split(".”c)(1).ToString.Split(":"c).Last.ToString.Split(","c).ToList

Step 17: Insert a Message Box activity and right-click on Message Box container, and select *Annotations* from the context menu.

Step 18: Add an annotation "In this expression, String.Join is used to extract each element in the "places" List variable and display them".

Step 19: In the text area of the Message Box activity enter the expression: **String.Format("{0} from: {1}", study ,String.Join(";", places))**

Step 20: Save and run the workflow.

6.2.2 String Manipulations – Practice 2

6.2.2.1 Objective

Build a workflow using Split and Contains methods that extract sentences containing "RPA" from a paragraph.

- Store a paragraph in a string variable using an Assign activity.
- Store all sentences from the text in an array using a Split method.
- Loop through each sentence and identify sentences containing "RPA" using Contains method.
- Store all identified sentences in an MS Word file.

6.2.2.2 Process Overview

- START
- Use an Assign activity to store a paragraph in a string variable called **newText**.
- Use **newText.Split(".")c)** and store all sentences in an array called **newSentence**
- Use a For Each activity and iterate through each item in the array **newSentence**
- Use an If activity within the For Each activity to identify sentence that contains the string "UiPath in it. Use **item.Contains("RPA")** as condition.
- Use a Type Into activity to store result in an MS Word file.
- STOP

6.2.2.3 Step by Step Process

Step 1: Open a new MS Word file.

Step 2: Open UiPath Studio.

Step 3: Create a new process and name it as "String Manipulations – Practice 2"

Step 4: Drag a Sequence activity from the Activities panel and drop it in the Designer panel.

Step 5: Name the Sequence activity as "Sequence – 'This code is to demonstrate the use of Split and Contains method.'"

Step 6: Right-click on the Sequence activity container and select *Annotations* from the context menu.

Step 7: Enter the annotation : “This block of code demonstrates a workflow using Split and Contains methods that extract sentences containing “RPA” from a paragraph”.

Step 8: In the Variables panel, create variables as shown below:

Name	Variable type	Scope	Default
newText	String	Sequence - This code is to Demonstrate the use of Split and Contains.	
newSentence	String[]	Sequence - This code is to Demonstrate the use of Split and Contains.	

Step 9: Drag an Assign activity and Drop it into the “Sequence - This code is to Demonstrate the use of Split and Contains.”.

Step 10: Name the Assign activity as “Assign - Message”, add an annotation: “This message is stored in the variable: newText” and enter the values as shown below:

To	Value
newText	"Robotic process automation (RPA) is a software technology that makes it easy to build, deploy, and manage software robots that emulate humans actions interacting with digital systems and software. RPA streamlines workflows, which makes organizations more profitable, flexible, and responsive. RPA is noninvasive and can be rapidly implemented to accelerate digital transformation. It also increases employee satisfaction, engagement, and productivity by removing mundane tasks from their workdays. "

Step 11: Drag and drop another Assign activity and place it after the first Assign activity.

Step 12: Name the Assign activity as "Assign - Split sentences and store it in an array.", add an annotation: "String Manipulation -Split string" and enter the values as shown below:

To	Value
newSentence	newText.Split("."c)

Step 13: Drag and drop a For Each activity, add an annotation: "Loop iterates for each item in newSentence" and in the first box enter **item** and in the second box enter **newSentence**. Change the "**Type Argument**" property of For Each activity as "String".

Step 14: In the Body section of the For Each activity, drag and drop an If activity. Name it as "If - item contains word "RPA"", and add an annotation: "This activity judges whether the sentence contains word 'RPA' or not." and enter condition as **item.Contains("RPA")**

Step 15: Drag and drop an Attach window activity, place it in the **Then** section of the If activity and name it as "Attach Window - MS Word". Add an annotation: "Attach MS Word window".

Step 16: Click on the "Indicate element on screen" link and select the MS Word window.

Step 17: Drag and drop a Type Into activity in the Do container of the Attach Window activity and name it as "Type Into - MS Word". Add an annotation: "Types each item in the word file."

Step 18: Click on the "Indicate element on screen" link and select editor area of MS Word.

Step 19: In the text area of the Type Into activity enter expression: **item.ToString+ "[k(enter)]"**

Step 20: Save and run the workflow

6.3 Data Table Manipulations

6.3.1 Objective

Build a workflow using data table activities to join two library databases using matching student ID and display the output in a message box.

- Create a data table variable and populate it with student ID and name of students.
- Create another data table variable, and populate it with student ID and book names
- Join both the data tables based on matching student ID.
- Remove the student ID column and sort the final data table as per student names in alphabetical order from A to Z.
- Display the final data table containing the student and book names in a message box as a string.

6.3.2 Process Overview

- START
- Use two Build Data Table activities to create two tables. Store them in two DataTable variables called **dt_users** and **dt_overdueBooks**.
  - **dt_users** variable contain ID of students and name of user as string.
  - **dt_overdueBooks** variable contain ID of students and name of books as string.
- Use a Join Data Table activity. Choose the Inner type for the Join activity. Write the two column names to be used as Join criterion and create a new data table variable to store the output called **dt_borrowedBooks**
- Use a Remove Data Column activity to delete duplicate column – student ID – by specifying its index.
- Use a Sort Data Table activity to sort the data table based on the name of students in alphabetical order from A to Z.
- Use an Output Data Table activity to print the content of the data table to a String variable.
- Use the Message Box activity to display the output.
- STOP

6.3.3 Step by Step Process

Step 1: Open UiPath Studio.

Step 2: Create a new process and name it as “Data Table Manipulations”.

Step 3: Drag a Sequence activity from the Activities panel and drop it in the Designer panel.

Step 4: Name the Sequence activity as “Sequence – ‘A workflow using data table activities to join two library databases’”.

Step 5: Right-click on the Sequence activity container and select *Annotations* from the context menu.

Step 6: Add an annotation : “This block of code demonstrates a workflow using data table activities to join two library databases using matching student ID and display output in a message box.”

Step 7: Drag and drop a Build Data Table activity and name it as “Build Data Table – Users”. Add an annotation: “Data table contains Columns Student ID, Student Name”.

Step 8: Create a variable through Variables panel for the Build Data Table activity as shown below:

Name	Variable type	Scope	Default
dt_users	DataTable	Sequence - ' A workflow using data table activities to join two library databases'	

Step 9: In the Properties panel of the Build Data Table activity, enter **dt_users** in the Output property

Step 10: Click on the “Data Table” button and create a Data Table as shown below:

Student ID (String)	Student Name (String)

Step 11: Drag and drop a Build Data Table activity and name it as “Build Data Table – Overdue Books”. Add an annotation: “Datatable contains columns Student ID, Book Name”.

Step 12: Create a variable through Variables panel for the Build Data Table activity as shown below:

Name	Variable type	Scope	Default
dt_overdueBooks	DataTable	Sequence - ' A workflow using data table activities to join two library databases'	

Step 13: In the Properties panel of the Build Data Table activity, enter **dt_overdueBooks** in the Output property.

Step 14: Click on the "DataTable..." button and create a Data Table as shown below:

Student ID (String)	Book Name (String)

Step 15: Drag and drop Join Data Table activity and name it as "Join Data Tables - Join dt_Users and dt_overdueBooks Datatable". Add an annotation: "This activity joins both the data tables resulting in a single datatable".

Step 16: Click on the "Join Wizard" button.

Step 17: Enter **dt_users** in Input Data Table 1, and enter **dt_overdueBooks** in Input Data Table 2 text box.

Step 18: Select the Join Type as Inner, and in the Output Data Table text box, press **Ctrl + K** to create a new variable called **dt_borrowedBooks**.

Step 19: In the Column Table 1, enter the column index as '0' (column index of Data Table 'dt_users'), in the Column Table 2 enter the column index as '0' (column index of Data Table 'dt_overdueBooks') and press OK.

Step 20: Drag and drop a Remove Data Column activity. Name it as "Remove Data Column of dt_borrowedBooks". Add an annotation: "This activity removes the duplicate column of Student ID". In its Properties panel, enter 2 in ColumnIndex property and enter **dt_borrowedBooks** in DataTable property.

Step 21: Create a variable through the Variables panel as shown below:

Name	Variable type	Scope	Default
dt_OuputDataTable	DataTable	Sequence - ' A workflow using data table activities to join two library	

		databases'	

Step 22: Drag and drop a Sort Data Table activity. Name it as "Sort Data Column of dt_borrowedBooks". Add an annotation: "This activity sorts the DataTable in an ascending order." and pass the values in the Properties panel as shown below:

Input DataTable	dt_borrowedBooks
Output DataTable	dt_OutputDataTable
Index	1
Order	Ascending

Step 23: Create a variable through the Variables panel as shown below:

Name	Variable type	Scope	Default
DataTable	String	Sequence - ' a workflow using data table activities to join two library databases'	

Step 24: Drag and drop an Output Data Table activity. Name it as "Output Data Column of dt_OutputDataTable"., add an annotation: "This activity coverts DataTable's variable type from the datatable to string" and pass the values in the Properties panel as shown below:

Input DataTable	dt_OutputDataTable
Output Text	DataTable

Step 25: Drag and drop a Message Box activity, name it as "Message Box - Text", add an annotation: "Prints the String form of DataTable" and in its text area enter the variable **DataTable**.

Step 26: Save and run the workflow.

6.4 Lists

6.4.1 Objective

Build a workflow using Concat and Join method that merges two lists containing the UK and Spain city names, sorts it, capitalizes the first letter of each item, and displays it in a message box.

- Create a list containing three UK cities in all capital letters.
- Create another list containing three Spain cities in small letters.
- Merge both the lists together.
- Sort the final list in alphabetical order from A to Z.
- Capitalize only the first letter of all the items in the final list.
- Display the final list in a message box in string format.

6.4.2 Process Overview

- START
- Create two List variables called **SpainCities** and **UKCities**
- For **SpainCities**, instantiate and populate the List from the Variables Panel, using the expression: **new List (of String) from {"madrid","valencia", "barcelona"}**
- For **UKCities**, instantiate the List from the Variables Panel using the expression: **new List (of String) from {"MANCHESTER","BRISTOL","GLASGOW"}**
- Merge the two List variables in a newly created List variable using **Enumerable.Concat** method inside an Assign activity.
- Use the expression: **Enumerable.Concat(SpainCities.AsEnumerable, UKCities.AsEnumerable).ToList**.
    - **.AsEnumerable** converts the two Lists to Enumerable data type.
    - **.ToList** converts the outcome to List
- Use an Invoke Method activity to sort the elements in the outcome List by specifying 'Sort' in the 'MethodName' field.
- Use a For Each activity to iterate through each element in the **AllCities** variable and convert them to ProperCase using the StrConv method. Use the expression: **StrConv(item, VbStrConv.ProperCase)**

- Add the converted items to a newly created List variable **AllCitiesProperCase** using an Add to Collection activity.
- Use a Message Box activity to display the converted values using the String.Join method. Use the expression: **String.Join(",",AllCitiesProperCase)**.
- STOP

6.4.3 Step by Step Process

Step 1: Open UiPath Studio.

Step 2: Create a new process and name it as "Lists"

Step 3: Drag a Sequence activity from the Activities panel and drop it in the Designer panel.

Step 4: Name the Sequence activity as "Sequence – 'List Demo'"

Step 5: Right-click on the Sequence activity container and select *Annotations* from the context menu.

Step 6: Enter the annotation "Prints an ordered list containing the city names from two initial lists of city names."

Step 7: Create three variables through the Variables panel as shown below:

Name	Variable type	Scope	Default
SpainCities	List<String>	Sequence - List Demo	new List (of String) from {"madrid", "valencia", "barcelona"}
UKCities	List<String>	Sequence - List Demo	new List (of String) from {"MANCHESTER", "BRISTOL", "GLASGOW"}
AllCities	List<String>	Sequence - List Demo	
AllCitiesProperCase	List<String>	Sequence - List Demo	new List(of String)

Step 8: Drag and drop an Assign activity and name it as "Assign – Merge", Add an annotation: "Merging Lists". Enter the values as shown below:

To	Value
AllCities	Enumerable.Concat(SpainCities.AsEnumerable,UKCities.AsEnumerable).ToList

Step 9: Drag and drop an Invoke Method activity, add an annotation: "To sort the list." and enter values as shown below:

TargetType	(null)
TargetObject	AllCities
MethodName	Sort

Step 10: Drag and drop a For Each activity and name it as "For Each City". Add an annotation: "Iterates through each item in the list".

Step 11: In the first box of the For Each activity enter "item" and in the second box enter the variable **AllCities**

Step 12: In the properties panel of the For Each activity, Change the **Type Argument** to 'String'.

Step 13: In the Body section, drag and drop an Assign activity and name it as "Assign ProperCase". Add an annotation: "Assigning Proper Case"

Step 14: In the Assign activity, enter values as shown below:

To	Value
Item	StrConv(item, VbStrConv.ProperCase)

Step 15: Drag and drop an Add to Collection activity below the Assign activity. Add an annotation: "Adding AllCitiesProperCase to collection".

Step 16: In the properties panel of the Add to Collection activity enter values as shown below:

Collection	**TypeArgument**

AllCitiesProperCase	String
Item	item

Step 17: Drag and drop a Message Box activity and name it as “Message Box – AllCitiesProperCase”

Step 18: In the text area of the Message Box activity, enter the expression: **String.Join(",",AllCitiesProperCase)**

Step 19: Save and run the workflow.

Lesson 7 – Automation Concepts and Techniques

7.1 Screen Scraping

This section contains two practice exercises on screen scraping. You will use Tesseract OCR scraping method in the first practice exercise and Full Text scraping method in the second practice exercise.

7.1.1 Screen Scraping – Practice 1

7.1.1.1 Objective

Build a workflow using a Screen Scraper Wizard that scrapes text using the Tesseract OCR scraping method from an image and stores it in a Notepad.

- Search for "text images" in Google Images.
- Pick one image containing text from the search results.
- Scrape the text from the image using Tesseract OCR.
- Store text in a Notepad file.

7.1.1.2 Process Overview

- START
- Use an Open Browser activity and enter the URL www.google.com/images.
- Use a Type Into activity in the Open Browser activity and search "text images" in the URL.
- Use Screen Scraping from the Design ribbon to select an image containing text.
- Choose 'Tesseract OCR' as the Screen Scraping method. Change Scale to 5.
- Use Write Text File activity to store content in a Notepad file.
- STOP

7.1.1.3 Step by Step Process

Step 1: Open UiPath Studio.

Step 2: Create a new process and name it as "Screen Scraping – Practice 1".

Step 3: Drag a Sequence activity from the Activities panel and drop it in the Designer panel.

Step 4: Name the Sequence activity as “Sequence – ‘Demonstrating use of Screen Scraping Wizard’”.

Step 5: Right-click on the Sequence activity container and select *Annotations* from the context menu.

Step 6: Enter the annotation “This block of code demonstrates a workflow using Screen Scraper wizard that scrapes text using Tesseract OCR scraping method from an image and stores in a Notepad file.”

Step 7: Drag and drop an Open Browser activity, name it as “Open Browser - Google Images” and enter the URL – “www.google.com/images”. Add an annotation “Open Google Images in a browser.”

Step 8: In the Do container of the Open Browser activity, insert a Type Into activity and name it as “Type Into -Types 'Text Images' and hit 'Enter' key.” Add an annotation “Type "Text Images" and hit enter to start search.”

Step 9: Click on the “Indicate element on screen” link and select the search bar of the Google search engine.

Step 10: In the text area of the Type Into activity, enter the text as "text images[k(enter)]".

Step 11: Click the Screen Scraping button in the Design ribbon and select an image that contains some text from the search results.

Step 12: From the Screen Scraper Wizard window, choose Tesseract OCR as the Screen Scraping method and change the Scale to 5.

Step 13: Drag and drop a Write Text File activity inside the Screen Scraping container (below Attach Browser), write the file name as “ScarpedText.txt” and pass the variable in which the extracted text is stored. Add an annotation “Saves data from the variable into a new text file called ScrapedText.txt.”

Step 14: Save and run the workflow.

7.1.2 Screen Scraping – Practice 2

7.1.2.1 Objective

Build a workflow using the Screen Scraper Wizard that scrapes text using the Full-Text scraping method and stores it in a Notepad file.

- Search for "UiPath" in Google Search.
- Scrape information about UiPath shown on the top right of the result page using the Full-Text scraping method.
- Store text in a Notepad file.

7.1.2.2 Process Overview

- START
- Use an Open Browser activity to open the URL – "www.google.com".
- Use a Type Into activity in the Open Browser activity to indicate the search bar of Google and search for "UiPath".
- Use the Screen Scraping button in the Design ribbon and select information about UiPath shown on the right of the Google search results page.
- Choose Full Text as the Screen Scraping method.
- Use a Write Text File activity to store scraped content in a Notepad.
- STOP

7.1.2.3 Step by Step Process

Step 1: Open UiPath Studio.

Step 2: Create a new process and name it as "Screen Scraping – Practice 2".

Step 3: Drag a Sequence activity from the Activities panel and drop it in the Designer panel.

Step 4: Name the Sequence activity as "Sequence - Demonstrating use of Screen Scraping Wizard."

Step 5: Right-click on the Sequence activity container and select *Annotations* from the context menu.

Step 6: Enter the annotation “This block of code demonstrates a workflow using Screen Scraper Wizard that scrapes text using Full Text scraping method and stores in a Notepad file.”

Step 7: Insert an Open Browser activity, name it as “Open Browser - 'UiPath' Google Search” and enter the URL – “www.google.com”. Add an annotation “Open Google Search.”

Step 8: In the Do container of the Open Browser activity, drag and drop a Type Into activity and name it as “Type Into - Types 'UiPath' and hit 'Enter' key.” Add an annotation “Type UiPath in the search bar and hit enter.”

Step 9: Click on the “Indicate element on screen” link and select the search bar of the Google search engine.

Step 10: Enter the text as "UiPath[k(enter)]" in the Type Into activity.

Step 11: Click the Screen Scraping button in the Design ribbon and select information about UiPath show on the right of the Google search results page.

Step 12: In the Screen Scraper Wizard choose Full Text as the Screen Scraping method.

Step 13: Drag and drop a Write Text File activity inside the Screen Scraping container (below Attach Browser), write the file name as “ScarpedText.txt”, and enter the variable in which the extracted text is stored.

Step 14: Name the Write Text File activity as “Write in a Notepad”, and add an annotation “Create a notepad and enter data in it.”

Step 15: Save and run the workflow.

7.2 Data Scraping

7.2.1 Objective

Build a workflow using the Data Scraping wizard that scrapes blog post titles from the UiPath Blog from multiple pages.

- Open the UiPath Blog (https://www.uipath.com/blog).
- Extract all blog titles and URL by navigating through all pages.
- Store scraped data in an Excel file.

7.2.2 Process Overview

- START
- Use an Open Browser activity and enter the URL – www.uipath.com/blog.
- Use the Data Scraping button in the Design ribbon to indicate the first two blog post titles
- Rename "Column1" to "Blog Titles".
- Use the Indicate Next Link window of the Data Scraping tool to indicate the Next link in the search results page.
- Use Write CSV activity and save the data in a Notepad file.
- STOP

7.2.3 Step by Step Process

Step 1: Open UiPath Studio.

Step 2: Create a new process and name it as "Data Scraping".

Step 3: Drag a Sequence activity from the Activities panel and drop it in the Designer panel.

Step 4: Name the Sequence activity as "Sequence – 'Demonstrating the use of Data Scraping Wizard".

Step 5: Right-click on the Sequence activity container and select *Annotations* from the context menu.

Step 6: Enter the annotation "This block of code demonstrates a workflow using Data Scraping wizard that scrapes blog post titles from UiPath Blog from multiple pages."

Step 7: Drag and drop an Open Browser activity and name it as “Open Browser - UiPath Blog website.” Add an annotation “Open UiPath Blog”.

Step 8: Enter the URL – “www.uipath.com/blog”.

Step 9: Save and run the workflow to ensure that the URL opens successfully

Step 10: Click the Data Scraping button in the Design ribbon.

Step 11: Click Next in the Extract wizard window and select the title of the first blog post.

Step 12: Click Next again and select the title of the second blog post.

Step 13: Rename “Column1” to “Blog Titles” in the Data Scraping Wizard window, and click Next.

Step 14: Click “Finish” in the preview window.

Step 15: Go to the blog page, and scroll down to the bottom to locate the Load More button.

Step 16: Go to the Indicate Next Link window, and click “Yes”, indicate the Load More button.

Step 17: Insert a Write CSV activity in “Data Scraping” container, and in the “Write to what file:” text box, enter “ScrapedText.txt”.

Step 18: Name the Write CSV activity as “Write in new CSV, add an annotation “Write in a new CSV file in the background.”

Step 19: In the Write From section pass the variable in which the extracted text is stored.

Step 20: Save and run the workflow.

7.3 PDF Extraction

7.3.1 Objective

Build a workflow using a Read PDF Text activity and extract only Email IDs and Phone Numbers from a PDF file and store it in an MS Word file.

- Download the practice excel file available on rpachallenge.com.
- Convert the file to PDF to use in this exercise.
- Read data from the PDF file using a Read PDF Text activity.
- Extract only Phone Numbers and email IDs from the PDF and store it in an MS Word file.

7.3.2 Prerequisites

- ✓ Download the practice excel file from www.rpachallenge.com.
- ✓ Save the file in PDF format.
- ✓ Open a new MS Word file in the background where data will be stored.

7.3.3 Process Overview

- START
- Use a Read PDF Text activity to read the contents of the PDF file and store it in a string variable.
- Use a Matches activity below the Read PDF Text activity.
  - In RegEx column, select **Email**.
- Use a For Each activity to iterate through each email item and store it in an MS Word file using a Type Into activity.
- Use another Matches activity below previous Matches activity.
  - In RegEx column, select **Advanced**.
  - In Value column, enter the expression: **(407)([0-9])**
- Use a For Each activity to iterate through each phone number item and store it in an MS Word file using the Type Into activity.
- STOP

7.3.4 Step by Step Process

Step 1: Open a new MS Word file.

Step 2: Open UiPath Studio.

Step 3: Create a new process and name it as "PDF Extraction".

Step 4: Drag a Sequence activity from the Activities panel and drop it in the Designer panel.

Step 5: Name the Sequence activity as "Sequence – 'PDF Extraction"

Step 6: Right-click on the Sequence activity container and select *Annotations* from the context menu.

Step 7: Enter the annotation "This block of code demonstrates a workflow using Read PDF Text activity and extract only Email IDs and Phone Number from a PDF file and store in an MS Word file."

Step 8: Drag and drop a Read PDF Text activity and name it as "Read PDF Text - Read 'challenge.pdf'". Add an annotation "Read data from challenge.pdf".

Step 9: Click the folder button and select the pdf file located in the folder.

Step 10: Create a variable through the Variables panel as shown below:

Name	Variable type	Scope	Default
pdfOutput	String	Sequence – PDF Extraction	

Step 11: Enter the variable **pdfOutput** in the Output property of the Read PDF Text activity's Properties panel.

Step 12: Insert a Matches activity and name it as "Matches – Email". Add an annotation "Use regular expression to identify email in the text."

Step 13: Click the *Configure Regular Expressions* button.

Step 14: In the RegEx column, select **Email**.

Step 15: In the Quantifiers column, select Any (0 or more).

Step 16: Exit the RegEx Builder window after updating as per Step 14 and 15.

Step 17: In the Properties panel of the Matches activity, press **Ctrl + K** in the Result property, and enter a new variable called **emails**. Enter **pdfOutput** in the Input property.

Step 18: Insert another Matches activity and name it as "Matches – Phone Number". Add an annotation "Use regular expression to identify phone number in the text."

Step 19: Click the *Configure Regular Expressions* button.

Step 20: In the RegEx column, select **Advanced**.

Step 21: In Quantifiers, select Any (0 or more).

Step 22: In the Value column, enter the expression: **(407)([0-9])**

Step 23: Exit the RegEx Builder window after updating as per Step 20 to 22.

Step 24: In the Properties panel of the Matches activity, press **Ctrl + K** in the Result property, and enter a new variable called **phones**. Enter **pdfOutput** in the Input property.

Step 25: Drag and drop a For Each activity, name it as "For Each - Email". Add an annotation "Iterate through each item in variable emails."

Step 26: In the first text box enter **"item"** and in the second box enter "**emails"**.

Step 27: Insert an Attach Window activity in the Body section of the For Each activity and name it as "Attach Window MS Word (Email)". Add an annotation "Attach an MS Word file". Click the "Indicate window on screen" link and indicate the MS Word window.

Step 28: Click the hamburger button of the Attach Window and open Selector Editor. Replace the title with an asterisk (*). It will make the workflow run for MS Word files with varying names.

Step 29: Insert a Type Into activity in the Do section of the Attach Window activity and name it as "Type Into - MS Word File". Add an annotation "Type data into MS Word file."

Step 30: Click on the "Indicate element on screen" link and select the text editor area of MS Word.

Step 31: In the text area of the Type Into activity, enter **item.ToString + vbCr**.

Step 32: Insert another For Each activity, name it as "For Each – Phone Number". Add an annotation "Iterate through each item in variable phones."

Step 33: In the first text box, enter **item** and in the second box enter **phones**.

Step 34: Insert an Attach Window activity in the Body section of the second For Each activity and name it as "Attach Window MS Word (Phone Number)". Add an annotation "Attach an MS Word file". Click the "Indicate window on screen" link and indicate the MS Word window.

Step 35: Click the hamburger button of the Attach Window activity and open Selector Editor. Replace the title with asterisk (*). It will make the workflow run for any MS Word files with varying names.

Step 36: Insert a Type Into activity in the Do section of the Attach Window activity and name it as “Type Into - MS Word File”. Add an annotation “Type data in the MS Word file.”

Step 37: Click on the “Indicate element on screen” link and select text editor area of MS Word.

Step 38: In the text area of Type Into activity, enter **item.ToString + vbCr**.

Step 39: Save and run the workflow.

7.4 Workbook Automation

7.4.1 Objective

Build a workflow using Read Range and Append Range activity to read data from a workbook and append data to another workbook.

- Create an excel file containing names of any five cities in small letters.
- Read the data from the file using the Read Range activity.
- Convert all city names in capital letters.
- Add the updated names in a new spreadsheet using the Append Range activity.

7.4.2 Prerequisites

✓ Create an excel file containing names of five cities in small letters under header City Names.

7.4.3 Process Overview

- START
- Use a Read Range activity from the Workbook category and to read data from the excel file path and store in a data table variable.
- Use an Add Data Column activity add a new column in the data table called "City in Capitals".
- Use a For Each Row activity to iterate through each row item in the data table.
- Use an Assign activity to convert each row item to uppercase.
- Use an Append Range activity below the For Each activity to store updated data tables with city names in capitals in a new excel file.
- STOP

7.4.4 Step by Step Process

Step 1: Open UiPath Studio.

Step 2: Create a new process and name it as "Workbook Automation".

Step 3: Drag a Sequence activity from the <u>Activities</u> panel and drop it in the <u>Designer</u> panel.

Step 4: Name the Sequence activity as “Sequence – ‘Workbook Automation’”.

Step 5: Right-click on the Sequence activity container and select *Annotations* from the context menu.

Step 6: Enter the annotation “This block of code demonstrates a workflow using Read Range and Append Range activity to read data from a workbook and append data to another workbook.”

Step 7: Drag and drop a Read Range activity from the workbook category. Click on the ellipses button and select the Excel file with City Names.

Step 8: Name the Read Range activity as “Read Range of data in Excel”, and add an annotation “Read data from the excel file and store in data table variable cityNames.”

Step 9: Empty the Range section of the Read Range activity.

Step 10: Create a variable through the Variables panel as shown below:

Name	Variable type	Scope	Default
dt_cityNames	DataTable	Sequence - Workbook Automation	

Step 11: Enter the variable “dt_cityNames” in the DataTable section of the Read Range activity’s Properties panel.

Step 12: Drag and drop an Add Data Column activity below the Read Range activity and pass the expressions in the Properties panel as shown below:

ColumnName	“City in Capitals”
DataTable	dt_cityNames

Step 13: Name the Add Data Column activity as “Add Data Column in cityNames”, and add an annotation “Add a new Data Column in the data table cityNames.”

Step 14: Drag and drop a For Each Row activity below the Add Data Column activity and pass the *VB expression* as **dt_cityNames**.

Step 15: Name the For Each Row activity as “For Each Row in cityNames”, and add annotation “Iterate through each item in cityNames.”

Step 16: Inside the Body section of the For Each Row activity, insert an Assign activity, and enter values as shown below:

Title	Label
row(1)	row(0).ToString.ToUpper

Step 17: Name the Assign activity as “Assign Uppercase”, and annotation “Change case of values to uppercase”.

Step 18: Insert a Write Range activity below the For Each Row activity and in the Properties panel enter values as shown below:

WorkbookPath	“CityinCapitals.xlsx
DataTable	dt_cityNames
Starting Cell	“”
SheetName	“Sheet1”

Step 19: Name the Write Range activity as “Write Range in excel file”, and name it as “Write values in excel file from cityNames.”

Step 20: In the properties panel of the Write Range activity, Check the checkbox for Add Headers.

Step 21: Save and run the workflow.

7.5 Excel Automation

7.5.1 Objective

Build a workflow that calculates the total monthly deposit of a bank from an Excel file and store output in a new sheet.

- Download the Excel file link given for practice.
- The file contains three deposit categories – Cash In, On-Us Check, and Not On-Us Check.
- Calculate the total amount received in all three categories for June.
- Store calculated values in a new sheet in the same excel file.

7.5.2 Prerequisites

✓ Download practice excel file from https://www.uipath.com/hubfs/Documentation/WorkflowExamples/QueueItem_Example_Reports.xlsx

7.5.3 Process Overview

- START
- Use a Build Data Table activity and in its Properties panel, enter a new variable **dt_sumTable**.
- Create a data table with three integer columns with names “Cash In Total”, “On-Us Check Total”, and “Not On-Us Total”.
- Use an Excel Application Scope activity and select the downloaded excel file.
- Use a Read Range activity in the Do container to read excel data from the “June Report” sheet.
- Use an Assign activity within a For Each Row activity below the Read Range activity
    - In the first text box, enter the new integer variable **intSum1**.
    - In the second text box, enter expression **CInt(row (1)) + intSum1**.
- Use a second Assign activity in the Body section
    - In the first text box, enter the new integer variable **intSum2**.
    - In the second text box, enter expression **CInt(row (2)) + intSum2**.

- Use athird Assign activity in the Body section.
    - In the first text box, enter the new integer variable **intSum3**.
    - In the second text box, enter expression **CInt(row (3)) + intSum3**.
- Use an Add Data Row activity below the For Each Row activity. Go to its Properties panel, and in the ArrayRow property enter the expression: {intSum1, intSum2, intSum3}. In the DataTable property enter "**dt_sumTable"**.
- Use a Write Range activity below the Add Data Row activity,
    - Insert "June Total" in the first text box,
    - **dt_sumTable"** in the second text box.
    - Check the box of Add Headers property in the Properties panel
- STOP

Step 1: Open UiPath Studio.

Step 2: Create a new process and name it as "Excel Automation".

Step 3: Drag a Sequence activity from the Activities panel and drop it in the Designer panel.

Step 4: Name the Sequence activity as "Sequence - 'Excel Automation (Read/Write)'"

Step 5: Right-click on the Sequence activity container and select *Annotations* from the context menu.

Step 6: Enter the annotation "This block of code demonstrates a workflow that calculates total monthly deposit of a bank from an Excel file and store output in a new sheet."

Step 7: Drag and drop a Build Data Table activity and name it as "Build Data Table – sumTable." Add an annotation "Create an initial data table called sumTable."

Step 8: Click on the 'DataTable…' button and create three integer columns named as "Cash In Total", "On-Us Check Total", and "Not On-Us Total".

Step 9: Create a variable through Variables panel as shown below:

Name	Variable type	Scope	Default
dt_sumTable	DataTable	Sequence - Excel Automation (Read/Write)	

Step 10: Enter the variable **dt_sumTable** in the DataTable property of the Build Data Table activity's Properties panel.

Step 11: Drag and drop an Excel Application Scope activity. Click on its ellipses button and select the downloaded excel file.

Step 12: Name the Excel Application Scope activity as "Read/Write Excel, and add an annotation "Read data from QueueItem_Example_Reports.xlsx and sum row values."

Step 13: Drag and drop a Read Range activity in the Do container of the Excel Application Scope activity. Change sheet name to "June Report" in the first text box of Read Range activity.

Step 14: Name the Read Range activity as "Read Range of Excel", and add an annotation "Read data from the excel file from June Report sheet and store in juneReport data table."

Step 15: Create a variable through the Variables panel as shown below:

Name	Variable type	Scope	Default
dt_juneReport	DataTable	Sequence - Excel Automation (Read/Write)	

Step 16: Enter the variable **dt_juneReport** in the DataTable section of the Read Range activity's Properties panel.

Step 17: Drag and drop a For Each Row activity below the Read Range activity. In the *Vb Expression* area enter the variable **dt_juneReport**.

Step 18: Name the For Each Row activity as "For Each Row in dt_juneReport", and add an annotation "Iterate through each row in data tabel dt_juneReport"

Step 19: Create three variables through Variables panel as shown below:

Name	Variable type	Scope	Default
intSum1	Int32	Do	
intSum2	Int32	Do	
intSum3	Int32	Do	

Step 20: Insert an Assign activity inside the Body section of the For Each Row activity. Name it as "Assign 2nd Column", and add annotation "Sum row items from 2nd column of the data table."

Step 21: In the Assign activity, enter values as shown below:

To	Value
intSum1	CInt(row (1))+intSum1

Step 22: Insert a second Assign activity. Name it as "Assign 3rd Column", and add annotation "Sum row items from 3rd column of the data table."

Step 23: In the second Assign activity, enter values as shown below:

To	Value
intSum2	CInt(row (2))+intSum2

Step 24: Insert a third Assign activity. Name it as "Assign 4th Column", and add annotation "Sum row items from 4th column of the data table."

Step 25: In the third Assign activity, enter the values as shown below:

To	Value
intSum3	CInt(row (3))+intSum3

Step 26: Insert an Add Data Row activity below the For Each Row activity. Name it as "Add Data Row in sumTable", and annotation "Add row in specified data table."

Step 27: In the Properties panel of the Add Data Row activity, enter values as shown below:

ArrayRow	{intSum1, intSum2, intSum3}
DataTable	dt_sumTable

Step 28: Drag and drop a Write Range activity below the Add Data Row activity, and enter values as shown below:

Sheet Name	Data Table
“June Total”	dt_sumTable

Step 29: Name the Write Range activity as “Write Range in New Sheet”, and add annotation “Write range in a sheet in the excel sheet using dt_sumTable data table.”

Step 30: In the Properties panel of the Write Range activity, check the box of AddHeaders property.

Step 31: Save and run the workflow.

7.6 Email Automation

7.6.1 Objective

Build a workflow that extracts attachments from emails containing the word "Resume" in its subject.

- Set up IMAP configuration to access the email.
- Loop through each email to identify subjects containing the word "Resume".
- Download the attachments from the identified emails in a folder.

7.5.2 Prerequisites

✓ Send 4-5 dummy emails to the email ID that you will use for practice. Emails must contain file attachments and the word "Resume" in their subject.

7.6.3 Process Overview

- START
- Use a Get IMAP Mail Messages activity to retrieve email data.
- Use a For Each activity to iterate through each retrieved email.
- Use an If activity within the For Each activity and check for the word "Resume" in the Subject.
- Use a Save Attachment activity within the If activity to store attachments that contain the word "Resume" in the Subject.
- STOP

7.6.4 Step by Step Process

Step 1: Open UiPath Studio.

Step 2: Create a new process and name it as "Email Automation".

Step 3: Drag a Sequence activity from the Activities panel and drop it in the Designer panel.

Step 4: Name the Sequence activity as "Sequence – Email Automation".

Step 5: Right-click on the Sequence activity container and select *Annotations* from the context menu.

Step 6: Enter the annotation “This block of code demonstrates a workflow that extracts attachments from the emails containing the word “Resume” in its subject.”

Step 7: Create a variable through the Variables panel as shown below:

Name	Variable type	Scope	Default
mailResumes	List<MailMessage>	Sequence - Email Automation	

Step 8: Click “Manage Packages” and check for any update available for Email Activities package, if it is available, update to the latest and save the file.

Step 9: Drag and drop a Get IMAP Mail Messages activity. Name it as “Get IMAP Mail Messages from Email”, and annotation “Receive mails from the emails and store in mailResumes.”

Step 10: In the Properties panel of the Get IMAP Mail Messages activity pass the values as shown below:

Mail Folder	“Inbox”
Port	993
Server	“imap.gmail.com”
Email	Enter your email address
Password	Enter your password
Messages	mailResumes

Step 11: Insert a For Each activity and pass the *VB expression* as **mailResumes**.

Step 12: Name For Each activity as “For Each Mail Item”, and add an annotation “Iterate through each email item in mailResumes”. Set the Type argument of For each activity to "System.Net.Mail.MailMessage".

Step 13: In the Body section of the For Each activity, insert an If activity and enter the condition **item.Subject.ToLower.Contains("resume")**. Name it as “If contains Resumes”, and add an annotation “Identify mails that contain the word ‘resume in its subject.”

Step 14: In the Then section of the If activity, insert a Save Attachments activity. Name it as “Save Attachments from Email” and add an annotation “Save attachments from email in specified folder.”

Step 15: In the first box enter **item**

Step 16: In the second box enter **Environment.CurrentDirectory.ToString**

Step 17: Save and run the workflow.