Advanced RPA Design and Development

v4.0

UiPath

Lesson 06

Debugging

Exam Topics - Debugging

1. Use debug modes, debug actions and the debug ribbon option to debug a file or the entire project

2. Use simple and conditional breakpoints and simple and conditional trace points

3. Describe and use all debugging panels

4. Use the Profile Execution feature to improve execution performance

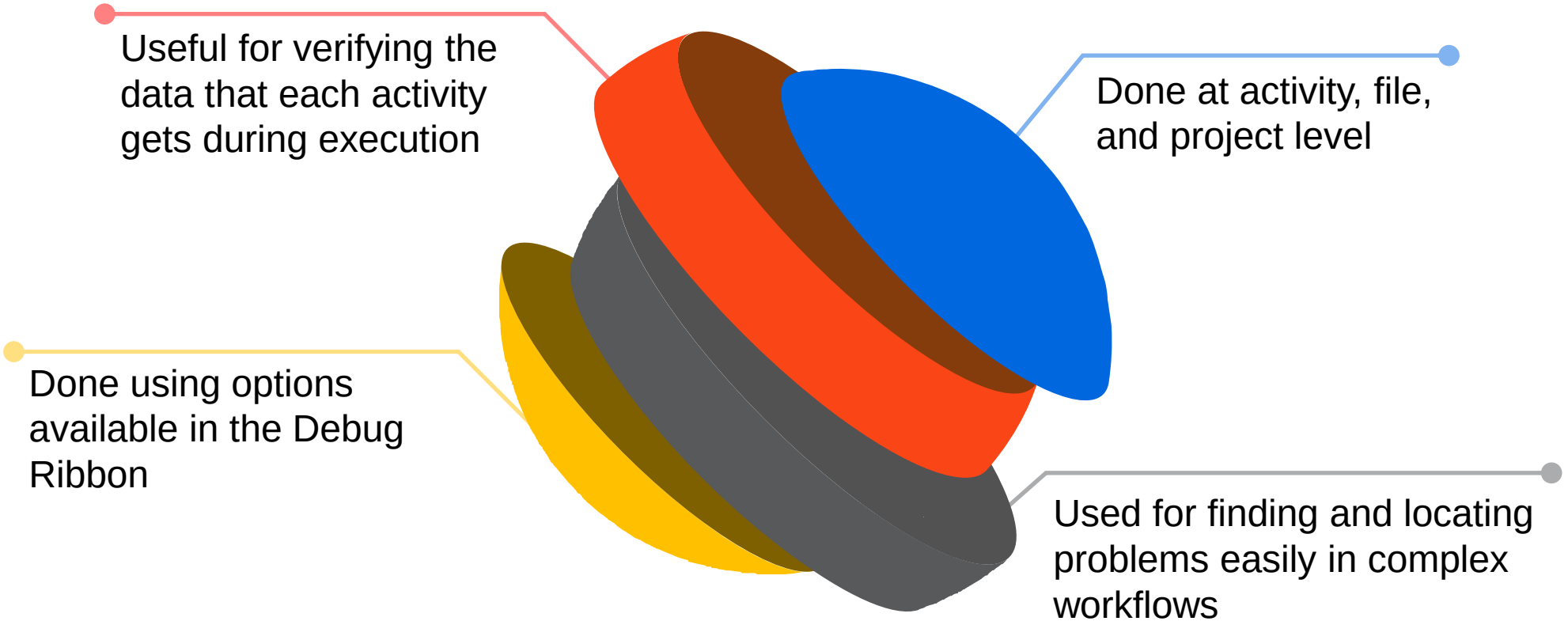
Debugging a Workflow

- Automation Debugging
- Troubleshoot, Debug, and Modify Processes
- Debugging Actions
- Setting Breakpoints
- Logging
- Logging Levels
- How to Do Logging?

UiPath

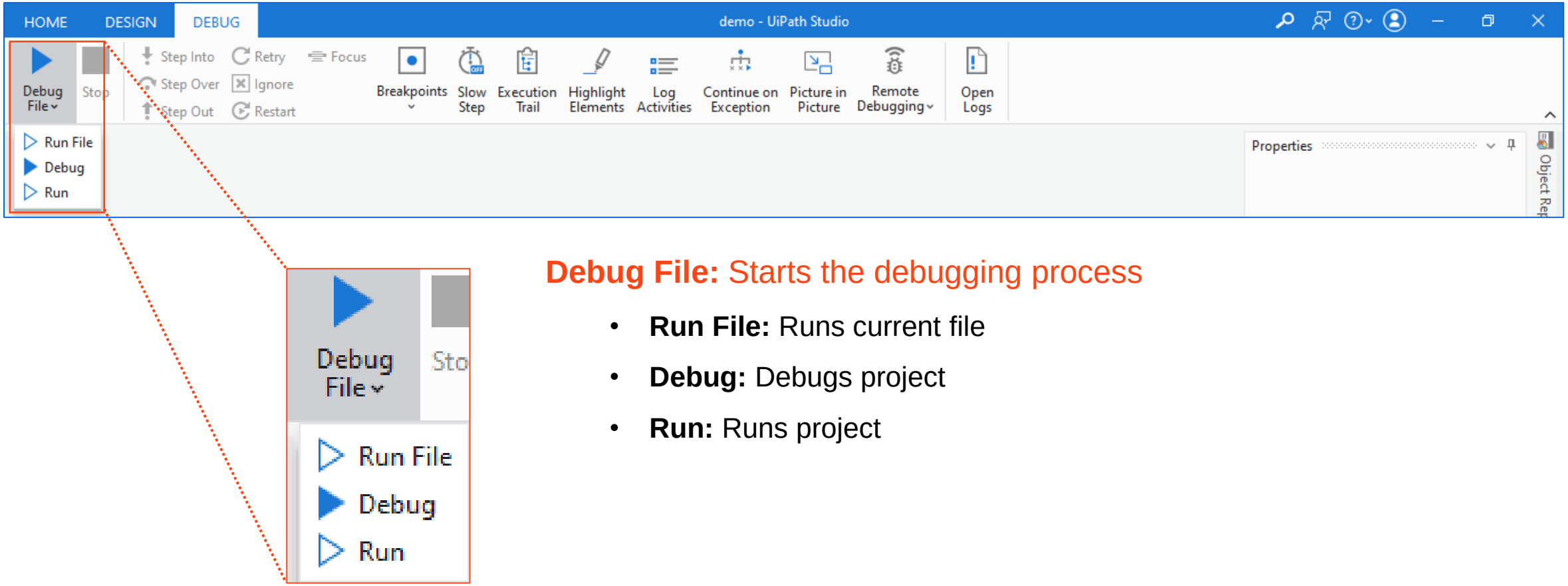
Automation Debugging

Debugging is the process of identifying and removing the errors which prevent the project from functioning correctly. In Studio, debugging is:



Troubleshoot, Debug, and Modify processes

The Debug tool in Studio is a real-time engine that checks for errors while working with the workflow. Whenever an activity has errors, Studio Process Designer notifies and gives details about the issues encountered.



Debug File: Starts the debugging process

- **Run File:** Runs current file
- **Debug:** Debugs project
- **Run:** Runs project

Debugging Actions

The actions for debugging are:

Step Into
Debugs activities step-by-step

Step Over
Debugs the next activity without opening it

Step Out
Pauses execution at current container

Retry
Re-executes previous activity

Ignore
Ignores an exception and executes from the next activity

Restart
Restarts debugging from the first activity of the project

Break
Pauses the debugging process

Focus
Returns to the activity that caused error and resumes debugging

Slow Step
Debugging at a slower rate, takes a closer look at the activity

Debugging Actions (Contd.)

The actions for debugging are:

Execution Trail
Shows the exact execution path at debugging

Log Activities
Displays debugged activities as Trace logs in the Output panel

Picture in Picture
Starts the process in a separate session

Highlight Elements
Highlights UI elements during debugging

Continue on Exception
Logs the exception in the Output panel and continues the execution

Remote Debugging
Runs and debugs attended and unattended processes remotely

Open Logs
Opens local folder where the logs are stored

Debugging - Best practices

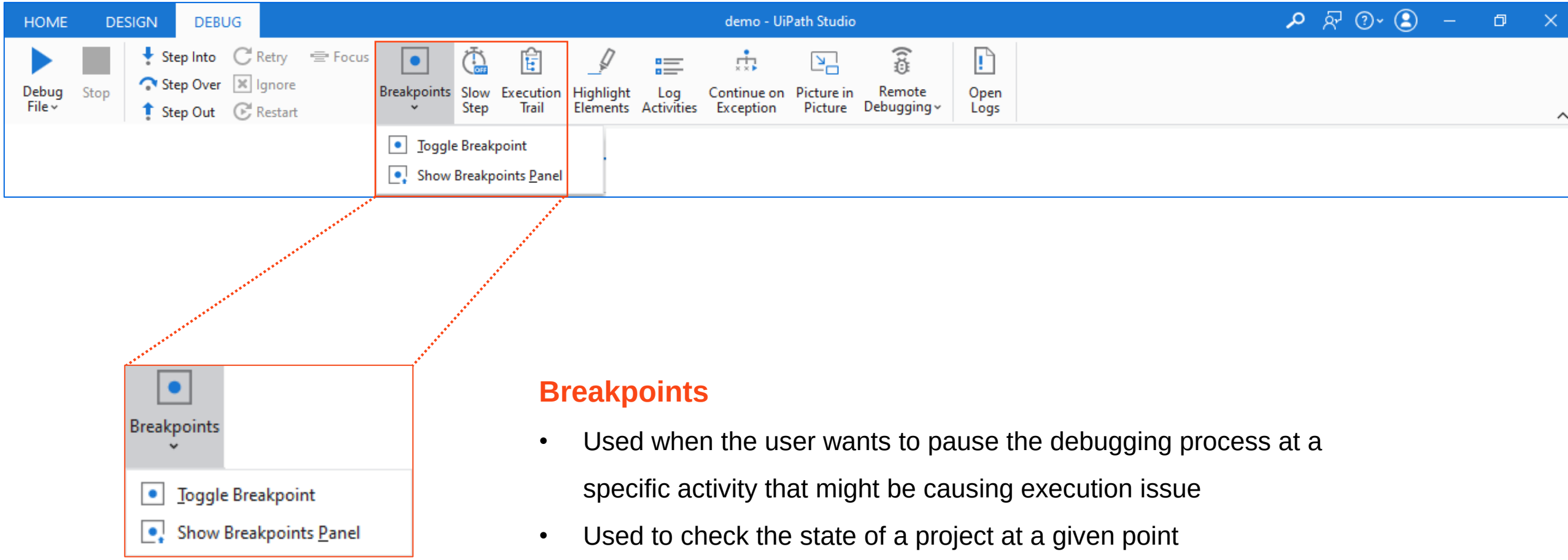
From the get-go, here are some of the best practices you should keep in mind:

- Don't build lengthy sequences, since it'll become difficult to understand, debug, and maintain
- Using Log Message activities to trace the evolution of a running process is essential for supervising, diagnosing, and debugging a process
- Avoid adding code on transitions. It isn't visible, and therefore difficult to debug and maintain
- After each component is built, unit testing should be conducted. If every component is thoroughly tested, the integration runs more smoothly, and debugging lasts for a shorter period of time

Debugging Resources

Feature	Link
About Debugging	https://docs.uipath.com/studio/standalone/2023.4/user-guide/about-debugging
Debugging Actions	https://docs.uipath.com/studio/standalone/2023.4/user-guide/debugging-actions
The Locals Panel Learn more about the Locals panel.	https://docs.uipath.com/studio/standalone/2023.4/user-guide/the-locals-panel
Remote Debugging Learn more about remote debugging and the connection types you can use.	https://docs.uipath.com/studio/standalone/2023.4/user-guide/remote-debugging

Setting Breakpoints



Breakpoints

- Used when the user wants to pause the debugging process at a specific activity that might be causing execution issue
- Used to check the state of a project at a given point

Types of Breakpoints

- Simple Breakpoint - pauses the execution of an activity.
- Conditional Breakpoint - pauses the execution of an activity when a condition and/or a hit count are met.
- Simple trace point - simple breakpoint with logging.
- Conditional trace point - conditional breakpoint with logging.

Types of Breakpoints

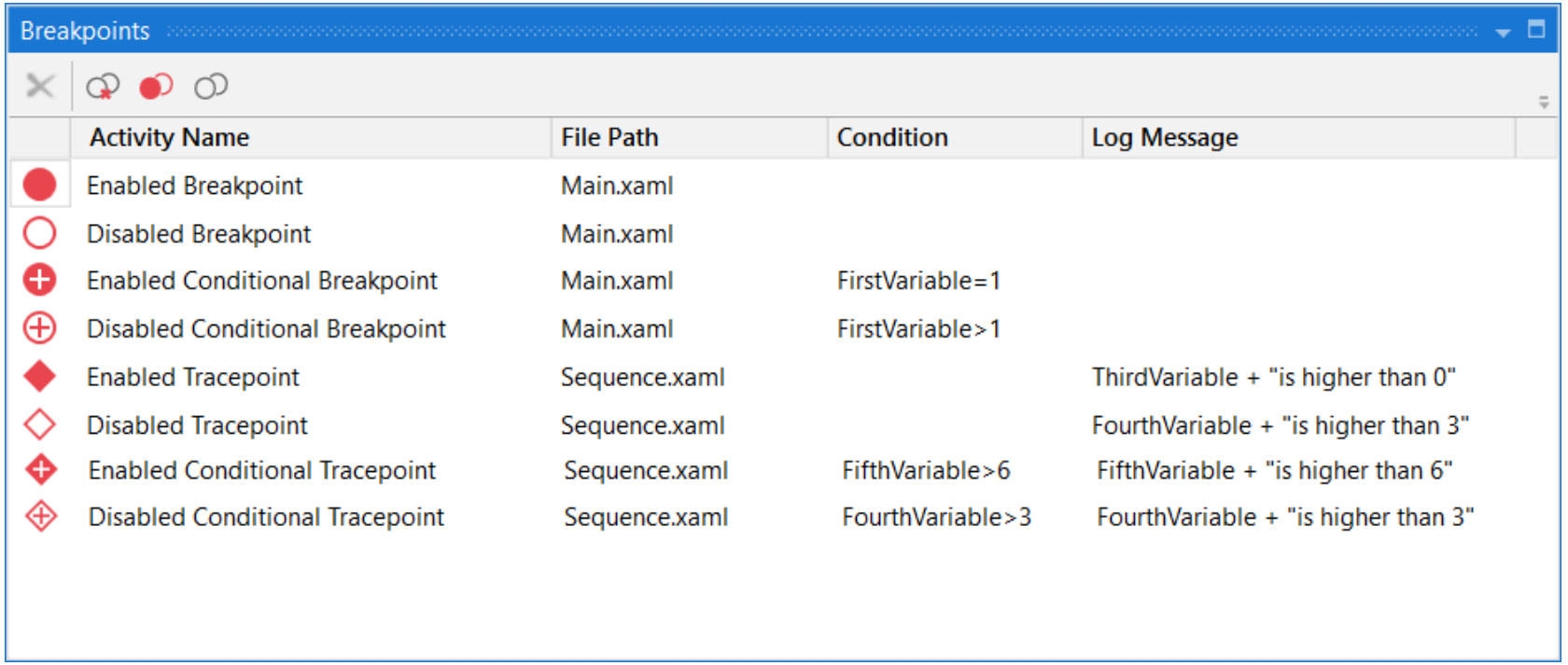
Breakpoints are used to pause the debugging process on an activity which may cause execution issues.

- A breakpoint will pause the execution **before running the activity or the container that holds it**.
- Breakpoints don't affect execution in run mode
- **A conditional breakpoint** is a simple breakpoint with a condition or a hit count
- **A simple trace point** is a simple breakpoint with logging
- **A conditional trace point** is a conditional breakpoint with logging

The Breakpoints Panel

You can place and modify a breakpoint on any activity as follows:

- from the context menu, right-click an activity and select **Toggle Breakpoint**;
- by selecting the activity, and clicking the **Breakpoints** button on the **Debug** tab;
- by pressing F9 while the desired activity is selected



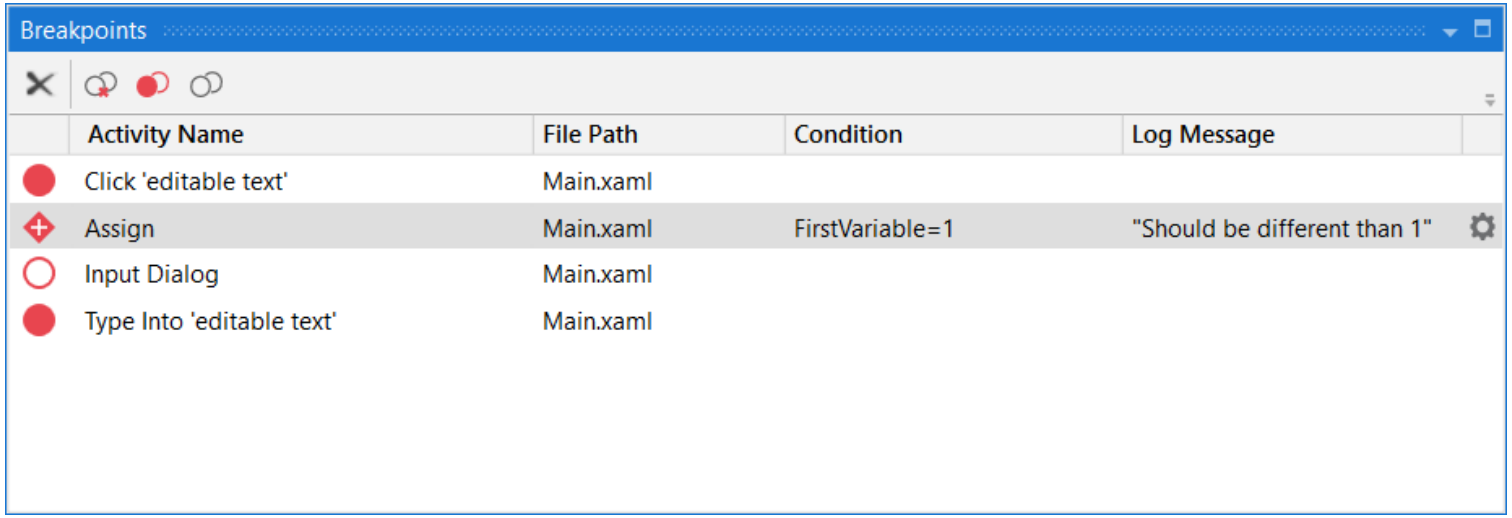
Activity Name	File Path	Condition	Log Message
Enabled Breakpoint	Main.xaml		
Disabled Breakpoint	Main.xaml		
Enabled Conditional Breakpoint	Main.xaml	FirstVariable=1	
Disabled Conditional Breakpoint	Main.xaml	FirstVariable>1	
Enabled Tracepoint	Sequence.xaml		ThirdVariable + "is higher than 0"
Disabled Tracepoint	Sequence.xaml		FourthVariable + "is higher than 3"
Enabled Conditional Tracepoint	Sequence.xaml	FifthVariable>6	FifthVariable + "is higher than 6"
Disabled Conditional Tracepoint	Sequence.xaml	FourthVariable>3	FourthVariable + "is higher than 3"

A single activity needs to be selected for a breakpoint to be toggled. You can, however, toggle as many breakpoints as you see fit.

Make sure that the order of activities in the workflow is not changed after the breakpoint is set.

Icons :
Each breakpoint or tracepoint receives a specific icon based on its state. The icon is set on the activity and visible in the **Breakpoints** panel.

The Breakpoints Panel



The Breakpoints panel displays all breakpoints in the current project, together with the file in which they are contained. The Activity Name column shows the activity with the toggled breakpoint, while the File Path column displays the file and its location.

The Condition column displays conditions set to breakpoints. The Log Message column shows messages to be logged if the condition is met. Hover over the breakpoint tag on an activity to view its condition and log message.

Double-click on a breakpoint to see the activity highlighted in the Designer panel. Use context menu options or the Breakpoints button in the ribbon to enable or disable breakpoints.

To delete multiple breakpoints, select them and click Delete in the context menu, or the Delete button in the panel. This removes the breakpoints from the current file.

The Delete all, Enable all and Disable all breakpoints buttons perform actions on all breakpoints listed in the panel, regardless if they are selected or not.

Breakpoints Types

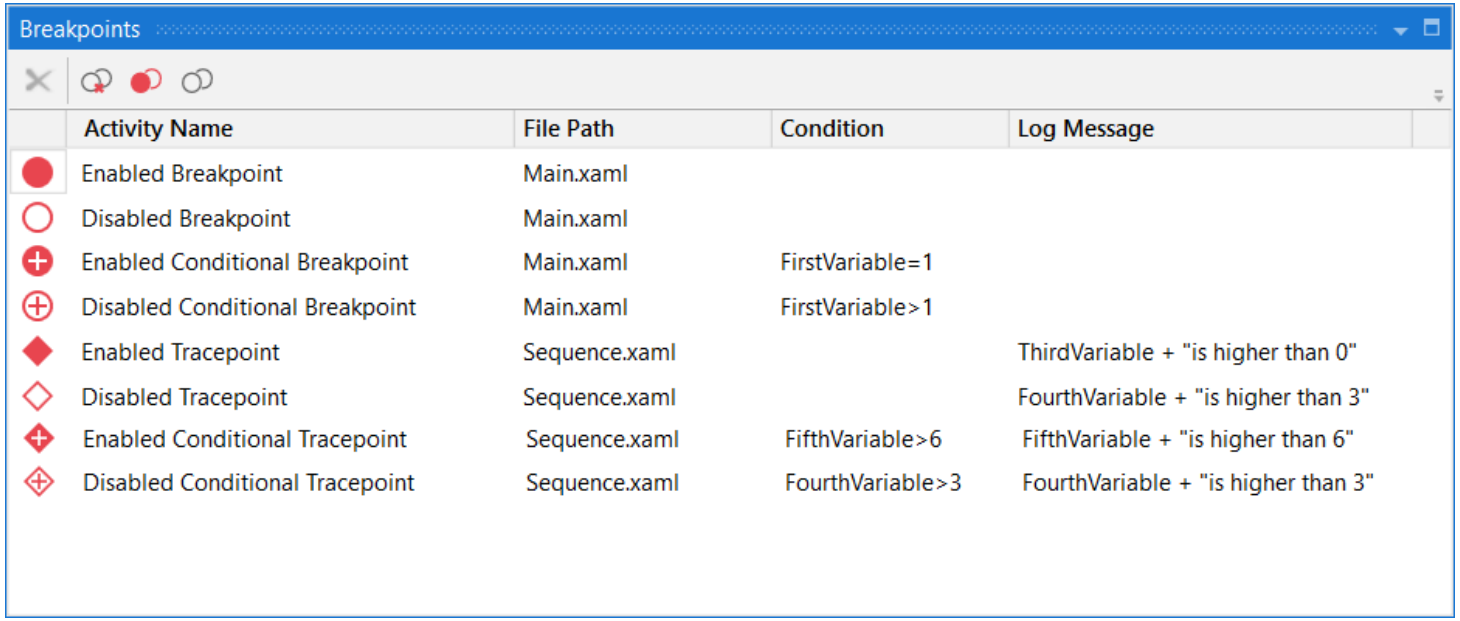
Type	Description
Breakpoints	Breakpoints pause the debugging process before the activity is executed. Breakpoints can have the following states: • Enabled -  • Disabled - 
Conditional Breakpoints	Conditional breakpoints are breakpoints that depend on a set condition and/or a hit count. Conditional breakpoints can have the following states: • Enabled -  • Disabled - 
Tracepoints	Tracepoints are breakpoints with set logged messages. When the tracepoint is reached during debugging, the message is logged at trace level. Tracepoints can have the following states: • Enabled -  • Disabled - 
Conditional Tracepoints	Conditional tracepoints have a set condition or hit count, and a logged message. The message is logged when the condition is met the number of times stated in the hit count field. Conditional tracepoints can have the following states: • Enabled -  • Disabled - 

To modify the state of a breakpoint or tracepoint select the activity and press F9, click the icon in the **Breakpoints** panel, or use the **Designer** or **Breakpoints** panel context menus.

You can also click the **Breakpoints** button on the **Debug** tab, open the drop-down menu and click **Toggle Breakpoint**.

Note: Breakpoints set during design time persist when reopening the automation project. Breakpoints don't persist at runtime, only at debugging.

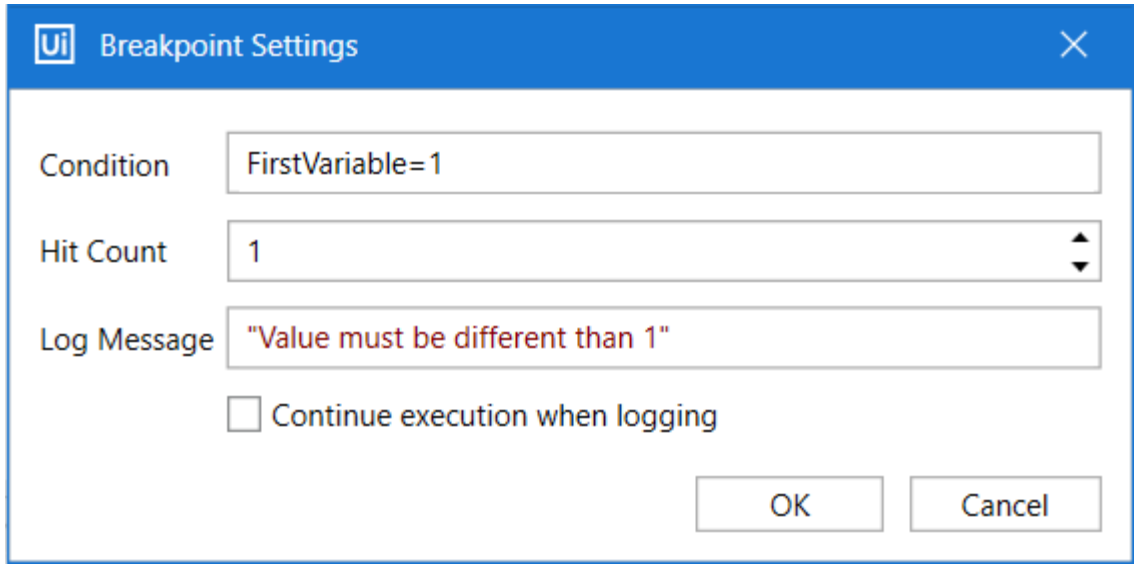
Breakpoints : Delete, Enable, Disable



- **Delete a breakpoint :** Select a breakpoint in the panel and click the Delete button to remove it
- **Delete all breakpoints option** enables you to delete all the breakpoints in the current project
- **The Enable all breakpoints option** helps you enable all breakpoints in the currently opened project
- Multiple selection is available in the Breakpoints panel

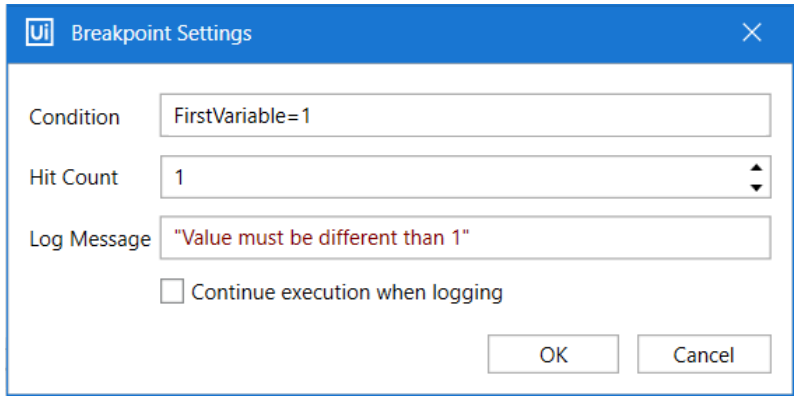
Breakpoint Settings

The Breakpoints panel comes with a set of settings that can be individually adjusted for each toggled breakpoint part of the automation project. Click the icon to open the window.



Note: Please take into consideration that any expression added in the **Condition** field is not validated.

Field Descriptions for Breakpoint Settings



The **Breakpoint Settings** window has the below options:

Option	Description
Condition	The condition for the breakpoint. If the condition is met during debugging, the execution breaks and the activity is highlighted.
Hit Count	Specifies the number of times the condition must be met before the execution breaks. If the hit count is higher than the number of times the condition can be met, the execution does not stop upon encountering the breakpoint. The maximum hit count value is 32,767.
Log Message	Specifies the message to be logged at trace level when the condition is met. The message is visible in the **Output** panel. If a condition is not set, the message is still logged.
Continue execution when logging	If selected, the execution is not paused when the condition is met and the specified message is logged. Available only if a log message was previously set.

Settings for any breakpoint in the project are visible upon hovering the breakpoint in the **Designer** panel.

UiPath Academy

Chapter 6:

Debbuging in Studio

Breakpoints

Breakpoint Resources

Feature	Link
Breakpoints	https://docs.uipath.com/studio/standalone/2023.4/user-guide/the-breakpoints-panel
Studio Release Notes	https://docs.uipath.com/studio/standalone/2022.4/user-guide/release-notes-2022-4-1#performance-and-usability

Test Activities

With Test Activities, you can run tests on specific activities, input values for variables and arguments, monitor their values, and observe execution logs.

Executing specific sections, verifying outputs, creating structured test cases, and validating the behavior of activities are all important steps in the testing and debugging process of automation workflows.

These activities help you identify any issues or errors, ensure that the automation logic is functioning correctly, and improve the overall reliability and accuracy of your automation solutions.

Test Activities – Run To, Run From

To access the Run to this activity and Run from this activity actions, we need to **right-click an activity in the Designer panel and open the context menu**.

Run to this activity will debug the project and stop before the selected activity is executed. We can use this action to check the first part of our workflow.

Run from this activity will start debugging from the selected activity. By using this action, we can skip the first part of our workflow and set values for variables for debugging the second part.

UiPath Academy

Chapter 6:

Debbuging in Studio

Run From This Activity and Run to This Activity

Test Activities – Test Activity

Test Activity

To test individual activities or containers in a workflow, we use the Test Activity action from the context menu of the activity in the Designer panel.

Right click on our new sequence to open the context menu and select the Test Activity option.

Using the Test Activity option, we can test the currently selected activity.
It is important to note that the Test Activity option is **not available in Debug mode**.

We enter Debug mode when Test Activity is selected with execution paused.
The Locals Panel appears and we can modify variables.
We can then “Continue” or “Step into”

Test Activities – Create Test Bench

Create Test Bench

To test workflow ideas, we can access the Create Test Bench action from the Activities panel.

This action lets us create automation building blocks, which can then be tested and added to the final workflow.

Go to the **Activities panel**, right click the activity we want to get started with and click **Create Test Bench**.
A separate workflow file is created
However, it is not visible in the Project panel. If we close this file without saving it, we will lose the test workflow.

We can then use the Test Bench to test code that is not part of our project.

UiPath Academy

Chapter 6:

Debbuging in Studio

Test Activity and Create Test Bench

Test Activities Resources

Feature	Link
Test Activities	https://docs.uipath.com/studio/standalone/2023.4/user-guide/test-activities

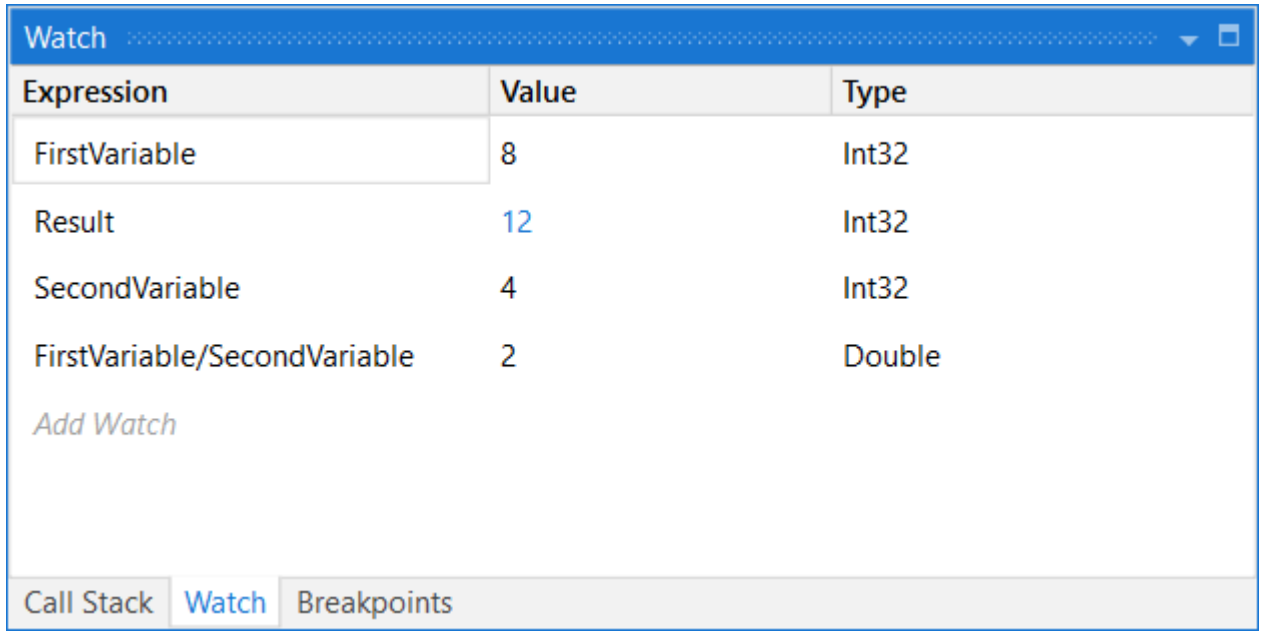
The Panels:
Watch, Immediate, and Call Stack, Locals

The panels work together to help you in localizing errors and understanding their root causes.

- **The Watch Panel** helps in identifying incorrect variable values or unexpected changes
- **The Call Stack Panel** helps in identifying the specific activities or functions where errors occur
- **The Immediate Panel** enables you to execute custom expressions and evaluate variables on-the-fly without modifying the workflow
- **The Locals Panel** displays properties or activities and user-defined variables and arguments

The Watch Panel

The Watch panel is **only visible during debugging**.
It can be set to display the values of variables or arguments & values of user-defined expressions that are in scope.
These values are updated after each activity execution while debugging.



Variables or arguments can be added to the **Watch** panel in the following ways:

- In the **Watch** panel, click the **Add Watch** field and type the name of the variable or argument;
- In the **Locals** panel, right-click a variable or argument and select **Add to Watch**;
- In the Variables or Arguments panel, right-click a variable or argument and select **Add Watch**.

To copy a value from the **Watch** panel, select the value, right-click and select **Copy**.

To remove one or more entries from the panel, right-click and select **Delete** or **Clear All**.

UiPath Academy

Chapter 6:

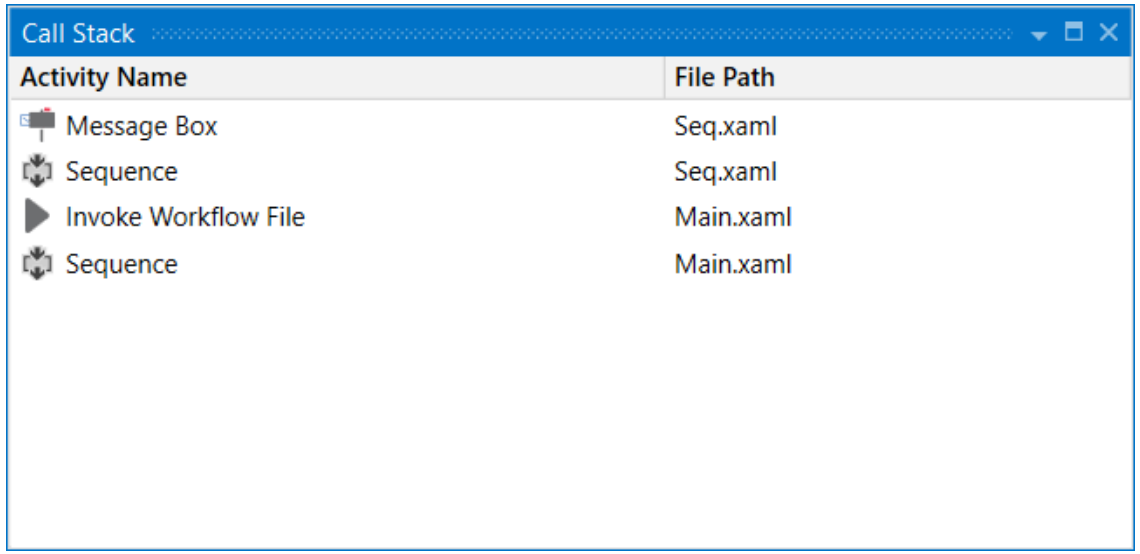
Debbuging in Studio

The Watch Panel

The Call Stack Panel

The Call Stack panel displays the next activity to be executed and its parent containers when the **project is paused in debugging**.

The panel is displayed during execution in debug mode and it gets populated after using Step Into, Break, Slow Step, or after the execution was paused because an error or a breakpoint was encountered.



Double-clicking an item in the Call Stack panel, focuses and highlights the selected activity in the Designer panel.

If during debugging, an activity throws an exception, it is marked in the Call Stack panel and the activity is highlighted in red.

UiPath Academy

Chapter 6:

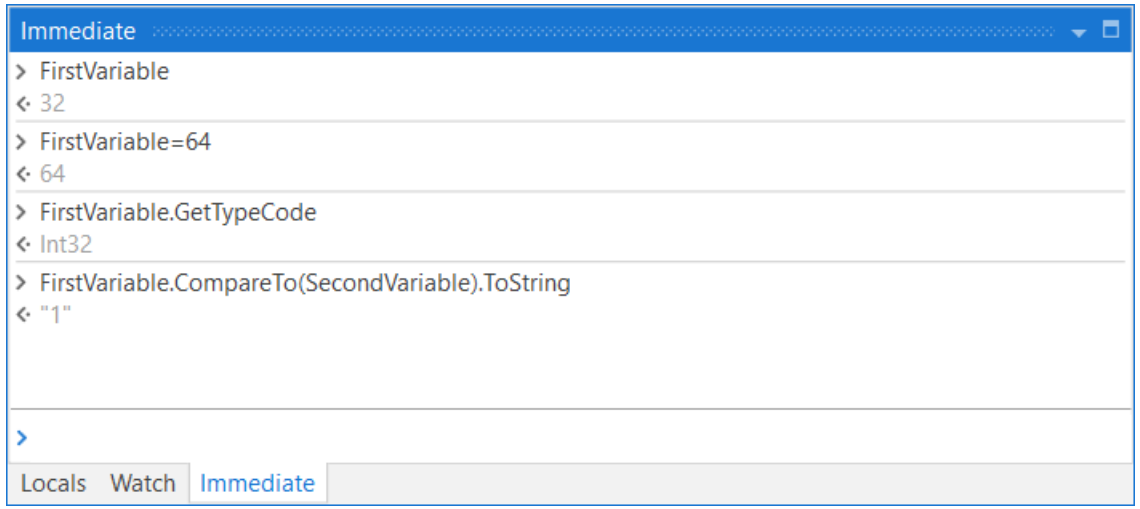
Debbuging in Studio

The Call Stack Panel

The Immediate Panel

The Immediate panel is **only visible during debugging**, and it can be used for inspecting data available at a certain point during debugging.

It can evaluate variables, arguments, or statements. To do so, simply type the variable or argument name in the Immediate window and press Enter.



Please take into consideration that the guidelines for calling a function apply to the Immediate panel as well, and parentheses should be used.

If you have a List<string> variable, it is recommended to use parentheses to view object-specific methods in the Intellisense window. For example, use Names.First().ToUpper instead of Names.First.ToUpper to capitalize the first element in a list of names.

The Immediate panel keeps the history of previously evaluated statements, and they can be removed using the Clear All context menu option.

To remove a single line from the panel, select the text and press the Space key. When clicking inside a line and starting to type, the text is added to the input field.

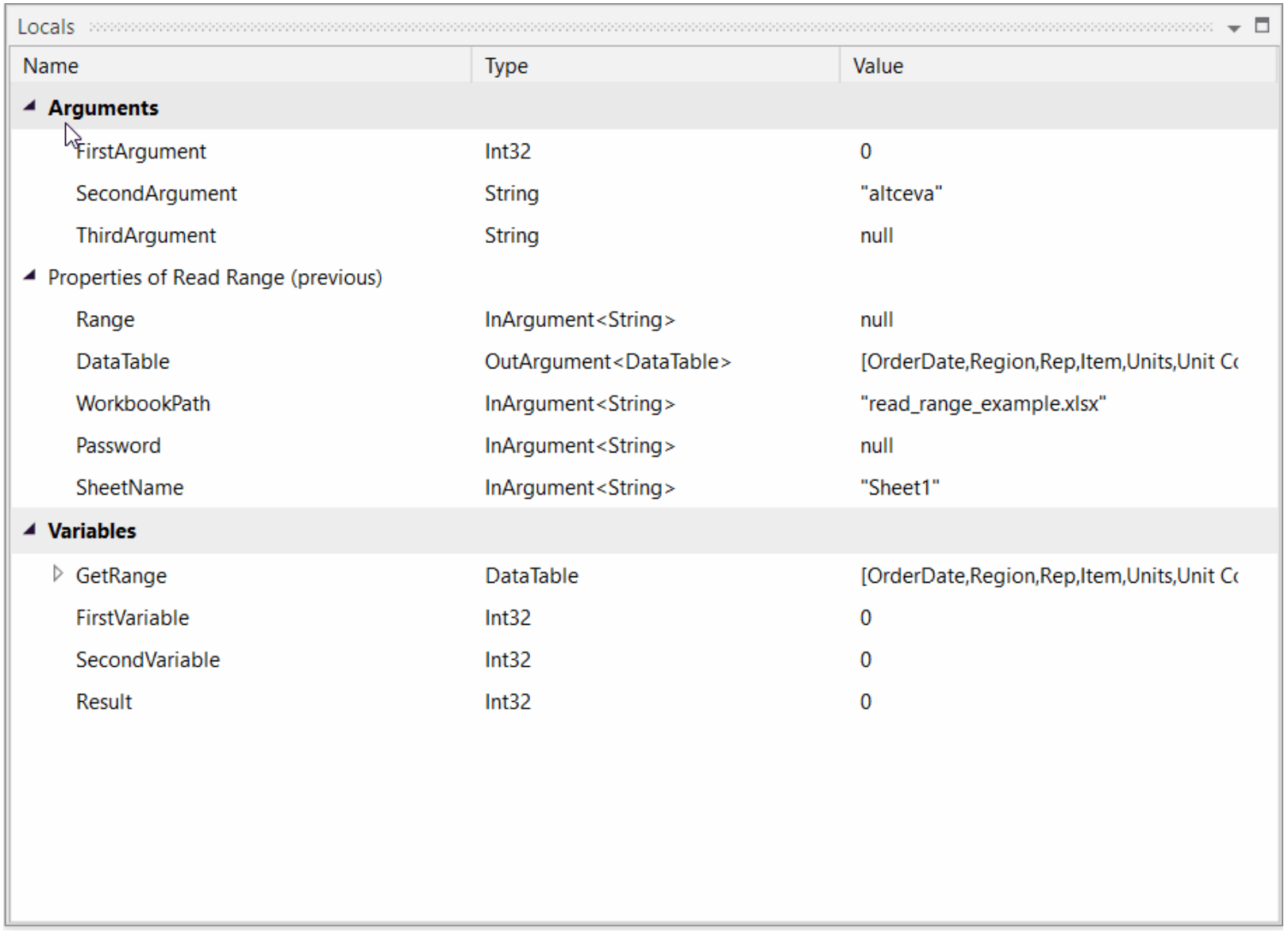
UiPath Academy

Chapter 6:

Debbuging in Studio

The Immediate Panel

The Locals Panel



The Locals panel displays properties or activities and user-defined variables and arguments.
The panel **is only visible while debugging**

The panel shows:

- **Exceptions** - the description and type of the exception
- **Arguments**
- **Variables**
- **Properties of previously executed activity** - only input and output properties are displayed
- **Properties of current activity**

The Locals Panel

The panel is **only visible while debugging**. Right-click an argument, variable or property of the currently executing activity to add it to the Watch panel and monitor its execution throughout the debugging process.

The Arguments, Properties, and Variables categories can be compressed or expanded. The same is available for complex objects, which are displayed in a tabular way.

When debugging is paused:
You can modify the properties of the current activity and the values of variables and arguments by hovering over their Value field, clicking the button next to them, and then making edits in the Local Value window.

You can inspect the values of other items in the Locals panel in detail by hovering over the Value field and clicking the button to open the Local Value window.

Note: The edits made in the Local Value window are not saved in the file. After debugging is finished, the old values are still present in the Designer.

UiPath Academy

Chapter 6:

Debbuging in Studio

The Locals Panel

Panels Resources

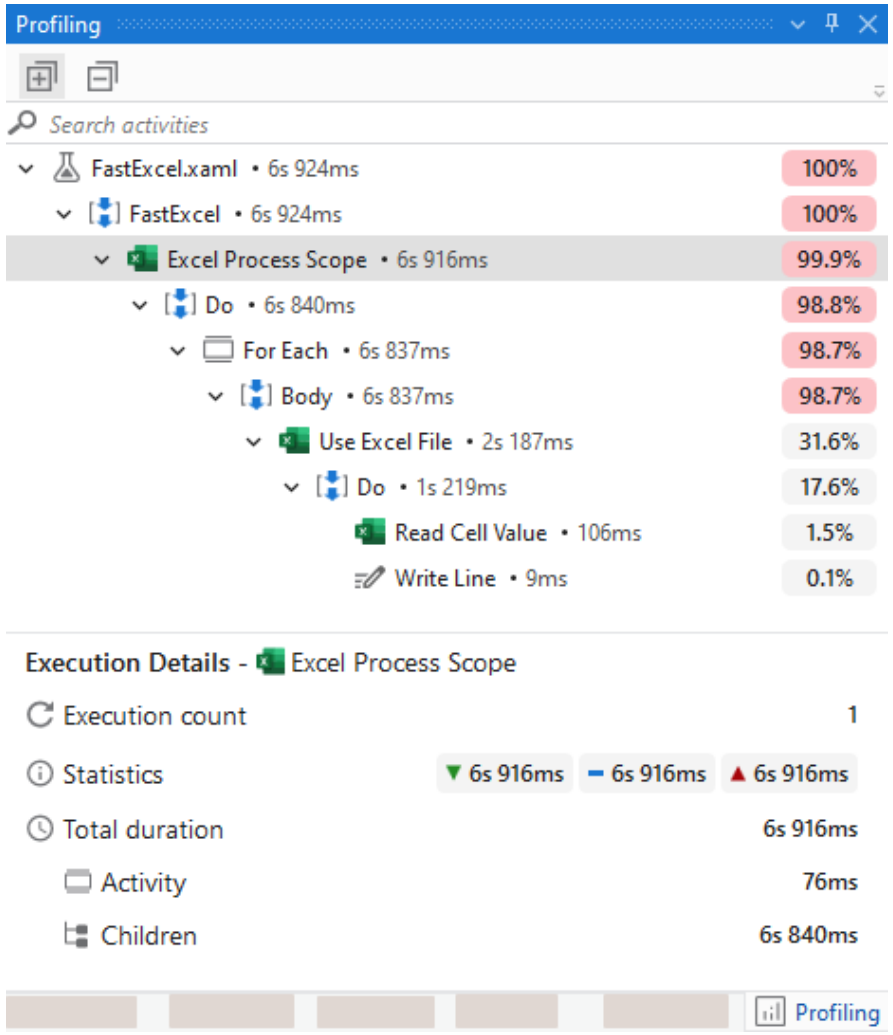
Debugging Panel	Link
The Locals Panel	https://docs.uipath.com/studio/standalone/2023.4/user-guide/the-locals-panel
The Watch Panel	https://docs.uipath.com/studio/standalone/2023.4/user-guide/the-watch-panel
The Immediate Panel	https://docs.uipath.com/studio/standalone/2023.4/user-guide/the-immediate-panel
The Call Stack Panel	https://docs.uipath.com/studio/standalone/2023.4/user-guide/the-call-stack-panel

Profile Execution

By leveraging the Profile Execution feature, you can analyze the execution time of each activity and gain insights into potential bottlenecks, allowing for optimization and improved workflow performance.

Profile Execution

Use Profile Execution to identify performance issues in workflow executions.



When you run or debug a workflow, Profile Execution provides a performance analysis of all the operations, showing you a cumulative percentage of the execution time of each activity.

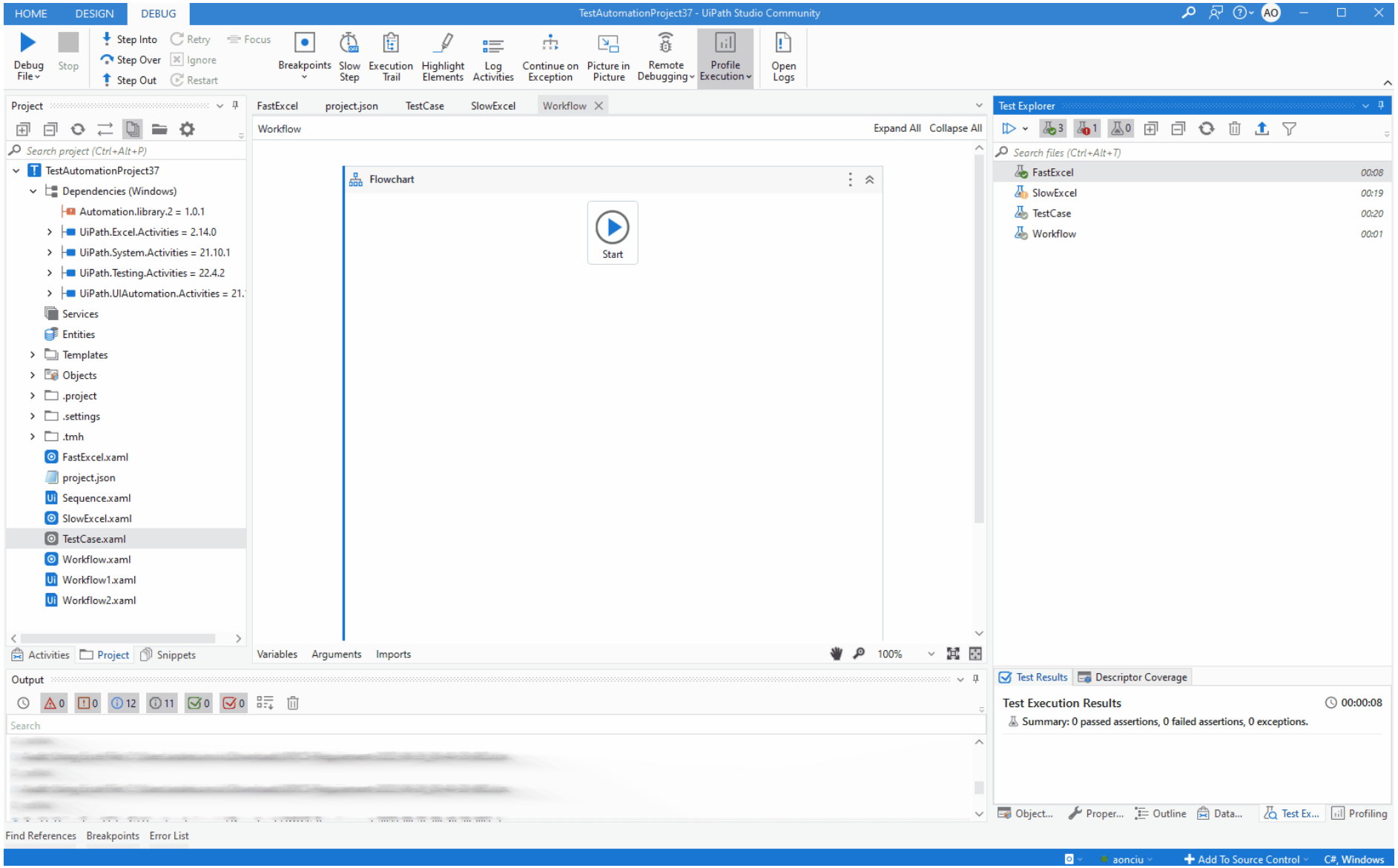
To profile an execution, run a file or debug the file, then go to the Debug ribbon tab.

Make sure that your automation works correctly without slow performing workflows. If you identify potential flow issues, you can review workflows that take longer to be completed. Use the context menu for profiling to take actions.

The Execution Details at the bottom of the Profiling tab shows the number of executions and descriptive statistics such as the average duration and the minimum and maximum duration values.

- Profiling data generated during debugging might be different from data generated during production (running the file)
- The data is stored for each session in C:\Users\username\AppData\Local\UiPath\ProfiledRuns (see Import previous profiling sessions)

Analyzing profiling results



To profile an execution, run a file or debug the file, then go to the Debug ribbon tab.

Make sure that your automation works correctly without slow performing workflows. If you identify potential flow issues, you can review workflows that take longer to be completed. Use the context menu for profiling to take actions.

The Execution Details at the bottom of the Profiling tab shows the number of executions and descriptive statistics such as the average duration and the minimum and maximum duration values.

Import profiling session

You can import profiling sessions to examine previous runs. **Focus is disabled on imported profiling sessions**.

Context menu for profiling

Action	Description
Open	Right-click the parent file in the Profiling tab and select **Open** to jump to the selected workflow.
Focus	Right-click an activity and select **Focus** to center the **Designer** panel view on the selected activity.
Search	Use the search function to look for specific activities.
Expand/Collapse All	Use **Expand All** and **Collapse All** to bring to view or collapse all activities.

UiPath Academy

Chapter 6:

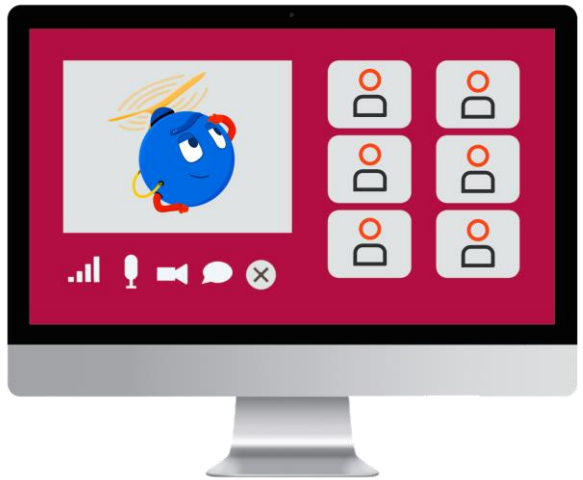
Debbuging in Studio

Profile Execution

Profile Execution Resources

Feature	Link
Profile Execution	https://docs.uipath.com/studio/standalone/2023.4/user-guide/profile-execution

Classroom Exercise



Problem statement

One of your team members has developed a workflow using UiPath Studio, but it is encountering errors during execution.

Your task is to effectively debug the workflow, identify the underlying issues, and apply the concepts and features covered in this course to rectify the problems.

This process should:

Read data from an Excel file.
Access the RPA Challenge website.
Input read data into the form fields on the RPA Challenge website.

Note: The fields will change position on the screen after every submission.

You can download the workflow provided below. Happy debugging!