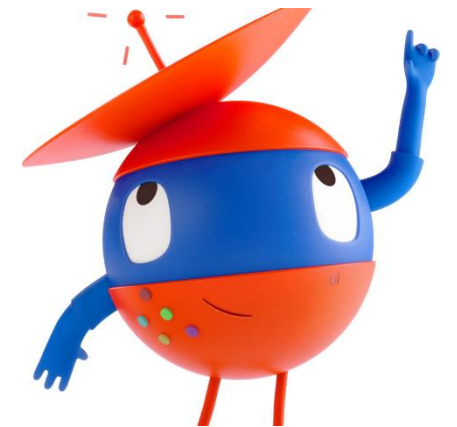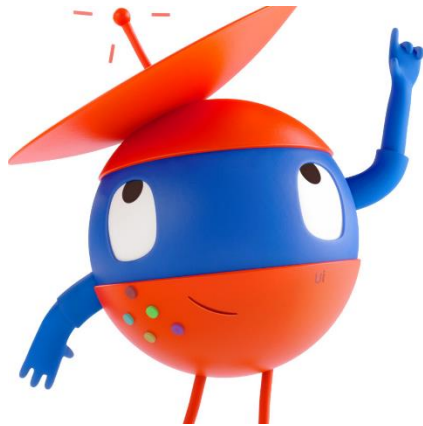# Introduction to Robotics Process Automation (RPA)

Subject Code: CSE552

Department of Computer Science and Engineering

October-December 2021

UNIT III

DATA MANIPULATION:
- Data Manipulation and Its Importance,
- String Manipulations,
- Data Table Manipulations,
- Collection, Its Types and Manipulations.

UI AUTOMATION & SELECTORS:
- UI interactions,
- Input actions and Input methods, Containers,
- Recording & its types,
- Selectors, Types of Selectors- Full and Partial,
- Containers and Partial Selectors, Dynamic Selectors
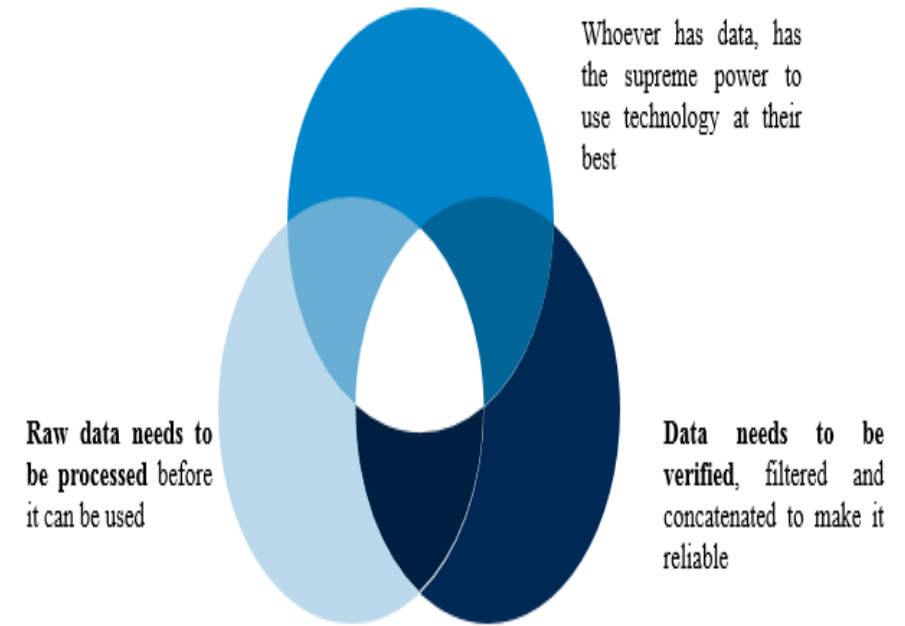
Data Manipulation

- Data Manipulation is defined as the process of altering the original behavior by applying mathematical operation of data structure like variable, data table, retrieval, filtering, verification and concatenation.

Retrieving Data

Inserting data

Examples of Data Manipulation

Selecting data

Deleting data
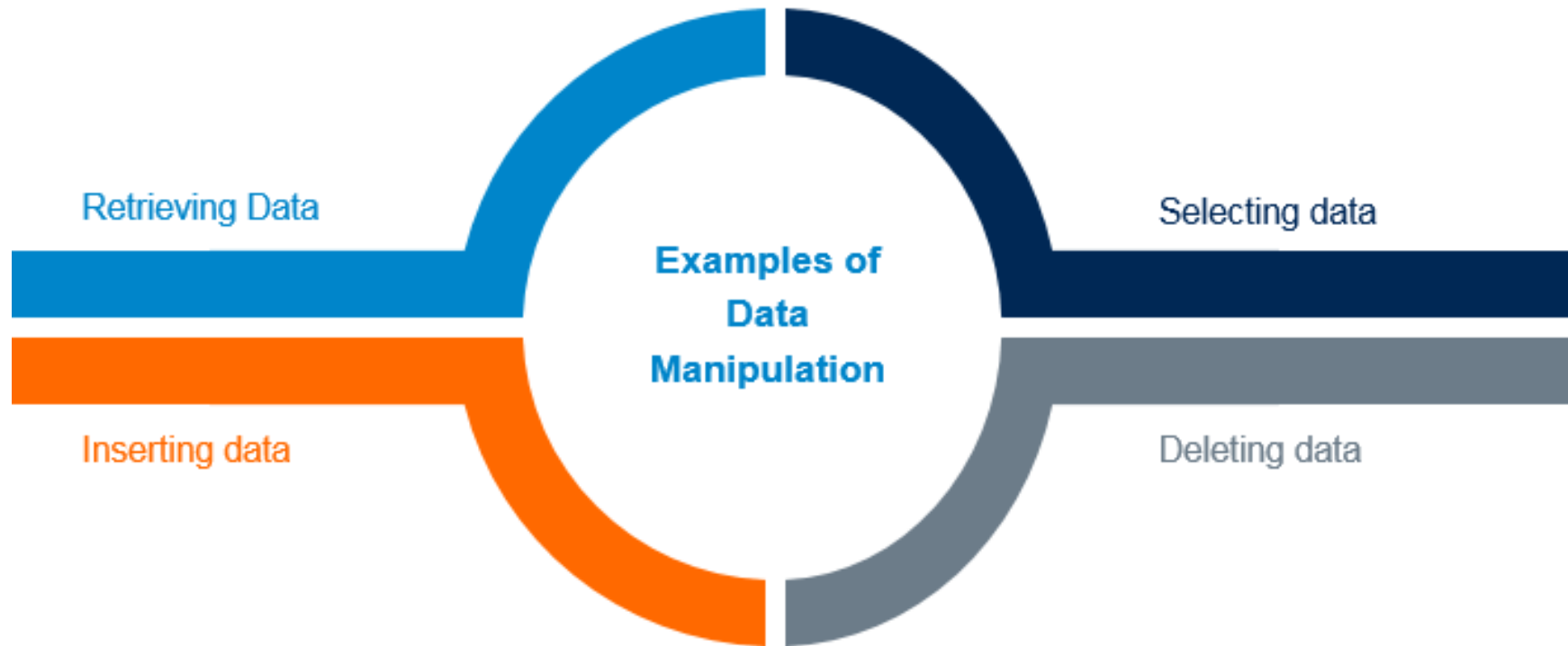
Importance of Data Manipulation

Data is defined as collection of information which can be stored and processed.

- Nowadays, **data is the top most priority for organizations**. Whoever has data, has the supreme power to use technology at their best.

- Organizations are in **deep competition to utilize and maximize their efficiency by having data.**

- However, having data is not enough as it should mean something to the company. Meaningless data could prove to be unreliable and impossible to analyze. Having raw data is not enough; it needs to be processed accordingly before it can be used.

- For example, **data collected from the user in the form of web forms** could prove to be unreliable and wrong in some cases. This type of data needs to be verified against other data of similar pattern to verify.

- Consider where **a database that collects all the links opened by a user within a year;** it's almost impossible to analyze it for the purpose of predicting the buying behavior of that user;

Whoever has data, has the supreme power to use technology at their best

Raw data needs to be processed before it can be used

Data needs to be verified, filtered and concatenated to make it reliable

Basic Operations of Data Manipulation

Most of the data manipulation languages (again, SQL is a good example) share 4 basic commands:

Select, Update, Insert (into) and Delete (from). These commands do not cover all the data processing techniques. But, by using conditions and configurations, complex manipulations can be performed.

A basic outline of these commands are mentioned below:
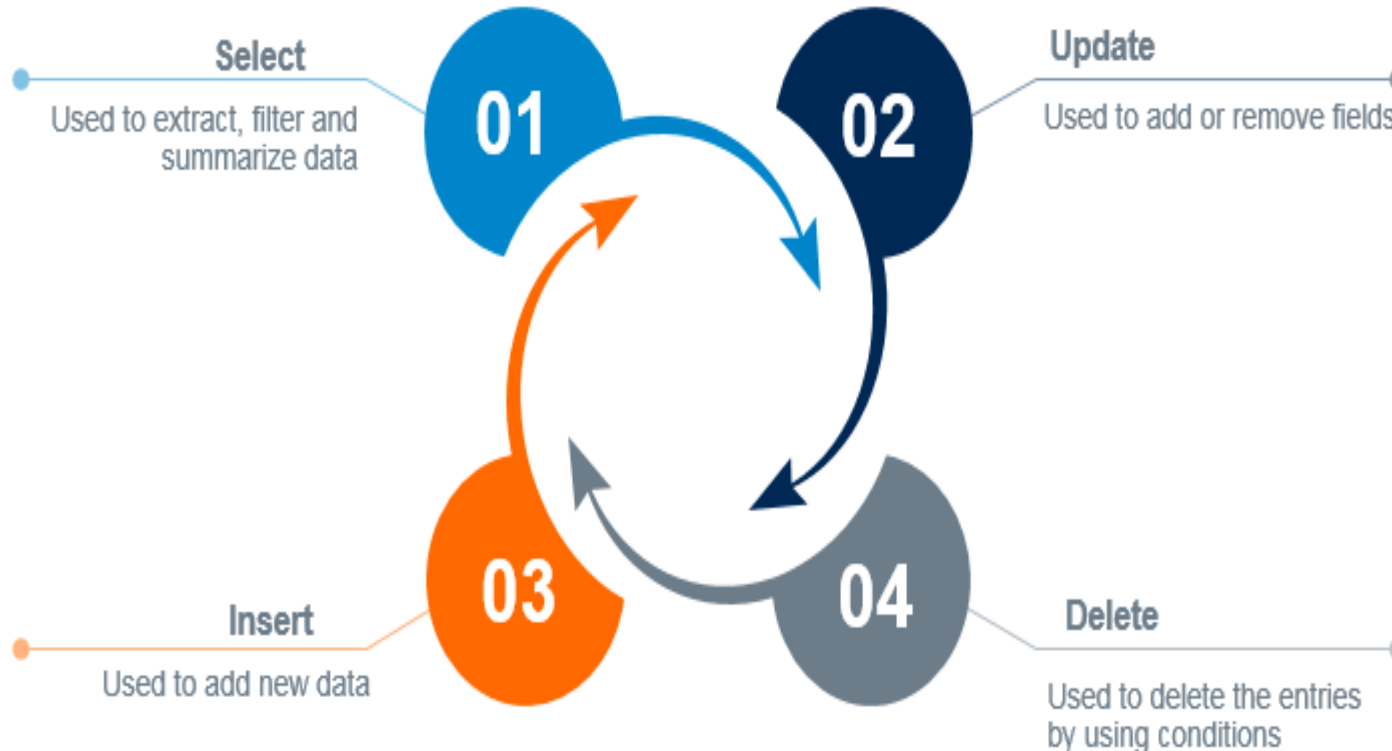
- **Select :** This is used to extract, filter and summarize data.
- **Update :** This is used to add new columns(fields) or remove (update the existing one). This is also used to merge databases.
- **Insert :** This is used to add new data (rows).
- **Delete:** This is used to delete the entries by using conditions.

Select
Used to extract, filter and summarize data

01

Update
Used to add or remove fields

02

Insert
Used to add new data

03

Delete
Used to delete the entries by using conditions

04

Types of Data Storing Variables

- **Number:** Used to hold the numerical value such as an integer or a float.

- **String:** Used to hold collection of characters.

- **Array and List:** Used to hold the array and list value.

- **Date and Time:** Used to store the data and time.

- **Boolean:** Used to store either true or false as the value.

- **Data Table:** Used to store the Row index and column name.

Data Manipulation Operations

- **Initializing:** Used to set up a variable

- **Selecting:** Used to retrieve rows selected from one or more tables

- **Inserting:** Used for inserting one or more rows into a database table with a specified table column

- **Updating**: Used to update rows

- **Deleting:** Used to delete one or more rows

Initializing   Selecting   Inserting   Updating   Deleting
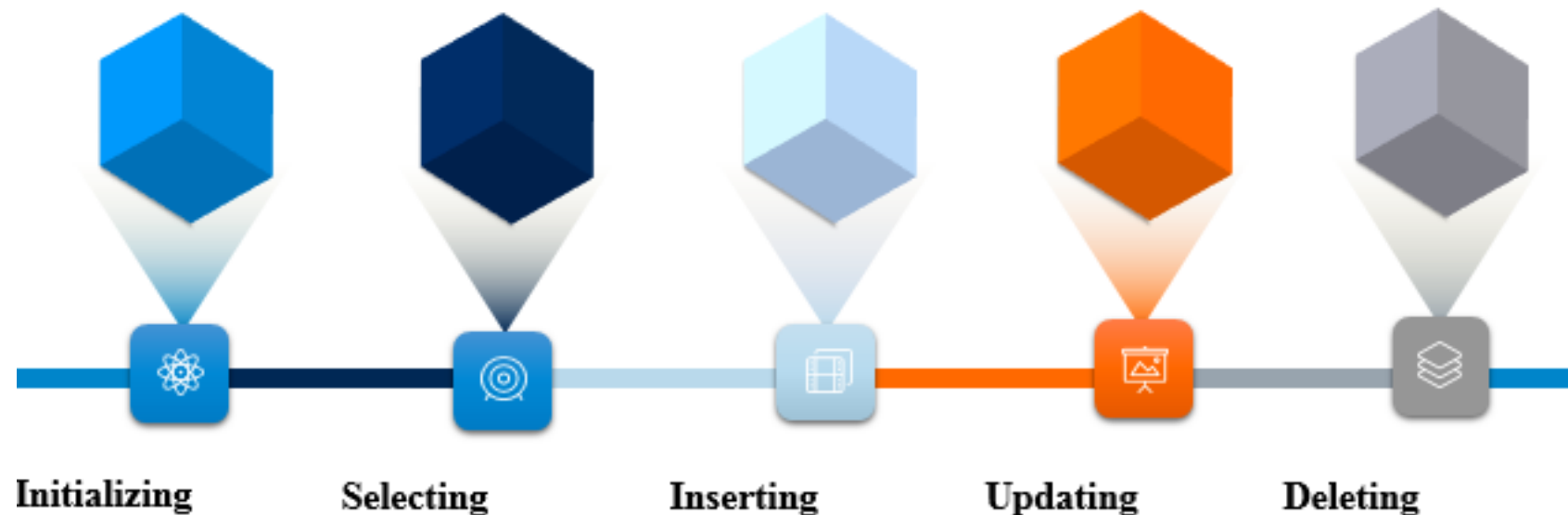
Variable Initialization

Step 1:
- Go to the Variables Panel and Click on "Create variable"
- Choose a name for the variable

Provide a name for the variable

Name	Variable type	Scope	Default
VarString1	String ▾	Sequence	*Enter a VB expression*
Create Variable			

Boolean
Int32
String
Object
Array of [T]
Browse for Types ...

Type Name: system.string

▲ <Referenced assemblies>
  ▲ mscorlib [4.0.0.0]
    ▲ System
      String
      StringComparer
      StringComparison
      StringSplitOptions
  ▲ System [4.0.0.0]
    ▲ System
      StringNormalizationExtensions

Step 3:
- Choose the scope of the variable

Choose the scope of the variable

Name		Variable type	Scope	Default
VarString1	☐	String ▾	Sequence	*Enter a VB expression*
Create Variable				

Boolean
Int32
String
Object
Array of [T]
Browse for Types ...

Step 2:
- Choose the type of the variable from the drop-down list

Choose the type of the variable

Name		Variable type	Scope	Default
VarString1	☐	String ▾	Sequence	*Enter a VB expression*
Create Variable				

Boolean
Int32
String
Object
Array of [T]
Browse for Types ...

Step 4:
- Specify a default value of the variable

Type a default value (optional)

Name		Variable type	Scope	Default
VarString1	☐	String ▾	Sequence	*Enter a VB expression*
Create Variable				

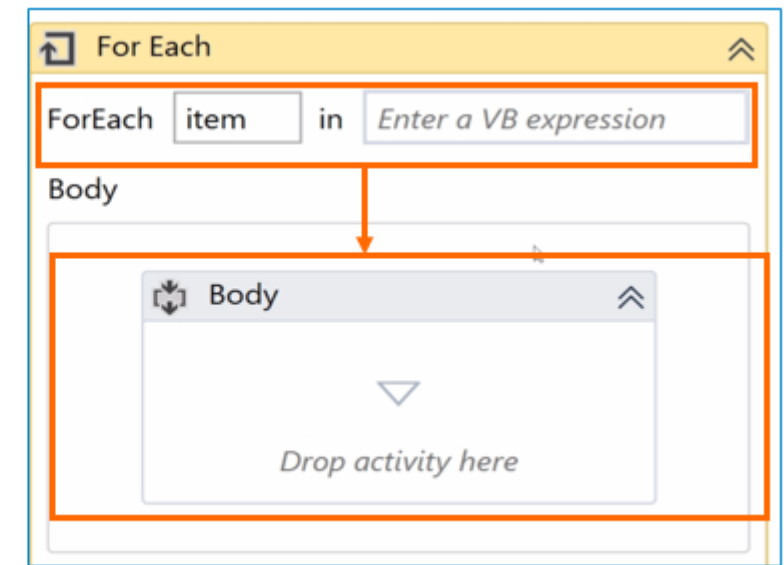Boolean
Int32
String
Object
Array of [T]
Browse for Types ...

When we are creating **Collection Variables (Arrays, Lists, Dictionaries),** the data type of the elements has to be specified.

Data Selection
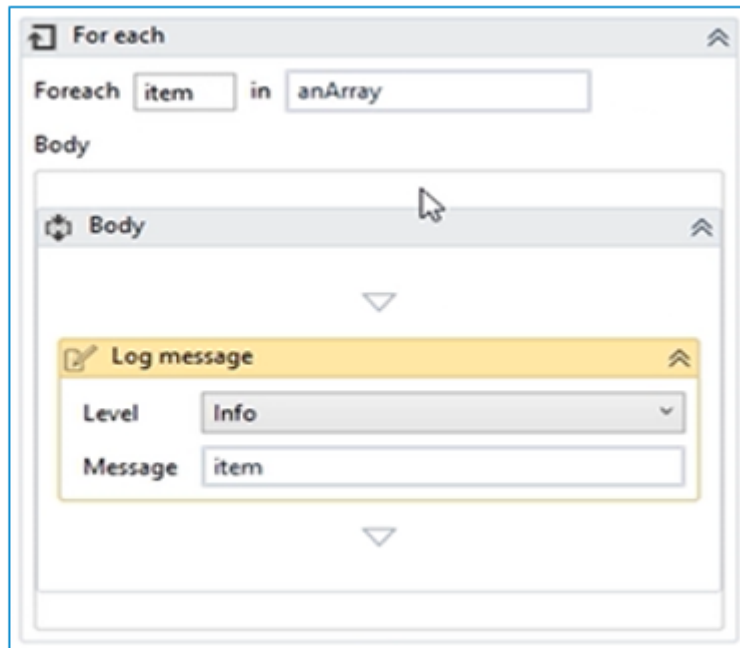
Common Selection Operations

- Looping through collection variables
- Retrieving and Referencing
- Counting
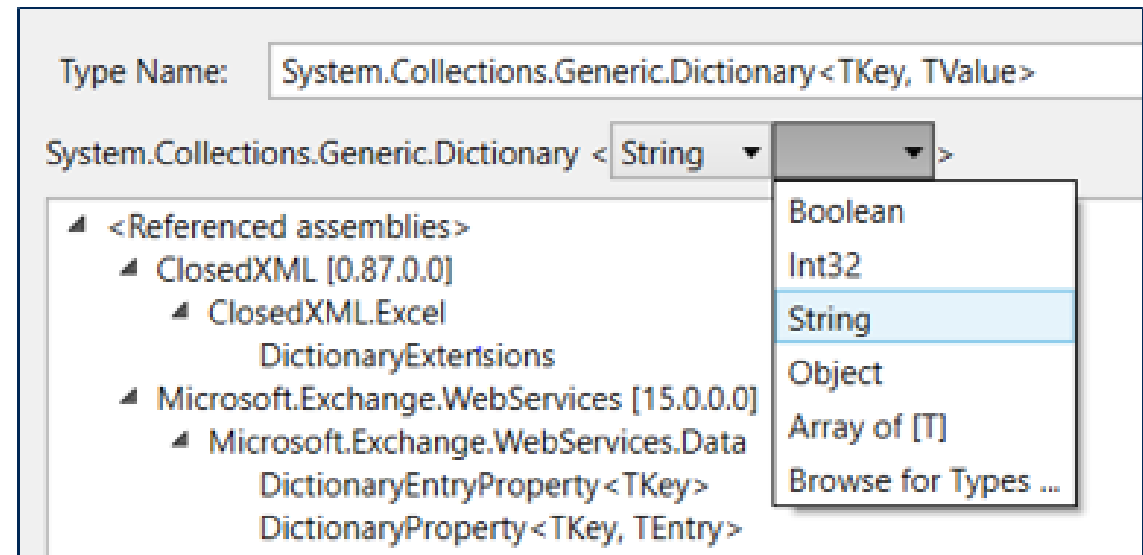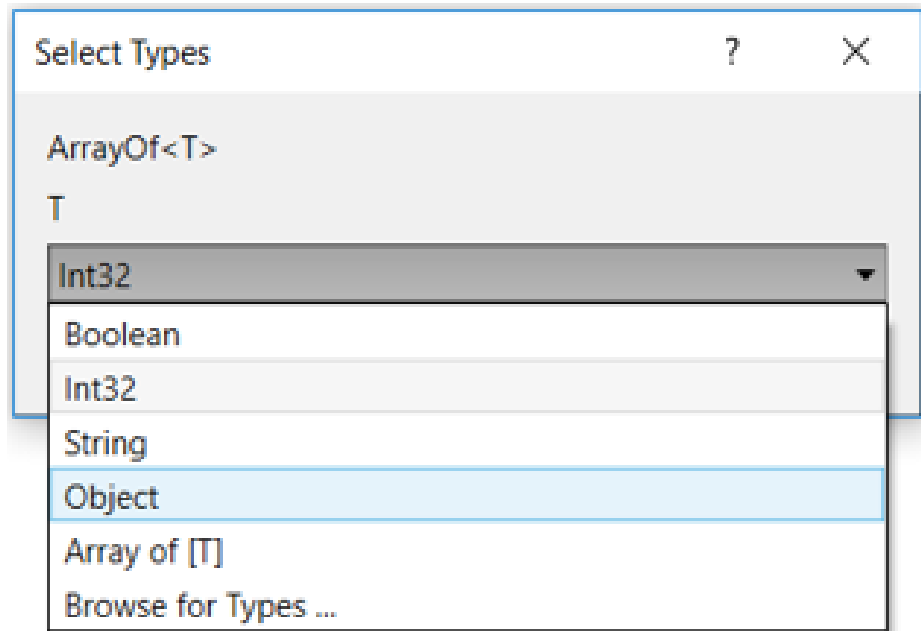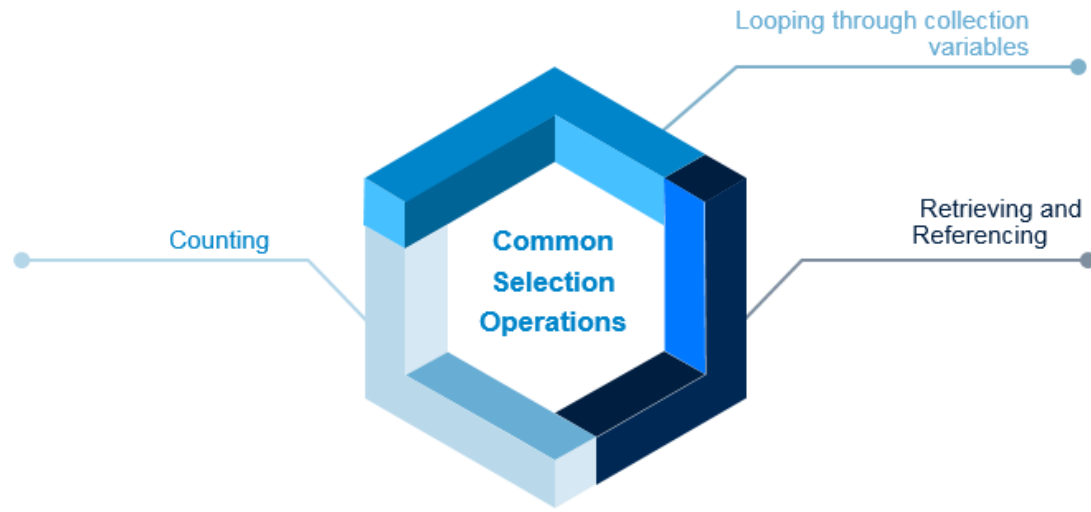
Step 1:
Drag a 'For Each' activity into the workflow

For Each

ForEach | item | in | Enter a VB expression

Body

Body

Drop activity here

Step 2:

- Go to the properties panel of the activity
- Change the 'TypeArgument' to the type of the variable you wish to iterate through

Properties

UiPath.Core.Activities.ForEach<System.Obje...

Common	
DisplayName	For Each
Misc	
Private	☐
TypeArgument	Object ▼

Boolean
Int32
String
Object
System.String[]
System.Collections.Generic.Dictionary<System.String,System.String>
Array of [T]
Browse for Types ...

Step 3:
Loop through the collection by inserting its name in the 'For Each' activity to display items

For each

Foreach | item | in | anArray

Body

Body

Log message

Level | Info ⌄

Message | item

Let's go back to the Array with the name of the Calculus 3 students: Evalyn Mineau, Altagracia Sentell, Karon Rosamond, Lucas Kowalewski, Tiny Hewlett, Sherlyn Smothers, Carley Chicoine, Charlesetta Carini, Felicia Phillips, Emogene Carrow, Andra Willard, Rachelle Alkire, Carissa Yoshimura, Sheba Holben, Earlean Rames, Kendrick Cornett, Thomasina Mohler, Elly Bottorff, Meaghan Jacquemin, and Craig Brockwell
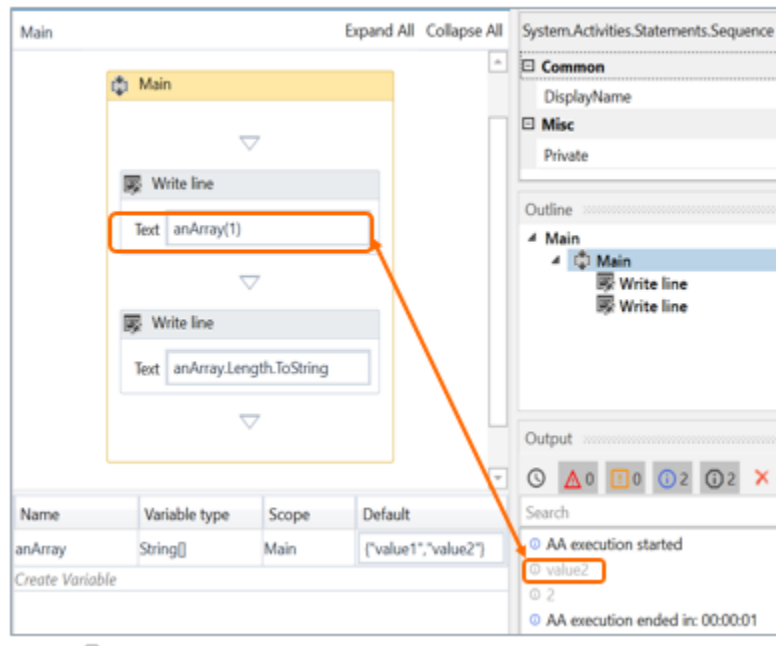
Create a simple workflow to loop through it.

Referencing an Item

Referencing an item in a collection variable by its index

VariableName(index of item as integer).toString

An item from a Dictionary variable can be retrieved by using the Item method.

Retrieve a value using the item method
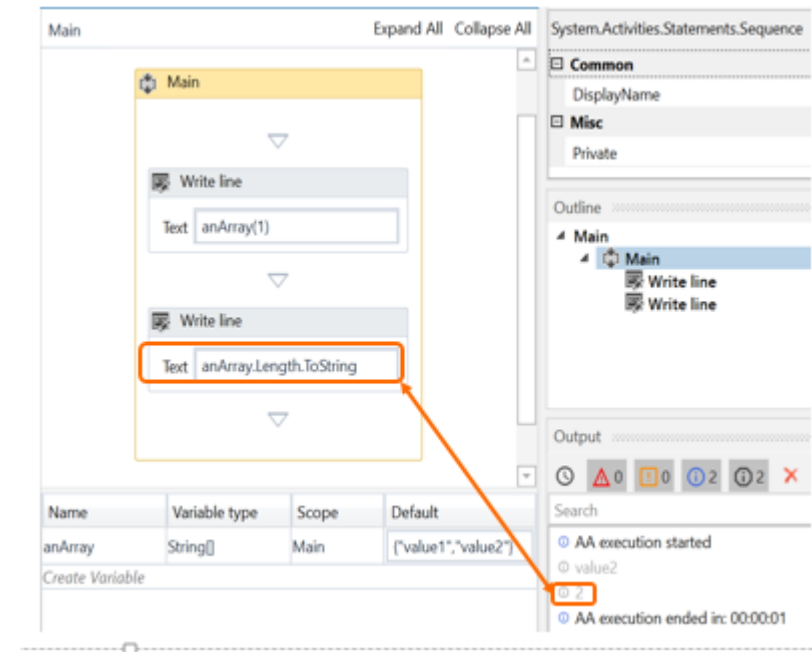**[DictionaryName].item("[key]")**
In our example:
config.Item("key2")
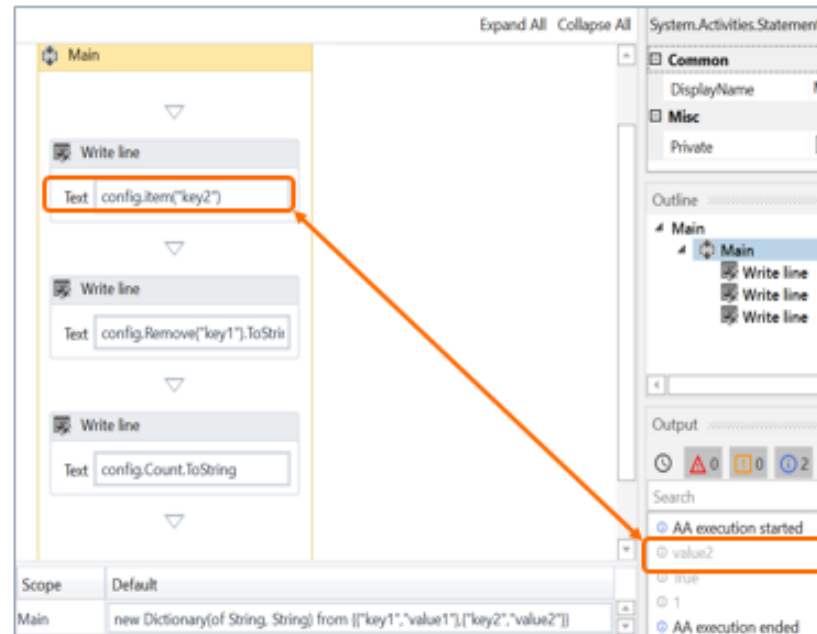
Counting an Item

Displaying the length of a collection variable through the command **'Length'**
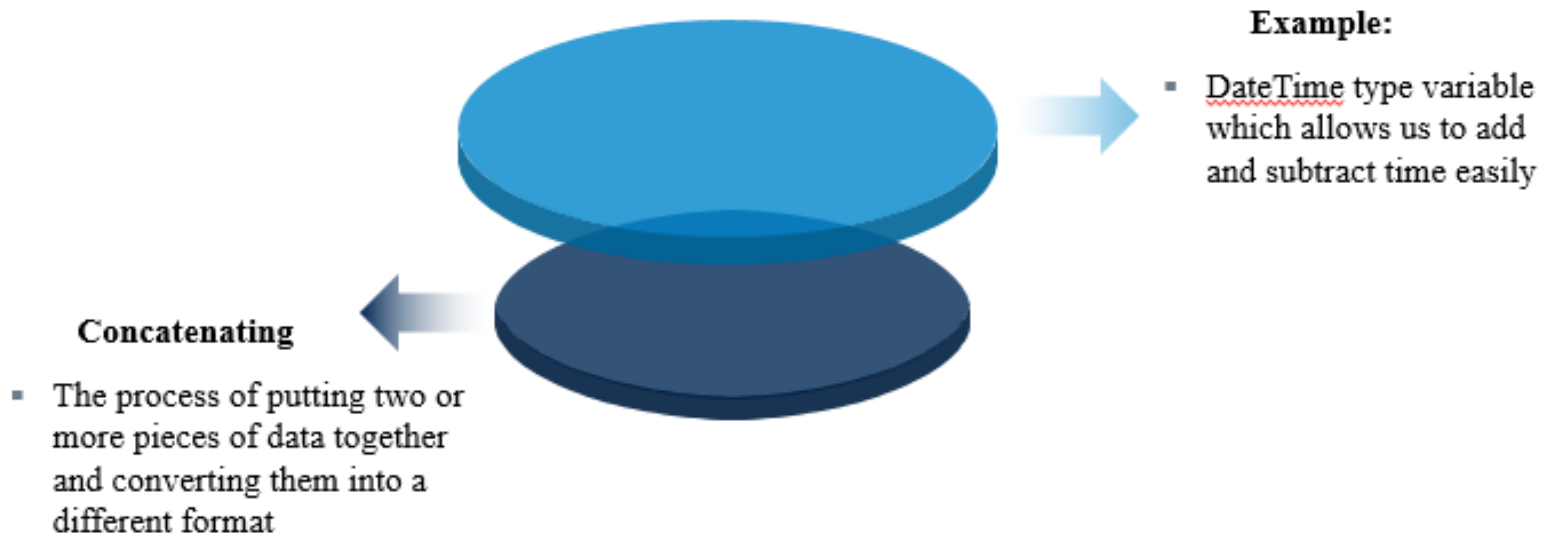
Data Inserting

Use an **Assign activity** to add a Key/Value pair to an existing dictionary.

In the To section insert **[DictionaryName]("[Key]")** and insert the value you wish to be stored in the opposite entry.

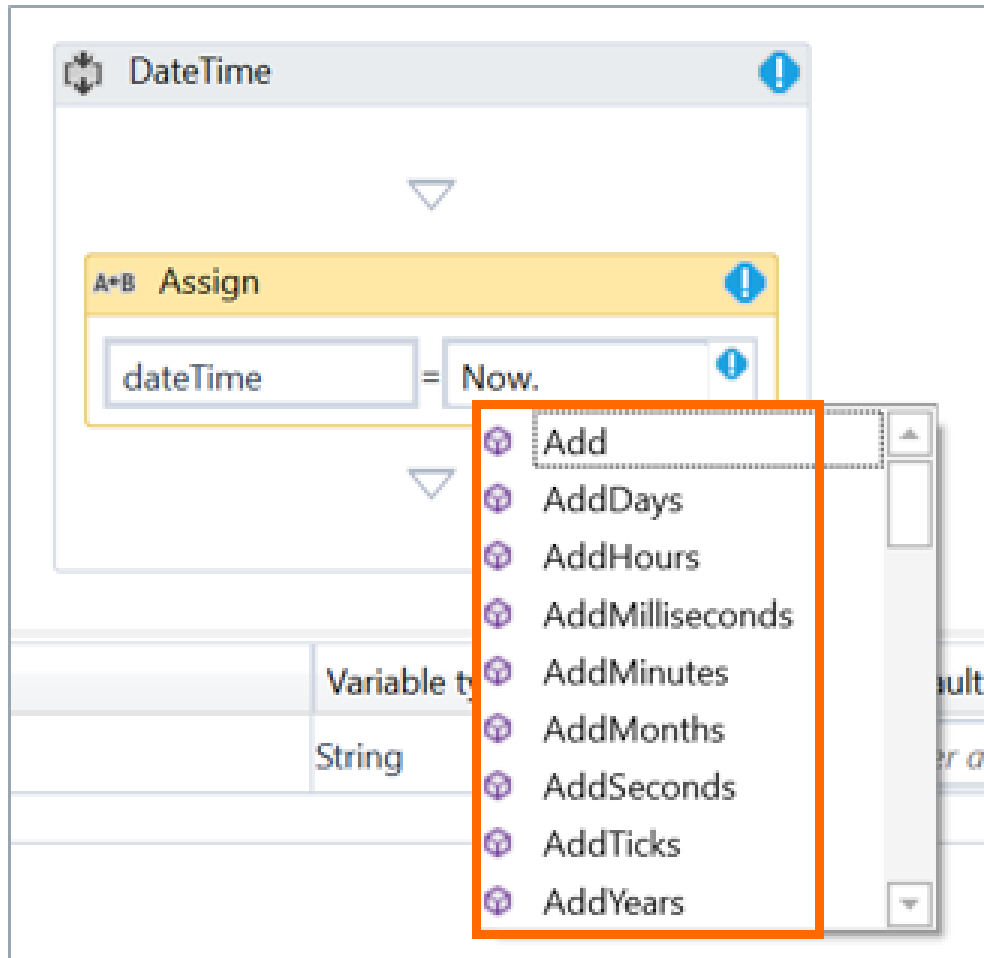| Name	Variable type	Scope	Default
config	Dictionary<String,String>	Main	new Dictionary(of String, String) from {{"key1", "value1"}, {"key2", "value2"}}

A→B Assign

config("key3") = "value"

Data Updating

Example:
- DateTime type variable which allows us to add and subtract time easily

Concatenating
- The process of putting two or more pieces of data together and converting them into a different format

Manipulating DateTime Variables

UiPath offers many **built-in methods to manipulate data**. In the illustration above, users can see that "Now." triggers a series of functions that we can use to manipulate the DateTime variable.

- It can prove to be very useful in a situation where an email needs to be processed, and we can add five minutes to the time before we process it. This way, in case of any error, we could remove the error and recover the email.
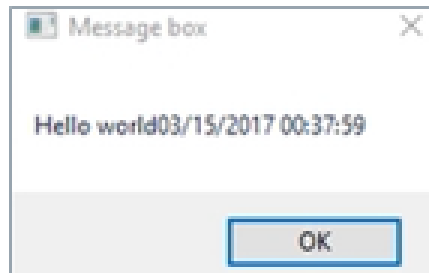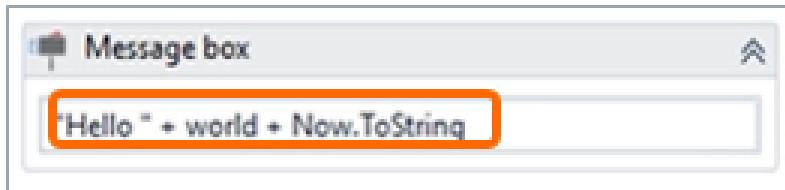
Concatenating and Converting

In UiPath, **concatenating the same type of data** is relatively simple, we can use the "+" symbol.

In UiPath, the "+" symbol is used to **concatenate** same type of data. To concatenate different data types, they are first **converted** to string type using the "ToString" method.
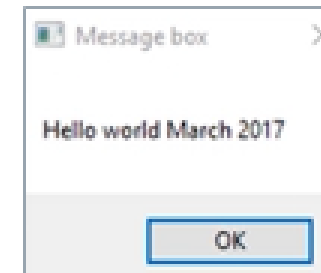
- In most cases, the best option is to convert the data into a string type. To do this, we can use the ".ToString" method.

- In the example on the right side, "Now.ToString("MMMM yyyy")" is used to specify the format it needs to be in. As we can see ("MMMM yyyy") displays the date in a different format "March 2017".

Raw DateTime

Message box ⦁

"Hello " + world + Now.ToString

Message box ✕

Hello world03/15/2017 00:37:59

OK

Formatted DateTime

Message box ⦁

"Hello " + world + Now.ToString("MMMM yyyy")

Message box ✕

Hello world March 2017

OK

Data Deleting

As a general rule, Deleting should probably be the most accessible data manipulation command.

In the example above we are using the "Remove" command to delete entries from a dictionary and then later displaying its content using the "Write-Line" command for verification that whether the deletion was successful.

- The above example uses the "Write-Line" command to remove entries from the dictionary. Once the key/value pair has removed, the activity returns a True/False output, which converts to a string.

- In the third Write Line command, we are counting and displaying the key/value pairs in the Dictionary.

The delete command might look easy to use but adding the right operations to delete only the unrequired data is a challenge. It depends more on the ability of the one making the automation than the software.

Removing a Key/Value pair
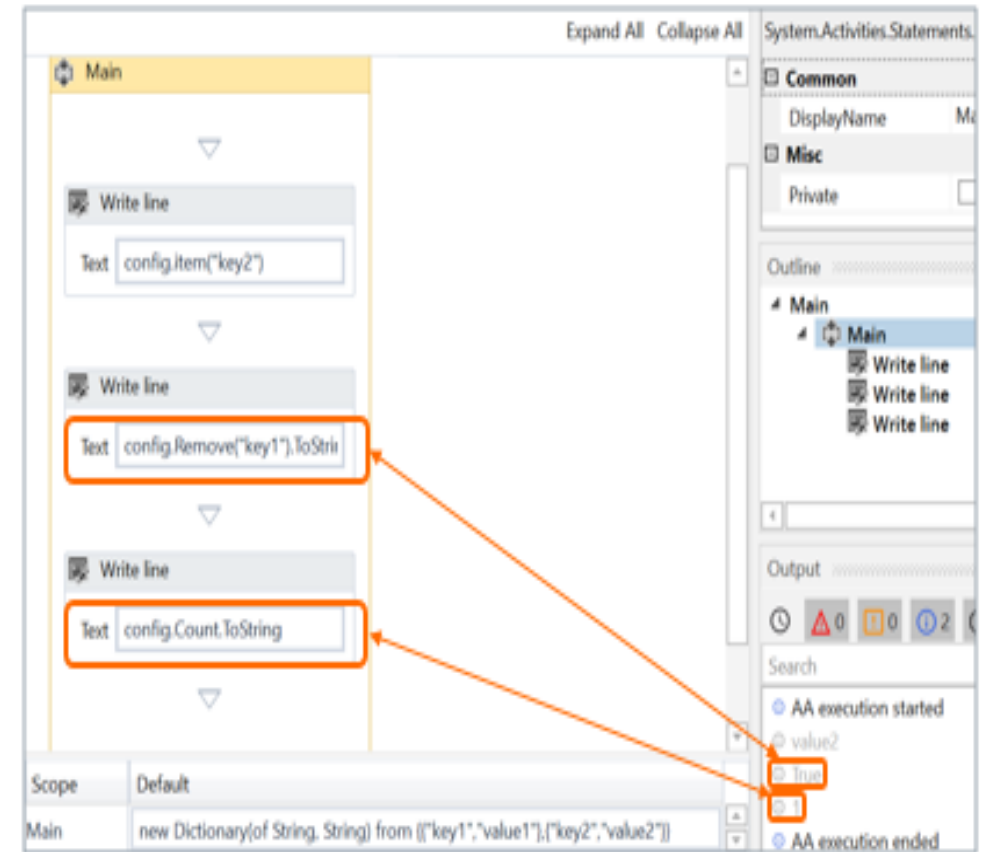
String Manipulations

Text manipulation refers to the operations performed on text. Since text is stored in a string or collection of string type variables, the text manipulation operations are called string methods.

- When a string variable is used in an activity, all available string methods can be accessed by putting a "." after the string variable.
- These methods are displayed in a drop-down list and can be selected after navigating with the arrow keys.
- If you, click outside the drop-down menu, the methods will disappear.
- To bring them back, press Ctrl + Space, it positions the cursor after the dot.
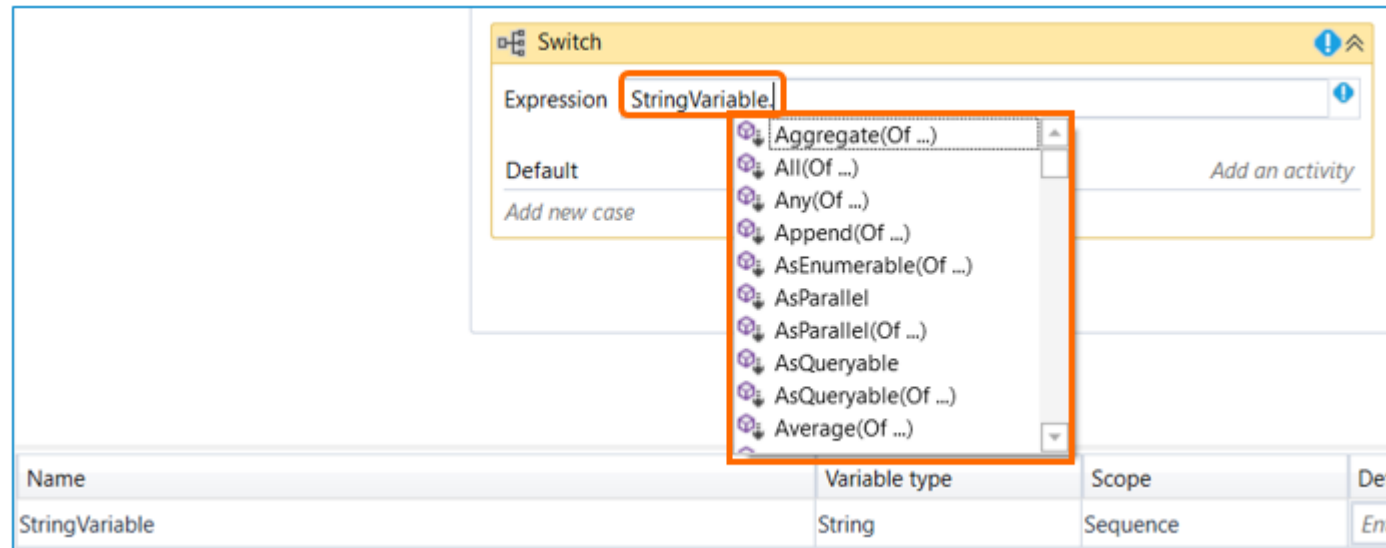
Main String Methods

Method	Syntax
Contains	Function String.Contains(value As String) As Boolean
EndsWith	Function String.StartsWith(value As String, ignoreCase As Boolean, culture As CultureInfo) As Boolean
StartsWith	Function String.EndsWith(value As String, ignoreCase As Boolean, culture As CultureInfo) As Boolean
Format	Function String.Format(provider As IFormatProvider, format As String, arg0 As Object, arg1 As Object, arg2 As Object) As String
Replace	Function String.Replace(oldChar As Char, newChar As Char) As String
Split	Function String.Split(separator As Char(), count As Integer, options As StringSplitOptions) As String()
Substring	Function String.Substring(startIndex As Integer, length As Integer) As String
ToLower	Function String.ToLower(culture As CultureInfo) As String
ToUpper	Function String.ToUpper(culture As CultureInfo) As String
Trim	Function String.Trim(ParamArray trimChars As Char()) As String

String.Contains

"String.Contains" is a method that is used to verify if a string contains a particular string of characters.

- **This method returns a Boolean value: True or False.** This method can be used in a loop or if statement.
- We can know whether a statement was successful or not by using the output of this command.

- The commands "StartsWith" and "EndsWith" are very similar to the "Contains" command. The only variation is that rather than searching the whole string, we use "StartsWith" to search the Beginning of the string and "EndsWith" to search the end of the string. These methods also return a Boolean value.
  - **StartsWith** checks if the string initiates with a certain substring. It could be useful in the case where we need to automatically forward an email that starts with "urgent."
  - **EndsWith** checks if the string finishes with the indicated substring.
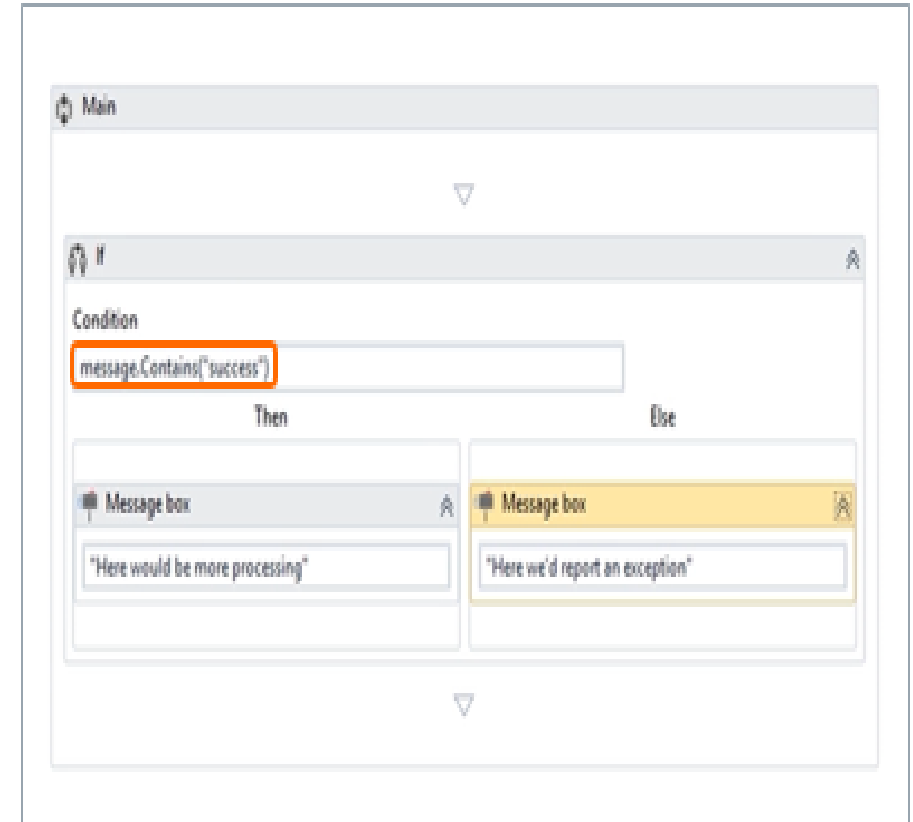
String.Format

The "String.Format" operation can be **used to convert an entire expression into a string.** This not only reduces complexity, but, increases readability.

- Consider a scenario where we need to send an email from a word document and the document contains not only strings but, date and time, number type variables also. The entire document then needs to be converted into the string to avoid any errors.

- In the example above, we have a simple conversion of a message containing text ("Hello"), the current date and time (We use the "Now" parameter to do this) and the variable world (which is a string with the value "world"). It shows the outcome in the message box.
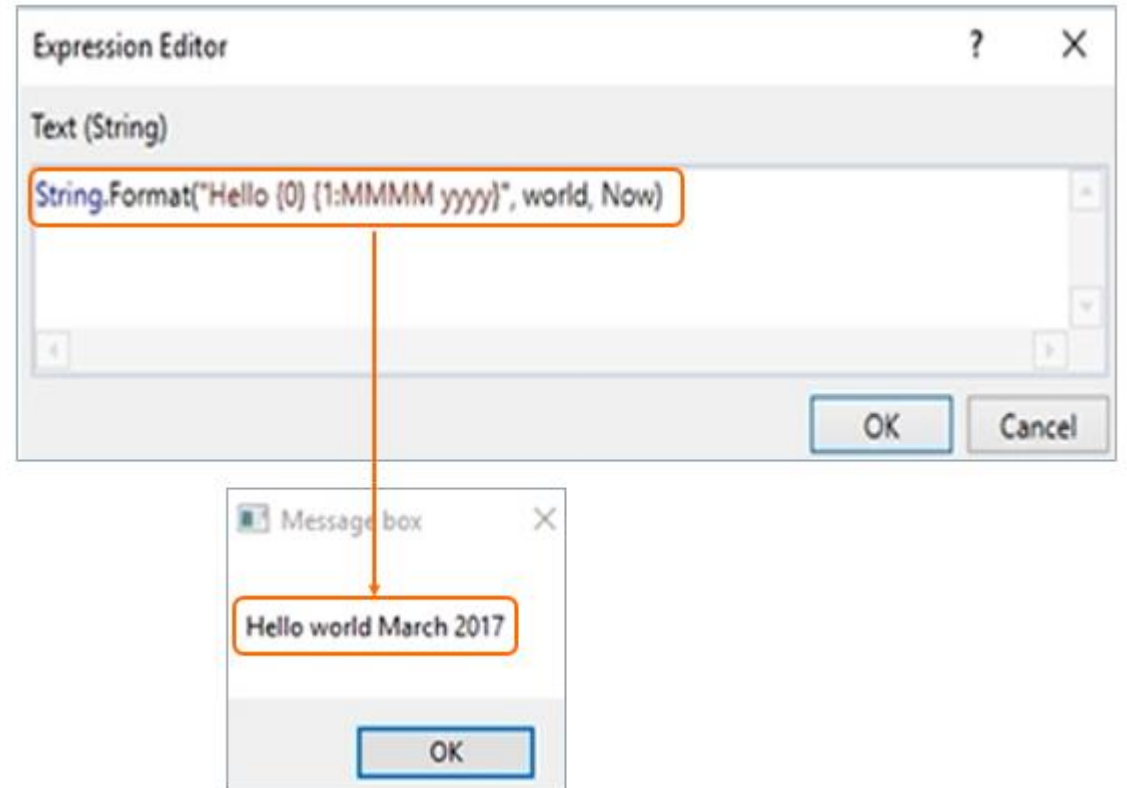
String.Replace

The String.Replace method is used to identify a sequence of characters (string) in a text and replace it with a given string.

Syntax   ⊕ Function String.Replace(**oldValue As String**, newValue **As String) As String**

According to above description, we can see the syntax of this method. The variable **'newValue'** **replaces the variable oldValue "As String".** You can see an example on the next slide.

Example

The variable "name" stores the name that the user inputs.

Name		Variable type	Scope	Default
name		String	Sequence	*Enter a VB expression*
message		String	Sequence	"Hello friend! What's your name?"

The variable "message" has an initial value: "Hello friend! What's your name?"

Step 1:

- Greet the user with the initial value of the "message" variable.
- Get the name that the user inputs.
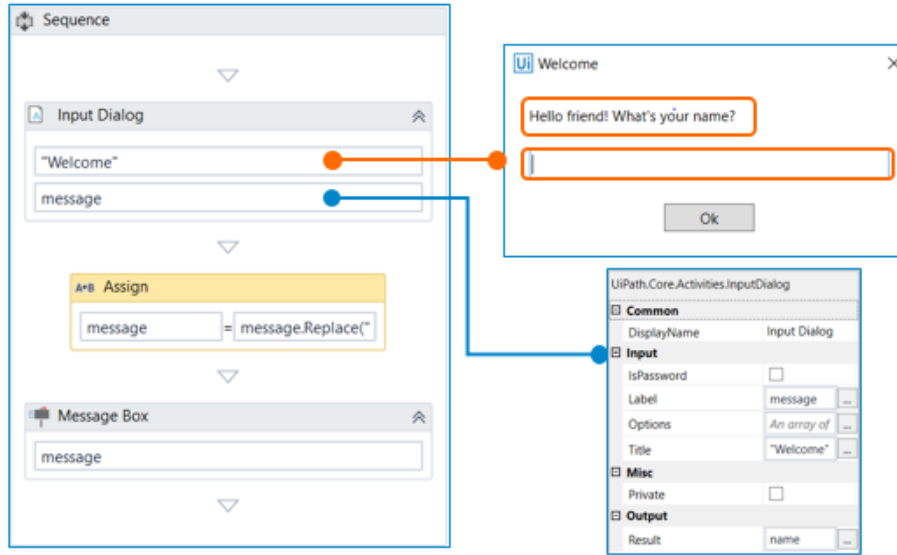
Step 2:

- Perform the "String Replace" method inside an Assign activity.
- Use the expression: "message.Replace ("friend! What's your name?" name)."
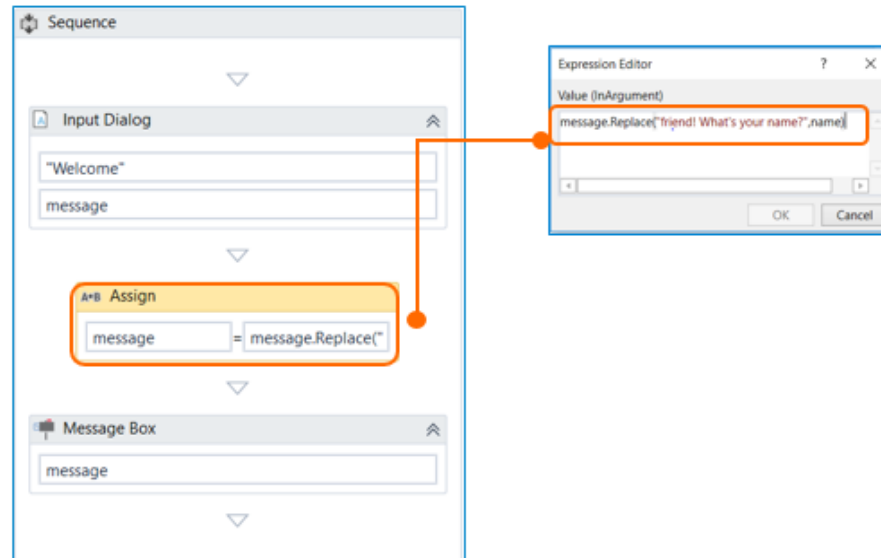
Step 3:

- Show the new value of the message variable ("Hello" + the name that the user has input.

Sequence

Input Dialog

"Welcome"

message

Assign

message = message.Replace("

Message Box

message

Welcome

Hello friend! What's your name?

Ok

UiPath.Core.Activities.InputDialog

Common	
DisplayName	Input Dialog
Input	
IsPassword	☐
Label	message
Options	An array of
Title	"Welcome"
Misc	
Private	☐
Output	
Result	name

Sequence

Input Dialog

"Welcome"

message

Assign

message = message.Replace("

Message Box

message

Expression Editor

Value (InArgument)

message.Replace("friend! What's your name?",name)

OK   Cancel

Sequence

Input Dialog

"Welcome"

message

Assign

message = message.Replace("

Message Box

message

Message ...

Hello John

OK

String.Split

"String.Split" is a command that is used to **separate strings in words based on specific criteria**. This criterion could be a space, a comma, etc. This method comes in handy in a situation where we need to separate a Full name and convert it based on First and Last name.
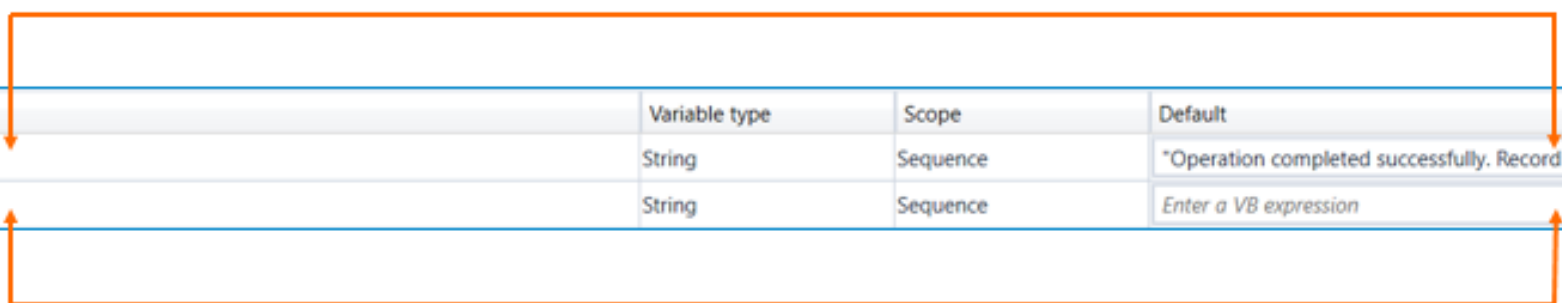
- In our example, **a record number is automatically generated**, and we need to extract it and show it in a message box. The variables we need are **"status" (to store the status) and "record' (to assign the record number).**

Example

The variable "status" stores the status.

Name	Variable type	Scope	Default
status	String	Sequence	"Operation completed successfully. Record 1234 ha:
record	String	Sequence	*Enter a VB expression*

The variable "record" assigns the record number.

Step 1:

- Use an Assign activity with a "String.Split" method.
- Use space ("") as a separator.
- Isolate the substring "Record".
- Use the expression status.split (" "C)(4)

Sequence

▽

A+B Assign

| record | = | status.Split(" "c)(4) |

▽

Message Box ≫

"Your ticket number is " + record

▽
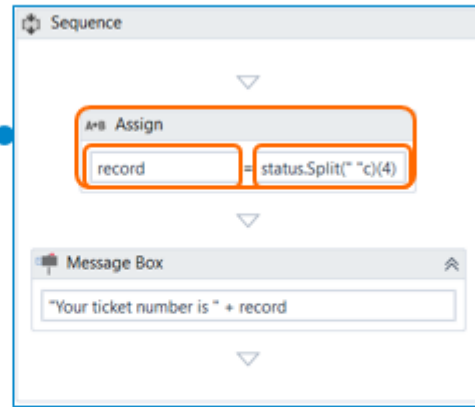
"Operation completed successfully. Record 1234 has been created"

(0) (1) (2) (3) (4) (5) (6) (7)

Step 2:

- Display the result in a message box.

Sequence

▽

A+B Assign

| record | = | status.Split(" "c)(4) |

▽

Message Box ≫

"Your ticket number is " + record

▽

Warning ✕

Your ticket number is 1234

OK
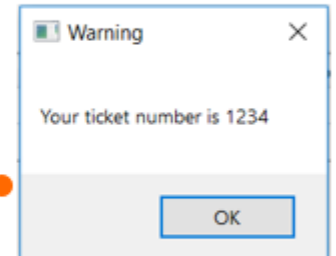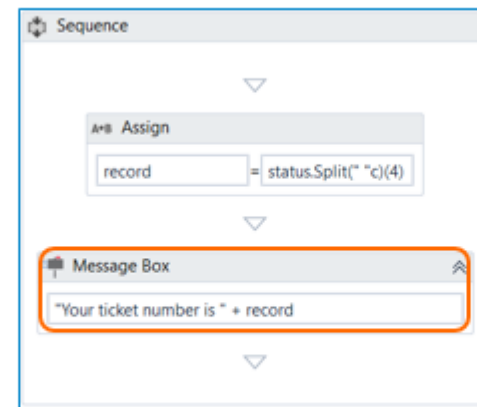
String.Trim and String.Substring

String.Trim is used to remove a part of a string.

Syntax	Function String.Trim(ParamArray trimChars As Char()) As String

String.Substring is used to isolate a part of a string for further use.

Syntax	Function String.Substring(startIndex As Integer, length As Integer) As String

- **"String.Trim"** is used to remove an indicated part of a string. It is the part that we no longer need within a larger string.

- **"String.Substring"** is used to isolate/separate a substring from the original string.

In order to find out the location from where a person called based on their phone number kept in string variable "Phone Number 00123456789

Steps:

1. Use "String.Trim" to remove everything before ":".
2. Use "txt.Split(":"c)(1).Trim" to split the part before the ":".
3. Use (Substring(2,2):) to isolate the 2 figures after the first 2 figures.

1. To isolate the actual phone number, we use "String.Trim" to remove everything before ":".
2. After trimming out the number, we need the 10 digits after the country code "0". Hence, we will trim the obtained number after the String "0". In this example, we are using both methods in the same syntax: "+" + txt.Split(":"c)(1).Trim.Substring(2,2).
3. The first half of the expression "txt.Split(":"c)(1).Trim" is used to splitting the part that is before the ":" from the part that is after.
4. The second half (Substring(2,2):) is used to isolate the 2 figures that are after the first 2 figures (in our case "91").

In order to find out the location from where a person called based on their phone number kept in string variable "Phone Number 00123456789

String.ToLower and String.ToUpper

String.ToLower is used for converting a string to lowercase, while String.ToUpper converts a string to uppercase.

1. Use "Name.Split(" "c)(1).ToUpper + " " + Name.Split(" "c)(0).ToLower" to convert a string to lowercase, and uppercase respectively.

2. "Name.Split(" "c)(1).ToUpper" splits the "Name" string and converts the part after space to upper case.

3. "Name.Split(" "c)(0).ToLower" splits the Name string and converts the part before the space to lower case.

Data Table

- **A Data Table is a tabular format for displaying any information.** It is a collection of data, resources or information where you can insert or set any information that is displayed in the tabular form. It contains the data in rows and columns form. **For example Excel, CSV, etc.**

Name	Roll No	Marks
John	1	75
Paul	2	82
David	3	70
kerry	4	95
Simson	5	45

- The **Data table is a type of variable in RPA** that is used to store and write the data in the form of rows and columns.

- It represents the information of **data as a database in a spreadsheet**. The variables enable us to migrate data from one database to another. After that, we can easily extract the information by data store or website. In UiPath, you can create or **build a Data Table in one of the two ways:**

  - Build Data Table activity
 - Data Scraping

Build Data Table activity

Ways to build Data Table

Data Scraping

Build Data Table Activity

Build Data Table Activity is used when a user has to store the data manually inside the data table.

- You can **drag and drop sequence**. After that, you can insert or drag the build Data Table activity inside the sequence to create a process of a data table.
- **Click on the Data Table**; A pop up will appear on the UiPath project screen.
- Click on the **Add Column button**. Set the name and data type of the column.
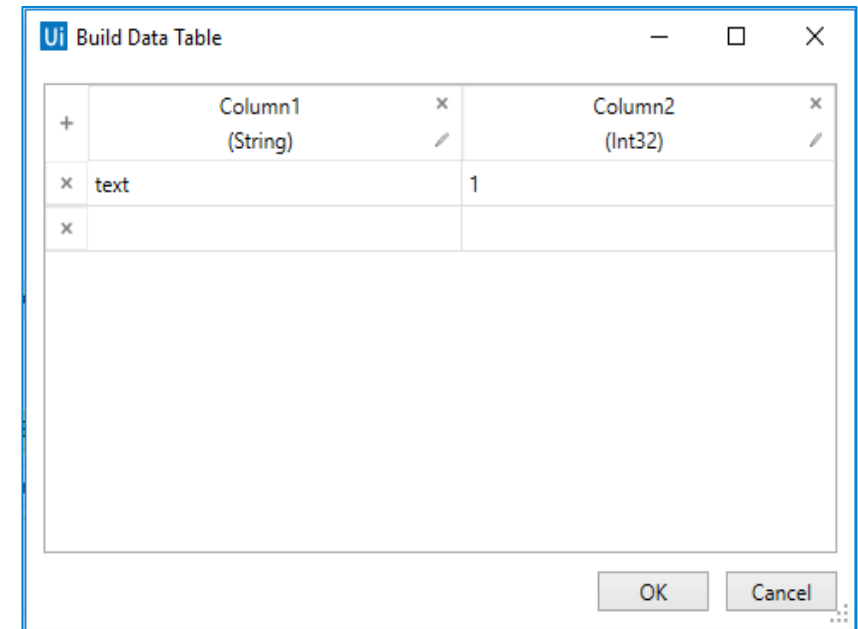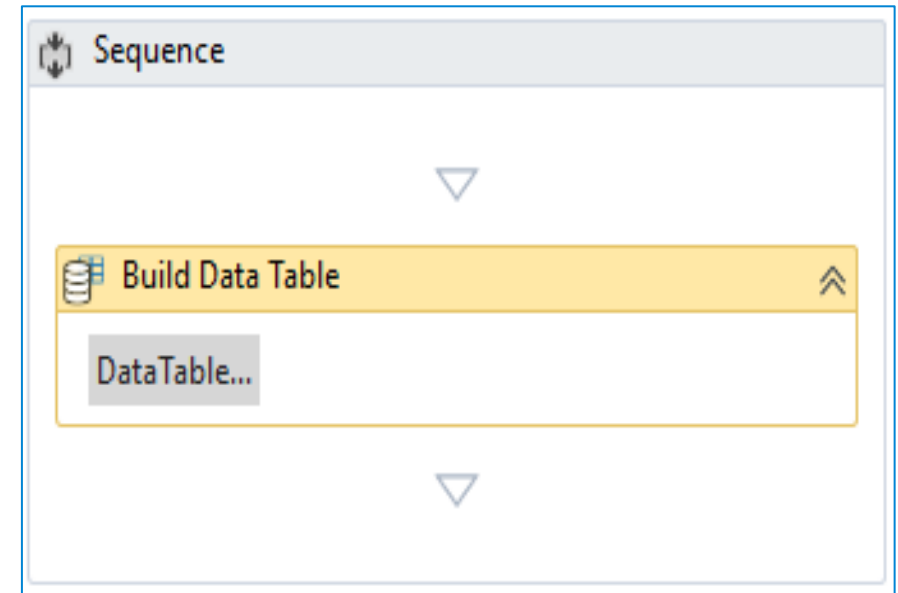
- **If Allow Null is checked**, it is not compulsory to have additional data in the column.

- .**If the Default Value column is blank**, it will automatically take default value .

  - **Max Length:** The number of characters allows the columns, and if you do not want to apply the length of maximum data, then you can set the default value -1 which is also a set of the default values.

  - **Unique:** If the dataset is selected for a specific value then you can create or develop unique data in the data table.

  - **Auto Increment:** When you set the Data Type in the form of int32 then the checkbox display on the screen after this the data automatically increase every time by new row activity with 1 new row.

Column Name	
Data Type	String
Allow Null	☑
Auto Increment	☐
Default Value	
Unique	☐
Max Length	-1

OK   Cancel

Column Name	
Data Type	String
Allow Null	☑
Auto Increment	☐
Default Value	
Unique	☐
Max Length	-1

OK   Cancel

- We can **add several data** inside the data table.

- **Click on the Properties tab of the build Data Table.** Create the variable in the output column with the name "Sample Data."

	Name (String)	Roll No (Int32)	Marks (String)
×	"john"	1	75
×	"Paul"	2	82
×	"David"	3	70
×	"Kerry"	4	95
×	"Simson"	5	45
×			

OK   Cancel

Properties

UiPath.Core.Activities.BuildDataTable

Common

DisplayName — Build Data Tal

Misc

Private — ☐

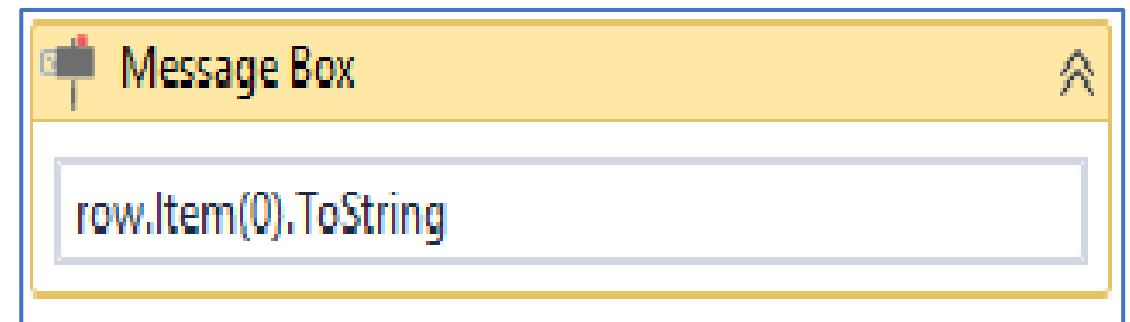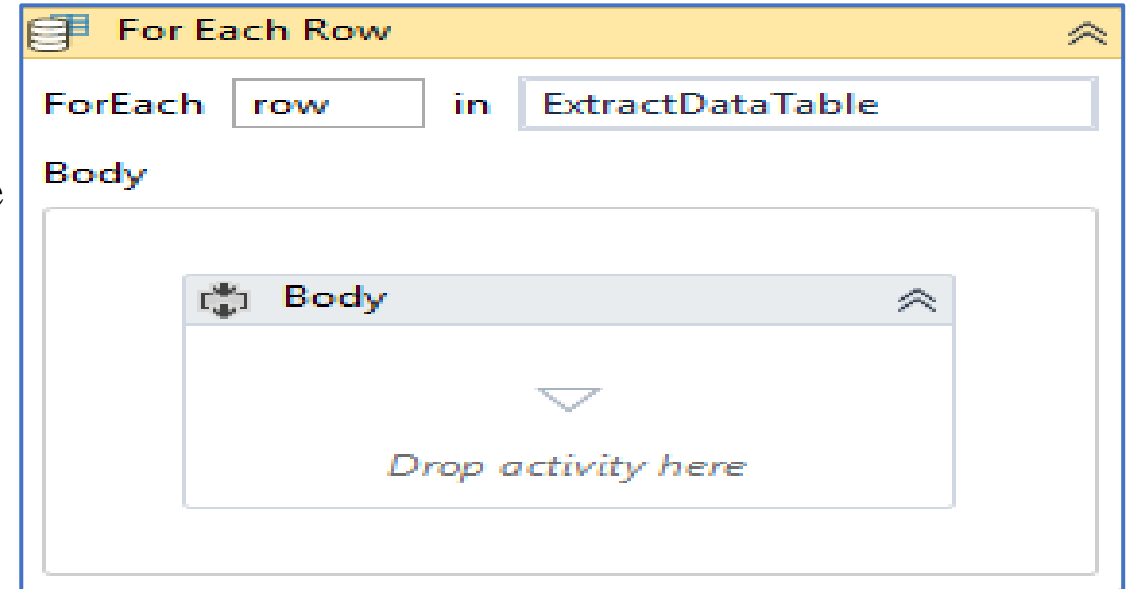**Output**

DataTable — SampleData

Extracting Data from Data Table

- In UiPath, the **"For Each Row" activity** is used to extract data from the data table. The steps to be followed are:

  1. Drag the "For Each Row" activity.
  2. Pass the variable "Extract Data Table" inside it.
  3. Drag the message box inside the body of the message
     .
  4. Write **"row.Item(0).ToString".** inside it.

- In UiPath, the "Filter Data Table" activity is used to filter data from the data table. The steps to be followed are:

  1. Drag the **"Filter Data Table"** activity.
  2. Click on the Filter Wizard.
  3. Pass the data table variable inside the input Data table

---

For Each Row

ForEach | row | in | ExtractDataTable

Body

> Body

> Drop activity here

Message Box

row.Item(0).ToString

data

▲ Programming

    ▲ DataTable

        🗄 Add Data Column

        🗄 Add Data Row

        🗄 Build Data Table

        🗄 Clear Data Table

        🗄 Filter Data Table

        🗄 For Each Row

        🗄 Generate Data Table

        🗄 Get Row Item

        🗄 Join Data Tables

        🗄 Lookup Data Table

        🗄 Merge Data Table

        🗄 Output Data Table

        🗄 Remove Data Column

        🗄 Remove Data Row

        🗄 Remove Duplicate Rows

        🗄 Sort Data Table

Data Scraping
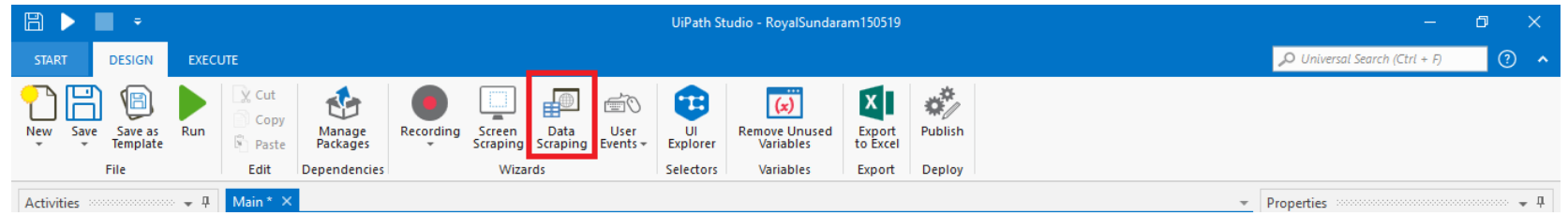
- **Data Scraping is used to create a data table at run time**. It can extract all the pattern-based data and store it into the form of the data table automatically. When you build a project, it enables you to extract data from a browser, document, Excel file, .CSV file and an application to a database.

  Let us consider extracting Data from Amazon website.

1. Drag and drop the Open Browser activity inside the sequence.
2. Type the URL inside the open browser text box.
3. An activity window will pop up. Click on the Next button.
4. Click on the Data Scraping icon.
5. Select the name of the mobile .
6. Modify the data or variable name. Click on the next button.
7. Data is displayed in one column table.
8. Click on Extract Correlated Data to extract the second field, price.
9. The data is displayed in a Data table.
10. Click on Extract Correlated Data to extract more fields.
11. Click on the Yes button to extract data from multiple pages
12. A variable with the name **"ExtractDataTable** is created.

Sequence

Open Browser

"https://www.amazon.in/s/ref=nb_sb_noss_2?url=search-alias%3Daps&fi

Do

Drop activity here

Ui Extract Wizard ✕

Select Element

1. Open your browser, application or document and navigate to where you want to extract data from.

2. Press Next in this dialog and hover the mouse cursor over a data source field.

3. Click that field.

| Help | | Cancel | < Back | Next |

Ui Extract Wizard ✕

Configure Columns

The identified fields are highlighted.

☑ Extract Text

Text Column Name   Column1

☐ Extract URL

URL Column Name   Column2

| Help | | Cancel | < Back | Next |

Preview Data

Column1
Apple iPhone X (64GB) - Silver
Apple iPhone 6 (Gold, 1GB RAM, 32GB Storage)
Apple iPhone 6s (32GB) - Space Gray
Apple iPhone XR (128GB) - Black
Apple iPhone 7 (32GB) - Black
Apple iPhone XR (64GB) - White
Apple iPhone 6s (32GB) - Rose Gold
Apple iPhone X (64GB) - Space Grey
Apple iPhone XR (64GB) - Black
Apple iPhone 8 (64GB) - Space Grey
Apple iPhone X (256GB) - Silver
(Refurbished) Apple iPhone 8 MQ6L2HN/A (Silver, 64GB)
[Sponsored]Apple iPhone Xs Max (512GB) - Space Grey
[Sponsored]Apple iPhone 8 (Red, 256GB)
[Sponsored]Apple iPhone 6s (32GB) - Silver
[Sponsored]Apple iPhone Xs (64GB) - Silver

Edit Data Definition   Maximum number of results (0 for all)   100

| Help | | Cancel | < Back | Extract Correlated Data | Finish |

Preview Data

Column1	Column2
Apple iPhone X (64GB) - Silver	74,999
Apple iPhone 6 (Gold, 1GB RAM, 32GB Storage)	21,999
Apple iPhone 6s (32GB) - Space Gray	29,599
Apple iPhone XR (128GB) - Black	72,999
Apple iPhone 7 (32GB) - Black	39,785
Apple iPhone XR (64GB) - White	67,999
Apple iPhone 6s (32GB) - Rose Gold	27,999
Apple iPhone X (64GB) - Space Grey	74,999
Apple iPhone XR (64GB) - Black	67,999
Apple iPhone 8 (64GB) - Space Grey	56,999
Apple iPhone X (256GB) - Silver	
(Refurbished) Apple iPhone 8 MQ6L2HN/A (Silver, 64	
	485
	295
	485
	499
[Sponsored]Apple iPhone Xs Max (512GB) - Space Gr	1,44,900
[Sponsored]Apple iPhone 8 (Red, 256GB)	68,999
[Sponsored]Apple iPhone 6s (32GB) - Silver	29,487
[Sponsored]Apple iPhone Xs (64GB) - Silver	96,900

Edit Data Definition   Maximum number of results (0 for all)   100

| Help | | Cancel | < Back | Extract Correlated Data | Finish |

Ui Indicate Next Link ✕

Is data spanning multiple pages?

Identify the element that navigates to the next pages. It could be a 'Next' button or an arrow (not a page number).

Press Yes to indicate it.

| Yes | No |

‹ Previous Page   1 2 3 ... 20   **Next Page** ›

UiPath.Core.Activities.ExtractData

⊟ **Common**	
ContinueOnError	True
Display Name	Extract Structured
⊟ **Input**	
ExtractMetadata	"<extract><cc
⊞ Target	Target
⊟ **Misc**	
Private	☐
⊟ **Options**	
DelayBetweenPagesMS	300
MaxNumberOfResults	100
NextLinkSelector	"<webctrl id='
⊟ **Output**	
DataTable	ExtractDataTab

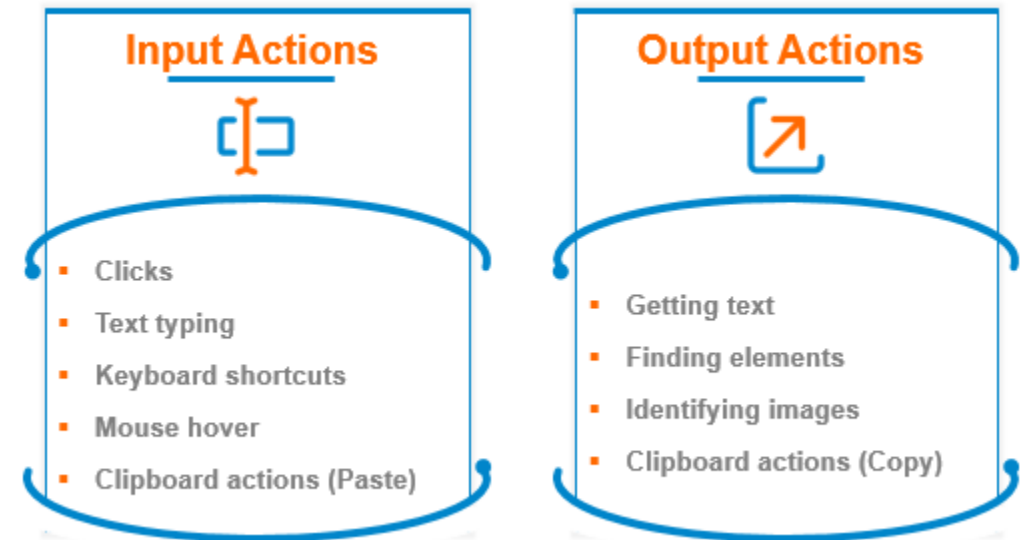# UI AUTOMATION & SELECTORS: Types of UI Interactions

- There are two types of User Interface interactions that can appear in an automation:
  - **Inserting data into an application (input),**
  - **Reading data from an application (output).**

Inserting data into an application (input):

- Every time we click on an UI element, we send information to the application and expect something from it.
- It's the same when we type text, we intend to produce an outcome, a change or a continuation.
- We aim to produce the same kind of effects when we send hotkeys, when we hover the mouse or when we use the paste activity.

Reading data from an application (output):

- If we take out data from an application, we are talking about output actions such as getting text out of a process,
- finding elements or identifying images as a way to read data that was produced or borrowed by an application. Also,
- when we copy text from an application using the clipboard activity, we are talking about an output action.

Input Actions

- Clicks
- Text typing
- Keyboard shortcuts
- Mouse hover
- Clipboard actions (Paste)

Output Actions

- Getting text
- Finding elements
- Identifying images
- Clipboard actions (Copy)

Practice Exercise - Input Methods

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

For humans, The only way to present input actions such as clicks, types and so on is to use the mouse and the keyboard.

For robots, things are different. UiPath supports 3 input methods:

- **The default method:** This method replicates the human method i.e. the mouse cursor moves on the screen.
- **Send Window Messages:** This method replicates the messages that an application receives when the user utilizes the keyboard and the mouse.
- **Simulate Type/Click:** This method acts like a developer programmatically changing the value of an editable field, using the technology of the target application.

	Compatibility	Background	Speed	Hotkeys	Empty Field
Default	100%	NO	50%	YES	NO
Window Messages	80%	YES	50%	YES	NO
Simulate Type/Click	99% (for Web Apps) 60% (for Desktop Apps)	YES	100%	NO	YES

The default method:

Working
- **Clicks:** The mouse cursor moves across the screen
- **Typing:** The keyboard driver is used to type individual characters

Implications
- Attended User cannot touch the mouse or keyboard during the automation
- It has a lower speed and load times can impact accuracy

Strong points
Supports special keys like Enter, Tab, and other hotkeys

Limitations
- Does not automatically erase previously written text
- Does not work in the background

⊟ Options	
⊞ CursorPosition	CursorPosition
KeyModifiers	None ⌄
SendWindowMessages	☐
SimulateClick	☐

- It has an advantage over the other 3 methods: it uses the keyboard driver. It can replicate the touch of every key, including the special keys. Moreover, it is the only method that has 100% compatibility, no matter which applications are automated.

Send Window Messages

Working
- Replays the window messages that the target application receives when the mouse/keyboard is used
- Clicking and typing occur instantly

Implications
- Works in the background
- Comparable to the Default method in terms of speed

Strong points
- Supports special keys like Enter, Tab, and other hotkeys
- Users can work on other activities during the execution of the automated processes

Limitations
- Does not automatically erase previously written text
- Works only with applications that respond to Window Messages

Options	
CursorPosition	CursorPosition
KeyModifiers	None
SendWindowMessages	☑
SimulateClick	☐
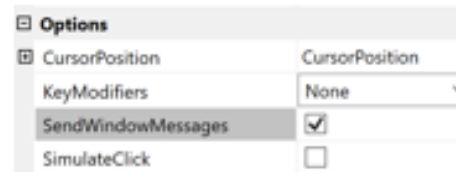
- **One of the biggest advantages of this method is that it works in the background**, thus the users that work on the same machines and the attended robots can continue their work and perform other tasks. As the window messages support hotkeys and shortcuts, this method can be used to send special characters in applications.

Simulate Type/Click

Working
- Uses the technology of the target application to send instructions
- Clicking and typing occur instantly

Implications
- Works in the background
- Actions are a lot faster, but there are some compatibility limitations

Strong points
- Can automatically erase previously written text
- Users can work on other activities during the execution of the automated processes

Limitations
- Does not support special keys like Enter, Tab, and other hotkeys
- It has a lower compatibility than the other 2 methods

Options	
CursorPosition	CursorPosition
KeyModifiers	None
SendWindowMessages	☐
SimulateClick	☑
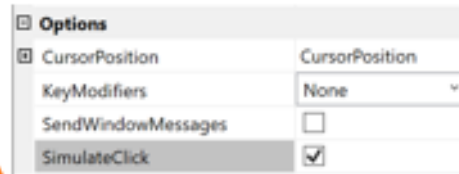
- This method uses the technology of the target applications to change the value of an editable control. **It attempts to communicate at API (application programming interface) level with each application.** Once the controls are identified, the commands are sent and processed instantly. **It is by far the fastest of the 3 input methods,** but only works with applications whose controls react to the commands sent. As a result of the way it works, it can function in the background.

Output Actions and Methods/ screen scraping methods

- Methods are the core of all the activities that enable **extracting data from a specified UI element or document**. UiPath currently has 3 output methods available:

	Speed	Accuracy	Background	Extract Text Position	Extracts Hidden Text	Supports Citrix
FullText	10/10	100%	YES	NO	YES	NO
Native	8/10	100%	NO	YES	NO	NO
OCR	3/10	98%	NO	YES	NO	YES

- **The FullText method**: It is the **default output method in most cases**. It is the fastest method, has 100% accuracy and can work in the background. Moreover, it is able to extract hidden text (for example, options in a drop-down list). However, it doesn't support Citrix and doesn't capture text position and formatting.

- **The Native method:** It is compatible with applications that **use the Graphics Design Interface**, the Microsoft API responsible for representing graphical object. Its speed is lower than FullText and it cannot work in the background. However, it can extract the text position and formatting and has 100% accuracy. Just like FullText, it doesn't support Citrix

- **The OCR (Optical Character Recognition) method:** It is the only one that works with Citrix. **Its technology relies on recognizing each character just like we recognize faces in a photography.** It cannot work in the background, cannot extract hidden text, and its speed is by far the lowest. Its accuracy varies from one text to another, and changing settings can also improve the results. Just like the Native method, it also captures the text position.

The FullText Output Method

The FullText output method **captures all the text from a terminal screen**. It is the **default method in UiPath**. It is fast, accurate and can work in the background.

- It captures all the text from the terminal screen, including the hidden text (from options, drop-down lists, and so on).
- On the other hand, it doesn't work on Citrix and other virtual environments and it doesn't retain formatting and text position.

Ignore Hidden: When this check box is selected, the hidden text from the selected UI Element is not copied.

The Native Output Method

- The Native output method works with applications that are built to render text with Graphics Device Interface (GDI). It can extract the screen coordinates of the text

The Native method offers the following options:

- **No Formatting:** When the font and color are not important, this box can be checked, and the wizard extracts only the text, just like the FullText method.
- **Get Words Info:** This option can extract the position of each word and can support several separators. By default, it can process all known characters as separators (comma, space, and so on), but when only certain separators are specified, it can ignore the others. Additionally, the Custom Separators field is displayed, enabling the user to specify the characters used as separators. If the field is empty, all known text separators are used.

No Formatting: when this check box is selected, the copied text does not extract formatting information from the text.

Get Words Info: when this check box is selected, the screen coordinates of each word are extracted.

Scraping Method | Native ▼

Scrape Options
☑ No Formatting
☐ Get Words Info

Refresh

The OCR Output Method

- The OCR output method uses the Optical Character Recognition technology, to extract information from virtual environments. It has two default engines: Google Tesseract and Microsoft MODI

- Although, both FullText and Native have excellent results in terms of accuracy and speed, there are specific cases in which both of them are unusable.

- For example, if we need to extract information from virtual environments (Citrix or Remote Desktop) or is "read" text from images, the **Optical Character Recognition(OCR)** output method should be used. It is based on the OCR technology used in recognition of scanned documents. Basically, it attempts to recognize each letter or given image in the target document.

  The OCR method has two default engines:
  - **Google Tesseract:** It gets better results on smaller size areas and supports color inversion. It has filters that can be used to select only certain categories of characters.
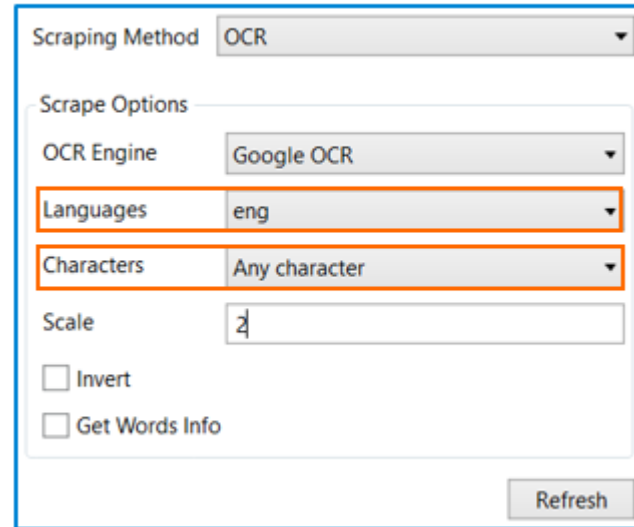
  - **Microsoft MODI:** It performs better when it comes to Microsoft fonts and on larger size areas. Moreover, it supports multiple languages.

Google Tesseract

The Google Tesseract OCR engine is more effective with character recognition on small size areas. It offers multiple customization options:

Languages: Enables language change for the scraped text. By default, English is selected.

Characters: Enables the selection of type of characters to be extracted: any character, numbers only, letters, uppercase, lowercase, phone numbers, currency, date and custom.
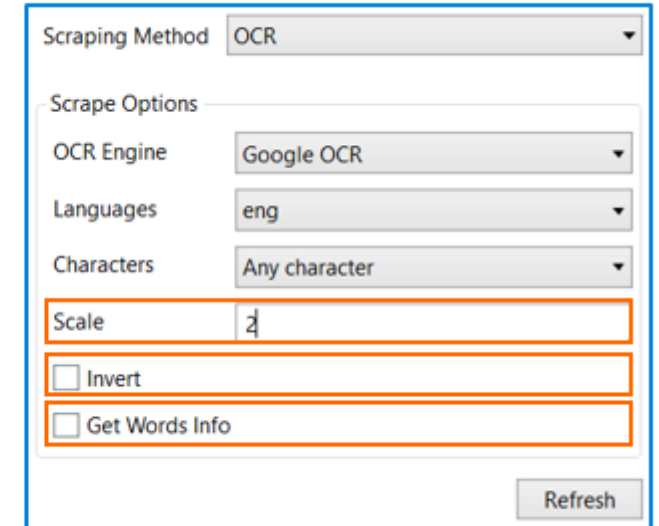
Scale: Helps the user specify the scale of the text to be scraped.

Invert: When this check box is selected, the colors of the UI element are inverted before scraping.

Get Words Info: Gets the on-screen position of each scraped word.

■ Languages: By default it offers only English support. To add more languages, go to https://github.com/tesseract-ocr/tessdata and search for the desired language.

■ Characters: It allows customization of the types of characters to extract. It provides a Custom option with two additional fields, Allowed and Denied, which allow the user to choose which types of characters to scrape and which to avoid.

■ Scale: In most of cases, the higher the scale, the better is the chance of recognition. But, there are cases in which a lower scale provides a better outcome.

■ Invert: This is a very useful option when the background is darker than the text color. This can happen for screen captures of darker themed apps and websites and scanned documents.

■ Get Words Info: This option comes in enables the user to specify the position of the text to be scraped.

Microsoft MODI

The **Microsoft Office Document Imaging (MODI)** is best used to read Microsoft fonts.

- Just like the Google Tesseract, it can also extract the exact screen position. In general, it works better with larger documents, but it's recommended to make some changes to the Scale and notice the differences. It offers multiple customization options:

Languages: Enables language change for the scraped text. By default, English is selected.

Scale: Helps the user specify the scale of the text to be scraped.

Get Words Info: Gets the on-screen position of each scraped word.

Scraping Method	OCR ▼

Scrape Options

OCR Engine	Microsoft OCR ▼
Languages	English (United States) ▼
Scale	1

☐ Get Words Info

Refresh

Takeaways : Input & Output Methods

UiPath supports 3 input methods:
- The default method
- Send Window Messages
- Simulate Type/click

UiPath supports 3 output methods:
- The FullText method
- The Native method
- The OCR method

Recording & its types

- The Recording functionality of UiPath Studio helps in **capturing user's actions on the screen** and translating them into sequences.
  - It captures or mimics every activity step by step. The UiPath Recorder works in the same way as Macros recording apps (Automation).
  - It automatically generates the sequence of all the activities which we need in the workflow for automation. Apart from this, it also allows you to modify the property of activities and to change in workflow sequence as well.
  - There are five recorders, in the UiPath. All of them come with their own controllers to perform common actions as well as actions particular to each environment.

How does recording work?

- Basic Recorder
- Desktop Recorder
- Web Recorder
- Image Recorder
- Native Citrix Recorder

- The recording tool can be accessed from the 'Design' tab in UiPath Studio.
- While recording, all user interface elements are highlighted, allowing easy identification of buttons, fields, menus, or elements the user interacts with.
- Once the recording ends, a sequence will be created, containing the activities performed by the user.
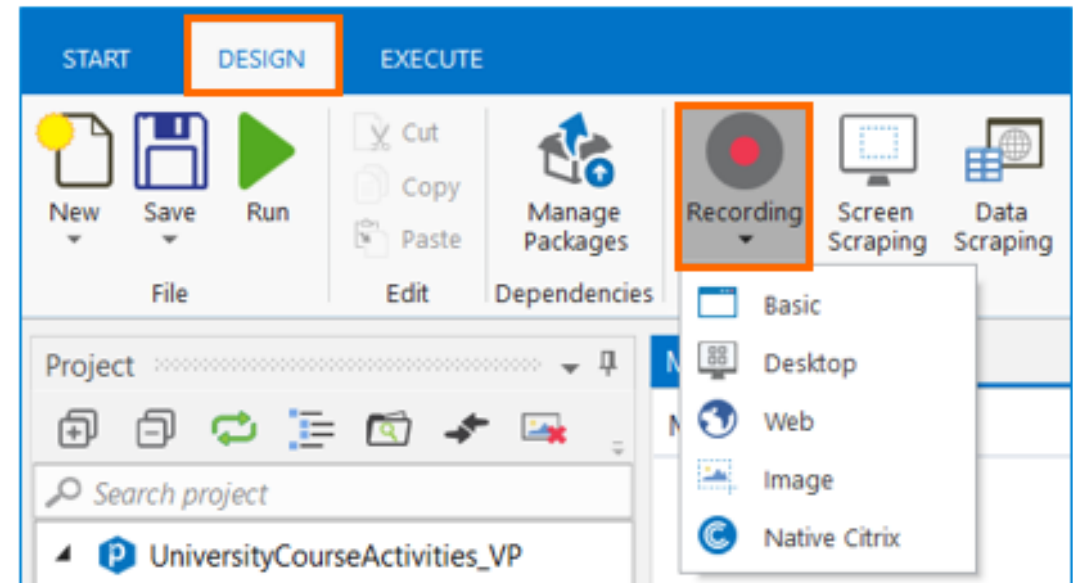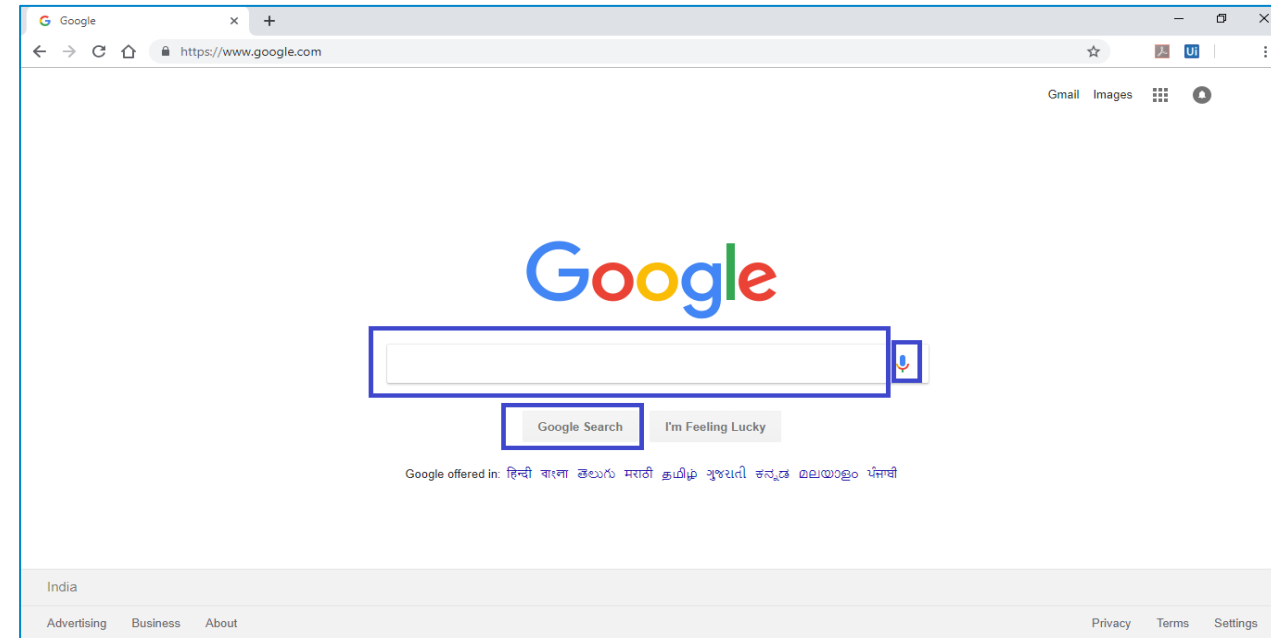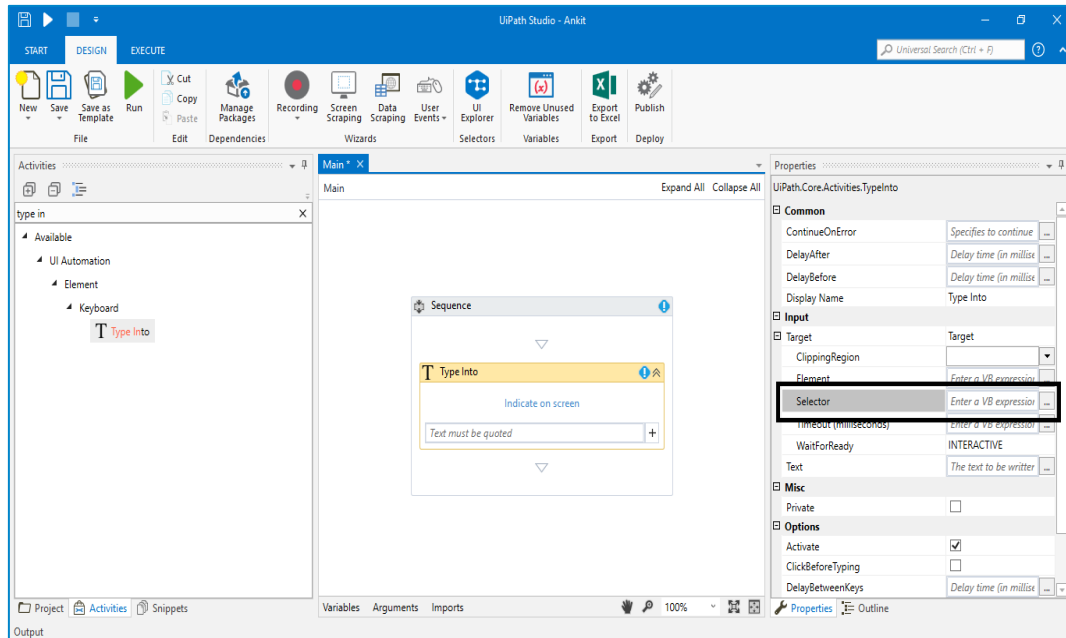
Selectors

UI Element

- **Selectors** are a fundamental part of UiPath Studio that are **used to recognize the objects** on the project screen.
- When we have to automate a specific task or activity action, we have **to communicate with various UI Elements.** The selector can be looked upon as a specific identifier to find a particular UI element amongst multiple applications

UI element refers to all graphical user interface pieces that construct an application.

For example, when you open **www.google.com**, you see the following items:

- A search bar(Text Field)
- Search Button
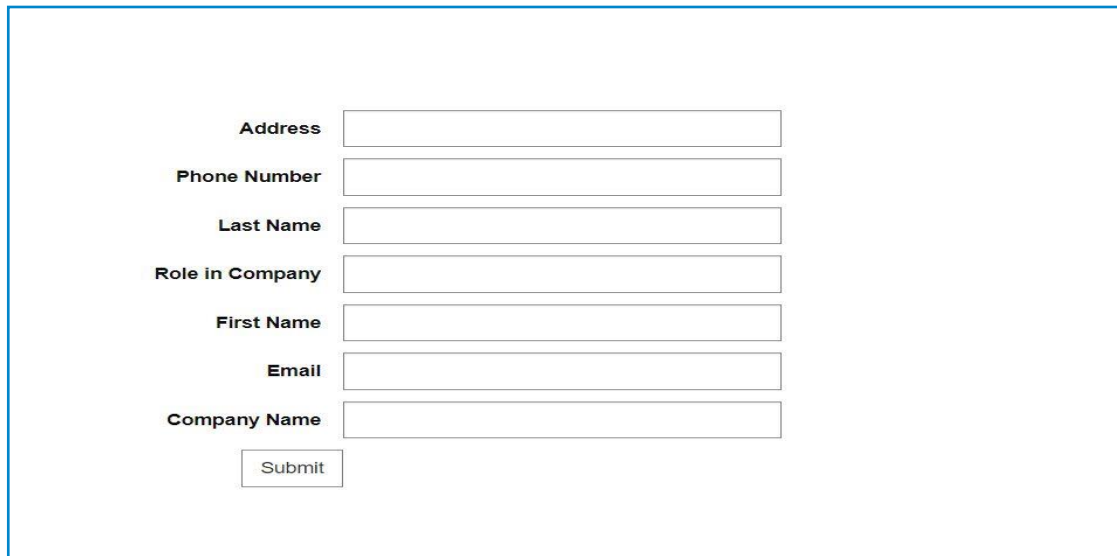- A mike shaped image for audio search.

User Interface and its Types

The UI elements are embedded in **User Interface** that acts like a container holding all the pieces that construct an application. Selectors are the way to find and identify these elements. In short, selectors are the address of the UI Elements.
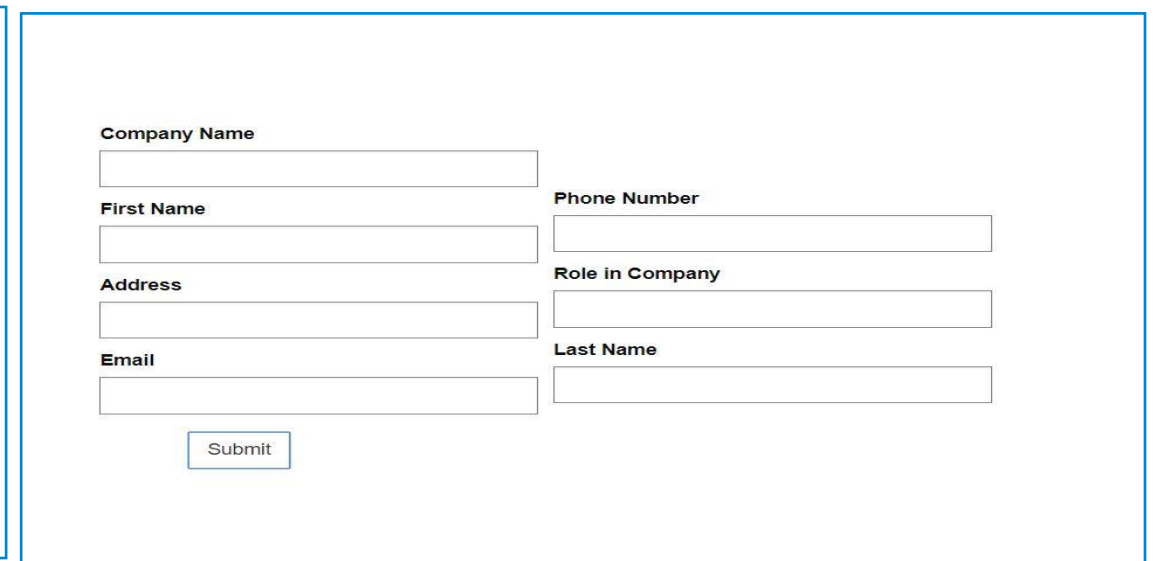
Static **and** Dynamic User Interface

Static Interface Scenario

Dynamic Interface Scenario

The UI element entitled "Address" will always be found at this exact pixel coordinate in the part of the web page. If the layout does not change, the selector will remain valid through its operations.

In this example, the layout has changed although it contains the same UI elements. The selector from the first example would become invalid as the pixel positioning of the "Address" element has changed.

Types of Selectors

The Selectors are defined by looking at the element they target to perform their specific activity.

Full Selectors: **Full selectors contain all the required elements to identify a UI element, including the top-level window and recommended when switching between multiple windows.** In the case of full selectors, the Editor and the Explorer are not grayed out and are displaying the full selector as we can see in the example.

Partial Selectors: **Partial selectors are the ones that are mainly generated by the Desktop Recorder**. They do not contain information about the top-level window, as you can see it's grayed out in the Editor section as well as the Explorer section. They mostly recommended when performing multiple actions in the same window. It can edit only elements belonging to the partial selector. The prepended ones are grayed out and read-only.

Dynamic selectors: **The dynamic selectors can change specific attribute values based on the selected variable**. For example, let's presume that we have this calendar on a web page, and we want to click a specific date and receive this action as user input. To do this, we can use the dynamic selector to click the specified date by the user. We store the input from the user in a variable, and then we put it inside the selector. The robot will receive the date, day, and month, identify the specific element from the calendar GUI and perform the required action.

Example for dynamic selector:
It would help you to create a robot that would add events in your calendar based on the emails that you receive, or the sticky notes that you save on your computer and other applications.
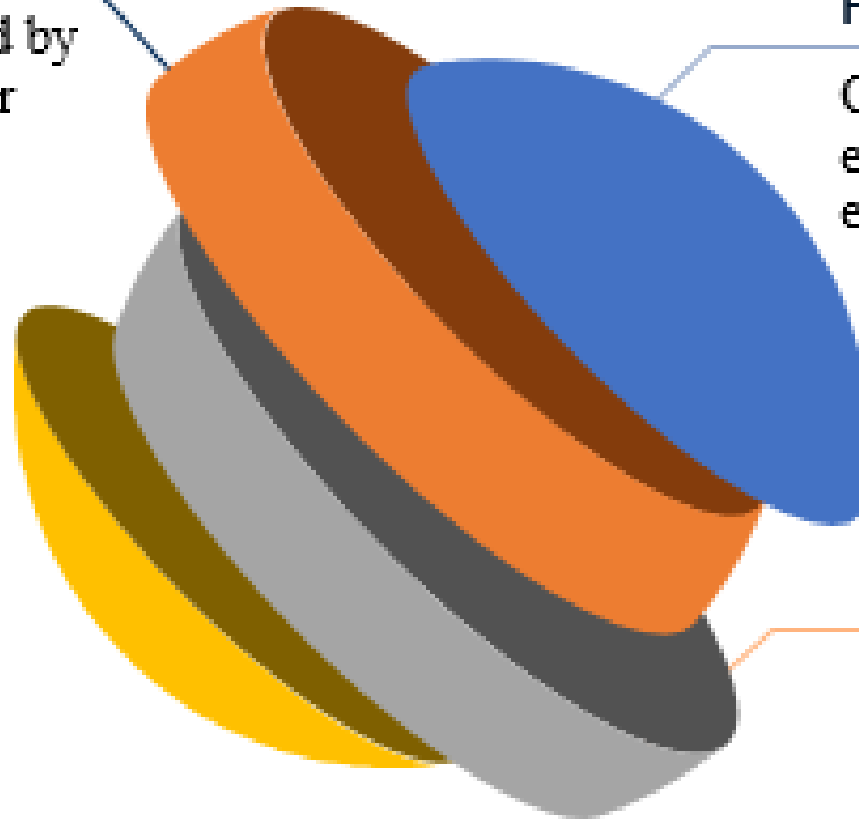
Partial Selectors

Are mainly generated by the Desktop Recorder

Full Selectors

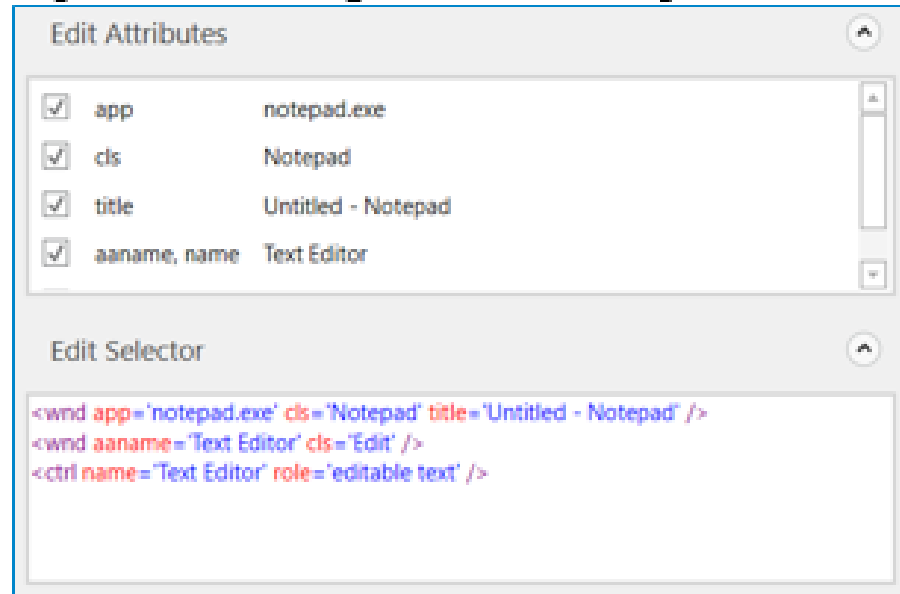Contain all the required elements to identify a UI element

Dynamic Selectors

Can change attribute values based on the selected variable.

Full selectors

Best suited for situations in which the action performed requires switching between multiple windows.

Partial selectors

Best suited for performing multiple actions in the same window.

Editor Section

Explorer Section

Dynamic selectors

Best suited for situations in which the targeted element can constantly change it's its value.

Customizing Selectors

A wildcard is a special character that can replace the dynamic part of a selector.

- When we select a UI element; the selector captures its properties. However, these properties differ when we select the UI element of a different instance of an application with the selector.

- After this, The selector does not recognize the same UI element of another instance of the application. In this case, we can fix this problem by using wildcard characters or by attaching it to a live Element. There are two types of wildcard:

  - **The question mark symbol, ?, which replaces one character**

  - **The asterisk symbol, that is, *, which replaces several characters**
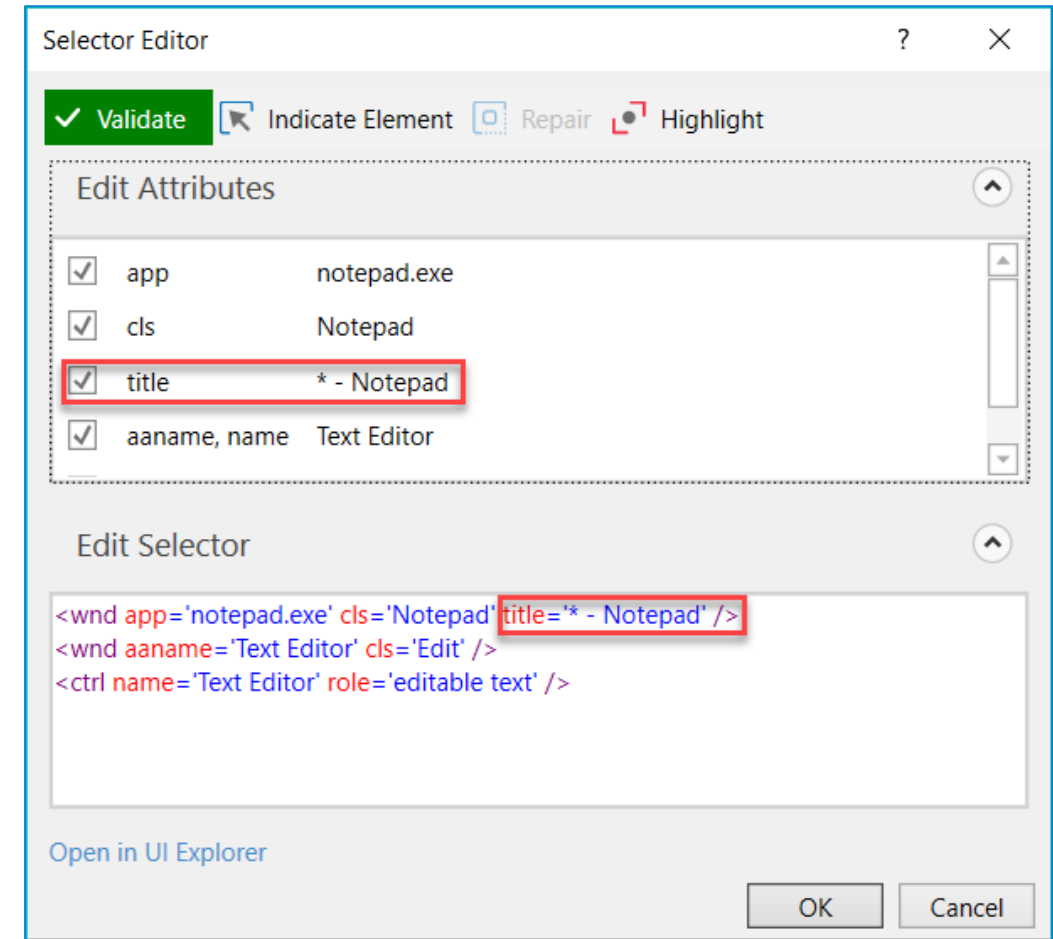
Properties of the Selector Editor
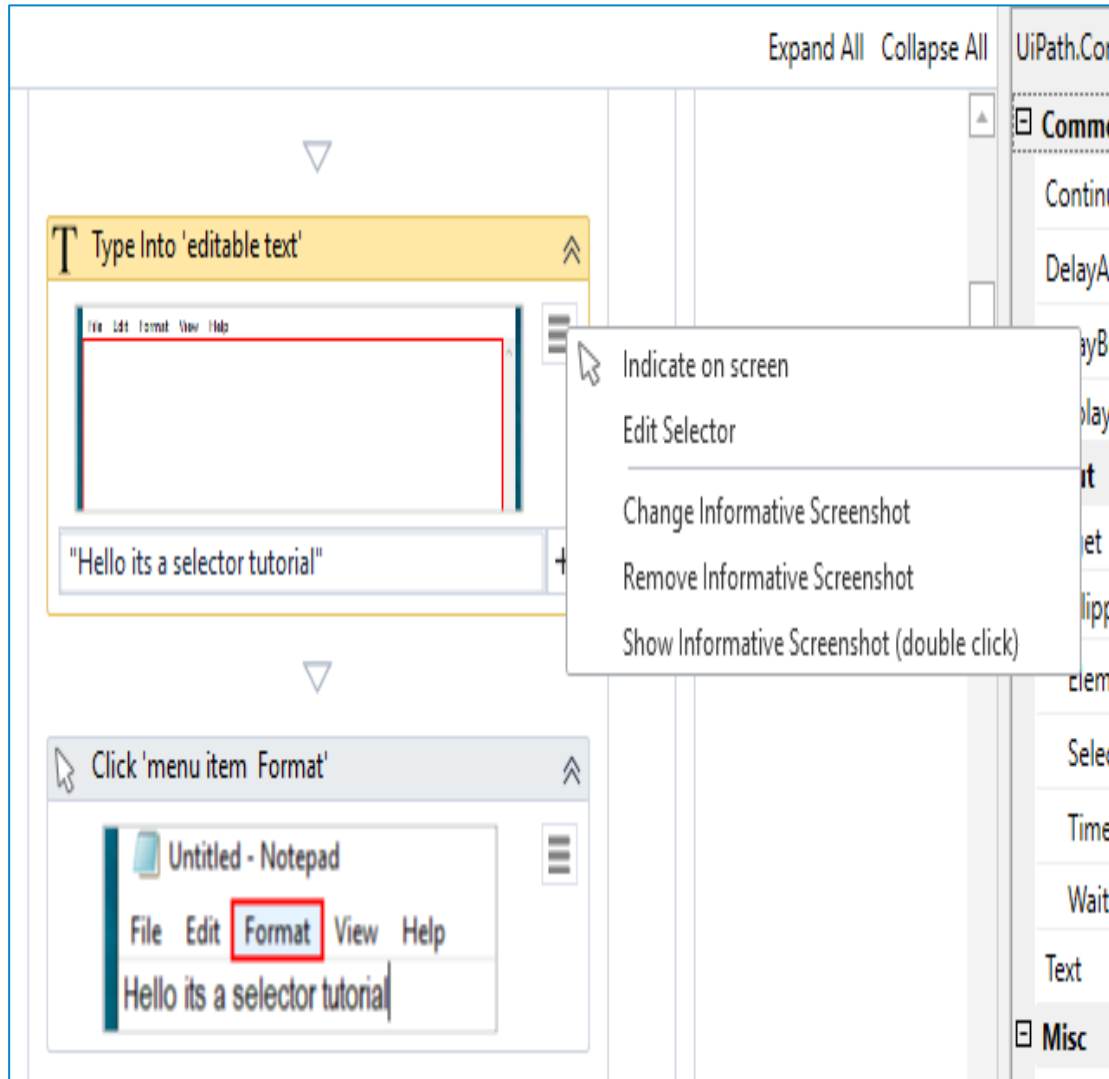
When we click on the hamburger button, the following options appear in a drop-down list:

Indicate on Screen: Indicate on screen is used to ease and simplify the automation where the target element is changed, and the selectors are automatically created.

Edit Selector: It is used to edit the selectors already created. If the selectors do not work due to any reason, they can be edited by using the "edit selector "option.

Change Informative Screenshot: When we select any element, a sample screenshot is created, in case we want to change the screenshot, this option can be used.

Remove Informative Screenshot: An auto-generated screenshot can be deleted by using this option.

Show Informative Screenshot (Double Click): You can double click on the process to show the informative screenshot or image.

Selector status

In the selector editor window, you can see the selector's status where the validity status can be either green, red, yellow or grey.

- If the selector is valid then it will appear as **green.**

- If the selector is validated then it shows in **grey color** but in the invalid selector case, it will show in red color.

- In case the change is made in selector after validations, the color appears as **yellow.**

Option	Description
Validate	The button shows the status of the selector by checking the validity of the selector definition and the visibility of the target element on the screen. The **Validate** button has three states: • ⟳ Validate — Selector is being validated • ✓ Validate — Valid selector • ✕ Validate — Invalid selector • ? Validate — Modified selector, revalidate The button is correlated with UI Explorer validation states.
Indicate Element	Indicate a new UI element to replace the previous one.
Repair	Enables you to re-indicate the same target UI element and repair the selector. This operation does not completely replace the previous selector. The button is available only when the selector is invalid.
Highlight	Brings the target element in the foreground. The highlight stays on until the option is disabled with a click. The button is enabled only if the selector is valid.
Edit Attributes	Contains all the application components needed to identify the target application (a window, a button etc.). This section is editable.
Edit Selector	Holds the actual selector. This section is editable.
Open in UI Explorer	Launches the UI Explorer. The option is enabled only for valid selectors.

Keyboard Based Automation

UiPath Studio features activities that simulate any type of keyboard input that a regular human user would use.
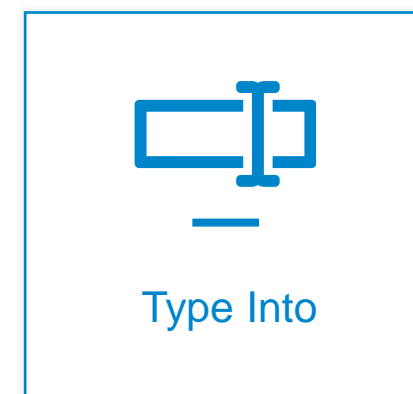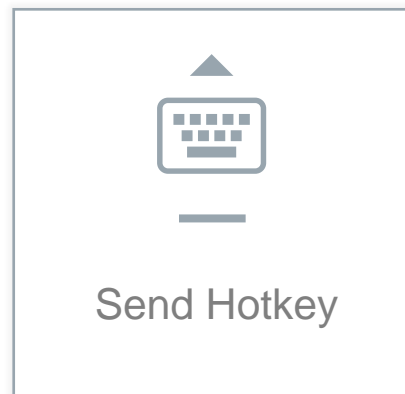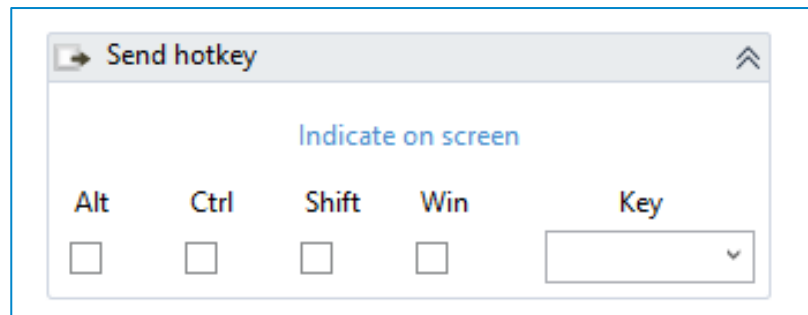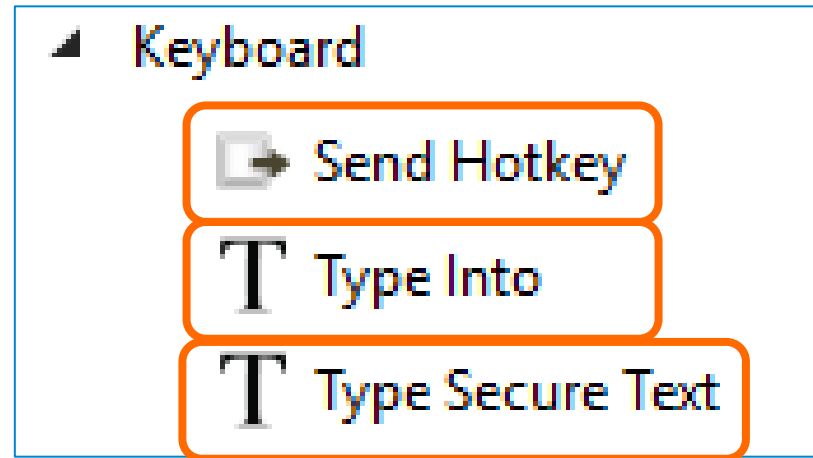
Information Retrieval
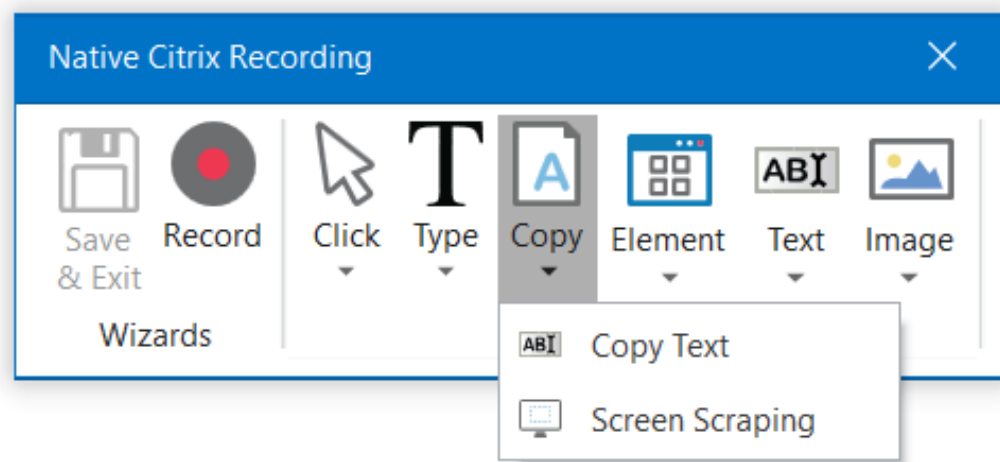
There are multiple processes involved in Retrieving Information:

A. Copy Text

Copy Text, as the name implies, **selects editable texts and copies it to the UiPath environment through the clipboard**. All action is performed on the active text field.

B. Screen Scraping

Screen Scraping allows you to scrape just a portion of an image relative to an anchor.

It locates a fixed element on the screen and then using OCR extracts the information.

The activities generated by the wizard are:
- Find Image (to find the anchor image)
- Set Clipping Region (to find the clipping region, offset to the anchor image)
- Get OCR text (to extract the data from the clipping region).

Anchor

In UiPath, the term Anchor is **used to describe a process that enables the user to catch or recognize the relative element of that element.**

- In UiPath, the term Anchor is described as a method that should be used when a constant selector is not available in the data table. The anchor is an efficient process that ensures that we are capable of catching or recognizing the relative element of that element.
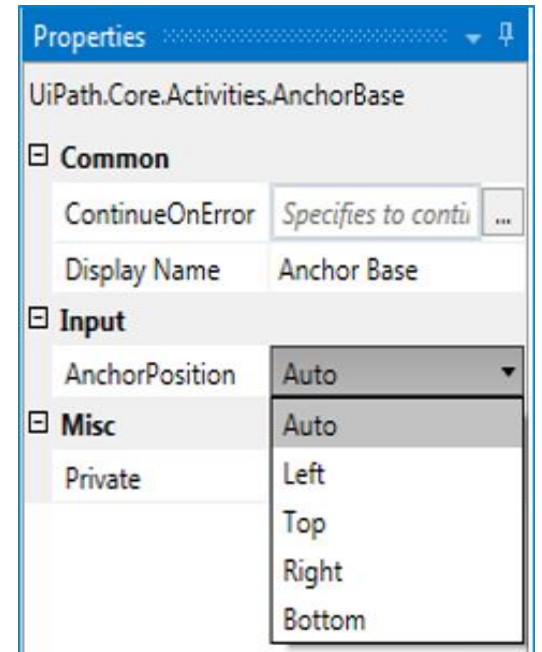
Properties of Anchor

Input: It refers to the **specific area or place where the UI element is to be referred**. The anchor position defines a significant portion of the input.

Anchor Position refers to the edge of the UI element that has been anchored.

Common: Under the Common section, there are two properties "ContinueOnError" & "Display name."

- **ContinueOnError is a common property for many activities**, It Accepts Boolean values, so there are only two options which we can enter here – True or False.

  - If we enter a true: The execution of the process will continue whether there's an error or not but if we enter false or leave it blank: The execution of the process stops, an error is thrown.

Types of Container Activities

01 Open Application
Launches a specified application & performs actions within it

02 Attach Browser
Enables the user to attach to an open browser & perform actions within it

03 Open Browser
Enables the user to open a browser with a specified URL & execute activities within it

04 Attach Window
Enables the user to attach to an already open window & perform actions within it

05 Get Active Window
Retrieves the current active Window & enables the user to perform actions within it

Full Selectors vs. Partial Selectors

UiPath Reboot Work.

01 Full Selectors

- Created through Basic Recorder
- Start with a window, or an HTML identifier
- Contain all the information of the UI Element
- Suitable when the workflow has to switch through multiple windows

02 Partial Selectors

- Created through Desktop Recorder
- Do not contain the top-level window information
- Suitable when the robot has to perform multiple actions in the same window or application

Editor Section

Edit Attributes

☑	app	notepad.exe
☑	cls	Notepad
☑	title	Untitled - Notepad
☑	aaname, name	Text Editor

Edit Selector

```
<wnd app='notepad.exe' cls='Notepad' title='Untitled - Notepad' />
<wnd aaname='Text Editor' cls='Edit' />
<ctrl name='Text Editor' role='editable text' />
```

Edit Attributes

☑	app	notepad.exe
☑	cls	Notepad
☑	title	Untitled - Notepad
☑	aaname, name	Text Editor

Edit Selector

```
<wnd app='notepad.exe' cls='Notepad' title='Untitled - Notepad' />
<wnd aaname='Text Editor' cls='Edit' />
<ctrl name='Text Editor' role='editable text' />
```

Explorer Section