



**RAMAIAH**  
Institute of Technology

# **RPA Tool Introduction And Basics**

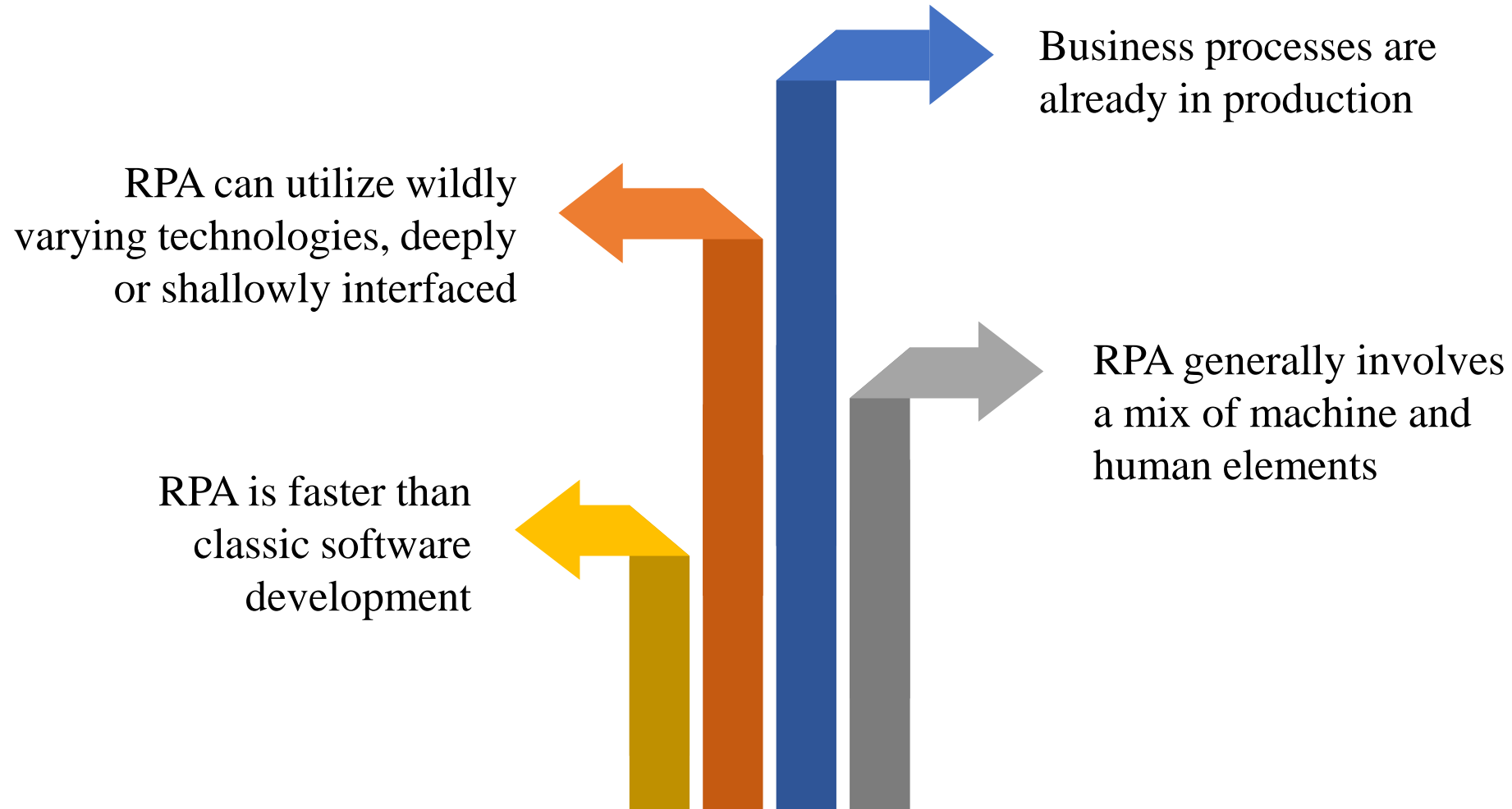
# Software Development Life Cycle (SDLC)

The SDLC is the framework used to develop software solutions.



# IS SDLC Applicable to RPA Development

The main stages of SDLC are found in RPA development projects, but there are some specific differences:



# RPA Development

- Software development Life Cycle plays a crucial role in completing the final delivery of product.
- The RPA development life cycle is categorized into four major parts:
- **Analysis:**
  - The term analysis refers to the discovery of automation opportunity from the existing business process.
  - This involves understanding the process map (as it is) and how the process is done manually by the user.

# RPA Development

- **Analysis:**

- The business analyst gathers all the important information about the underline automation and creates a document which has the relevant information about the project.

- **Development:**

- Once the requirement gathering is done, the RPA developer gets the design approved from project manager and starts the development of robot using the automation tool.

# RPA Development

- **Testing:**

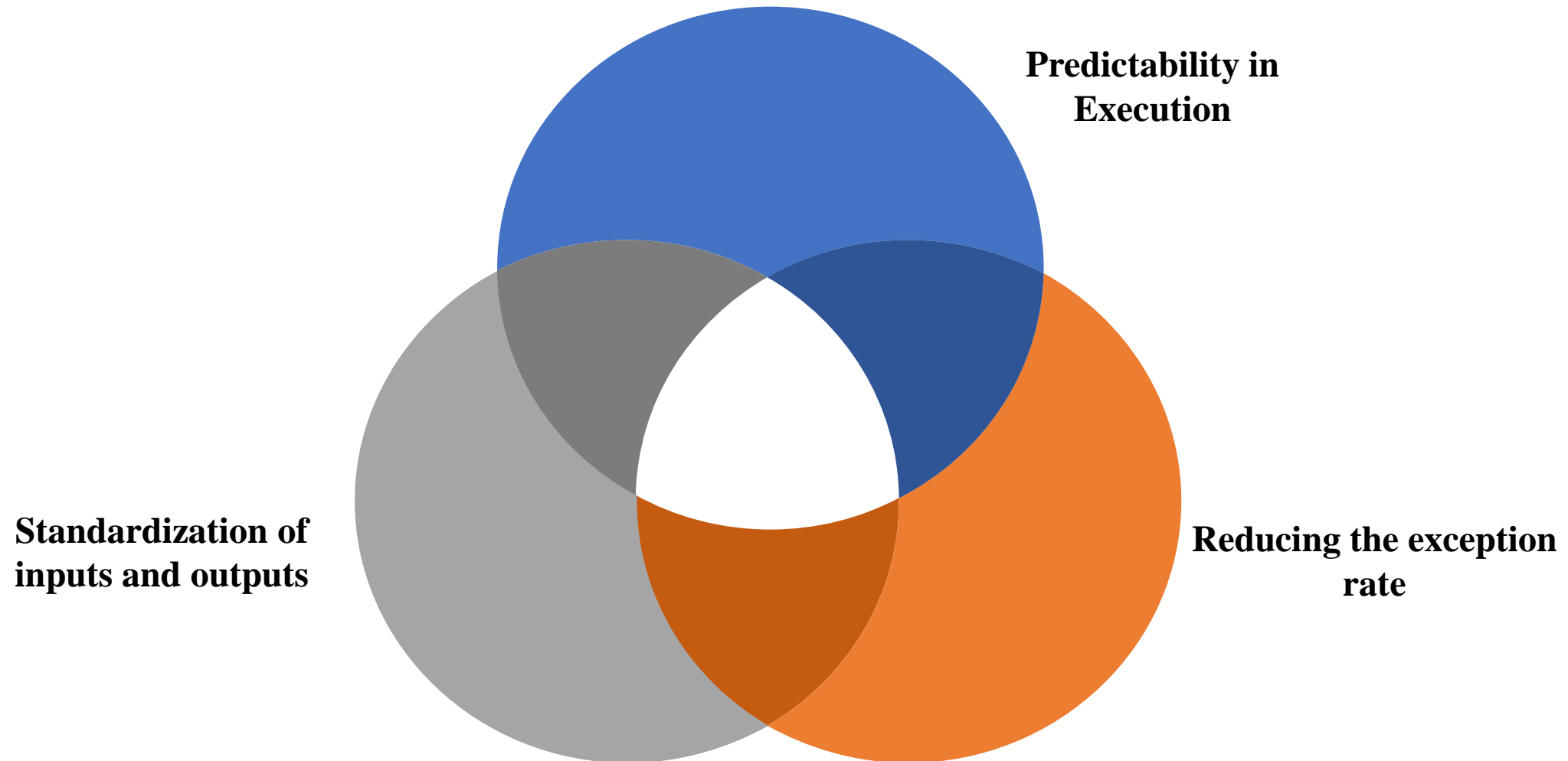
- The developer shares the code (ready product) with the business user to test the code and check if the delivered product (code) meets all the business requirements.

- **Support and Maintenance:**

- Once the robot is deployed, it is very important to ensure and monitor the working of the robot.

# Process Standardization

Process standardization consists of establishing rules that govern the way to execute a given task or sequence of tasks. The typical areas of standardization are:



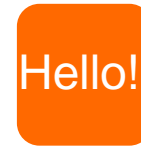
# Variables

Variables are storage containers for data that can be later used throughout the program.



## Counter

Keep track of a certain repetitive action, like clicking on an item



## Comments

Store the comments for each action item

## Organized Items

Store organized items of interest to the users that clicked on them



# Variables

- Variables are containers that store different types of information.
- Using variables in computer programs makes it easier to label and store data which can later be used throughout the program.
- It's useful to think of variables like boxes that keep different types of data.
- The data in the box might change, but the box remains the same.

# Variables

- In one box, you might keep a counter that tracks the number of time users clicked on an item.
- In another box, you could store ‘comments (text)’ written by the user(s) on that item.
- In another box, you could store a table of items that may be of interest to the user based on the item they click.

# Actions associated with Variables

Some of the actions associated with variables are:

Create it



Add content  
in it



Reuse it



Get info out  
of it



# Properties of Variables



## **Name**

Title of the information stored by the variable



## **Value**

The expression value assigned to the variable



## **Lifetime**

Lifetime of a variable ends when program leaves its scope



## **Type**

Type is classification of variable based on the type of data it is intended to store.



## **Scope**

Designate which parts of the program can see it or use it (local, global)



## **Location (Memory)**

The place where variable is stored in the computer hardware

# Best Practices for Naming Variables

- A variable should be meaningful and describe the information they are storing.
- **Naming Best Practices:**
  - When you start to build a major automation process then it is simple to ignore the variable declaration that what you had done that's why it is crucial to put the naming system good in place.
  - You can always use accurate names, username in the variable which stores the user name.

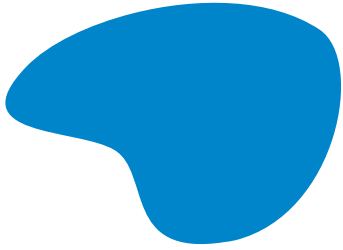
# Best Practices for Naming Variables

- The Naming Best Practices rules that should be applied:
  - Be consistent in the way you assign names.
  - Use meaningful names: the name of the variable should accurately describe its content.
  - Keep your variable names short.
  - Try to see if the code can explain itself, it shouldn't have a lot of comments.
  - Align to company naming standards (if any).

# Best Practices for Naming Variables

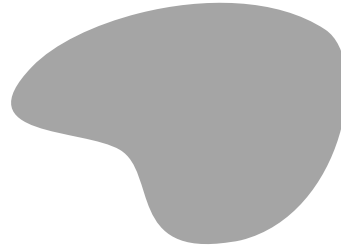
- The Naming Best Practices rules that should be applied:
  - Align to programming language standards:
    - use Camel Case (aka Upper Camel Case) for classes: `VelocityResponseWriter`
    - use Lower Case for packages: `com.company.project.ui`
    - use Mixed Case (aka Lower Camel Case) for variables: `studentName`
    - use Upper Case for constants : `MAX_PARAMETER_COUNT = 100`
    - use Camel Case for enumeration class names and Upper Case for enumeration values.

# Types of Variables



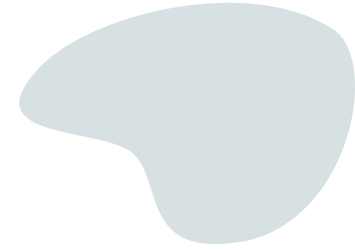
## **Number**

Stores Numerical Data



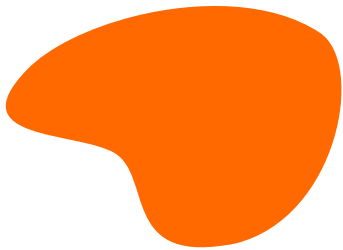
## **Array**

Stores multiple values



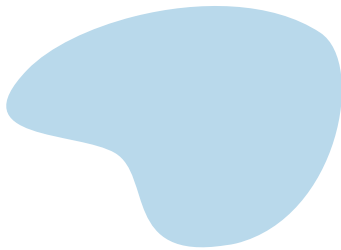
## **String**

Stores Text



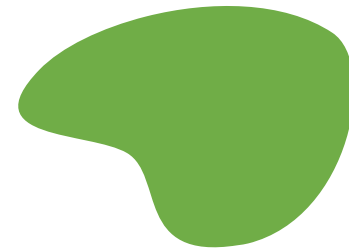
## **Date & Time**

Stores Date and Time



## **Boolean**

Stores True or False



## **DataTable**

It store the information



# Types of Variables

Number

Array

String

Date & Time

Boolean

DataTable

## Number

### Definition

- Number variables can store numerical values.;

### Example

- An example of number variable can be the age of a person.
- `Int age = 34;`

### Usage

- Used to store a certain numerical value inside them and reuse it whenever needed in the program.

# Types of Variables

Number

Array

String

Date & Time

Boolean

DataTable

## Array

### Definition

- An array is a data structure that contains a group of elements of the same data type, such as an integer or string.

### Example

- `arrayname[0] = "This`
- `arrayname[1] = "is "`
- `arrayname[2] =`
- `"pretty simple.";`
- `print arrayname[0];`
- `print arrayname[1];`
- `print arrayname[2];`

;

### Usage

- Used to organize data so that a related set of values can be easily sorted or searched.

# Types of Variables

Number

Array

String

Date & Time

Boolean

DataTable

## String

### Definition

- String Variables help us store any kind of text on which arithmetical operations will not be applied.

### Example

- **arrayname[0] = String firstName = Alex;**
- String lastName = Jackobsen;

### Usage

- The main usage of string variables is to store text and reuse it in our code for specific actions.

# Types of Variables

Number

Array

String

Date & Time

Boolean

DataTable

## Date & Time

### Definition

- Enables you to store information about any date and time.

### Example

- `DateTime value = new DateTime(2017, 1, 18);`

### Usage

- Used to store a specific date and time

# Types of Variables

Number

Array

String

Date & Time

Boolean

DataTable

## Boolean

### Definition

- The Boolean variable has true or false values.

### Example

- Boolean user = true;

### Usage

- Boolean variables are used with control statements to help determine the flow of a program.

# Types of Variables

Number

Array

String

Date & Time

Boolean

DataTable

## DataTable

### Definition

- Variable that can store large pieces of data.

### Example

- A database table
- A simple spreadsheet
- Structured data

### Usage

- They are used to store big pieces of information and do certain operations on it, such as, filtering, searching, copying, etc.

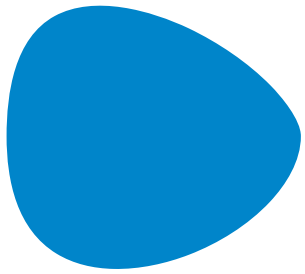
# Types of Variables

Number
Array
String
Date & Time
Boolean
DataTable

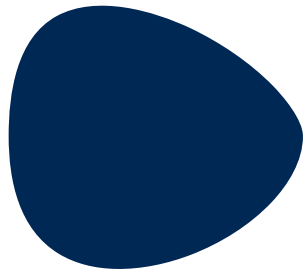
## String vs Array Variables

Array	String
Array is a sequential collection of elements of similar data types.	String refer to a sequence of single characters represented as a single data type.
Elements of arrays are stored contiguously in increasing memory locations.	Strings can be stored in any manner in memory locations.
An array is a special variable that can hold more than one value at a time.	Strings can hold only char data.
Arrays are mutable, the fields can be modified.	Strings are immutable, the value cannot be changed in memory once created.
The length of an array is predefined.	The size of a string is not predefined.

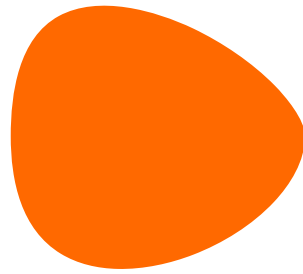
# Types of Variables in UiPath



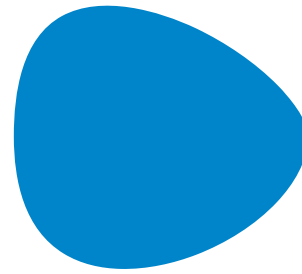
● **Integer**



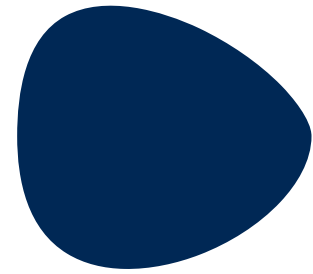
● **String**



● **Boolean**



● **Array**



● **Generic**



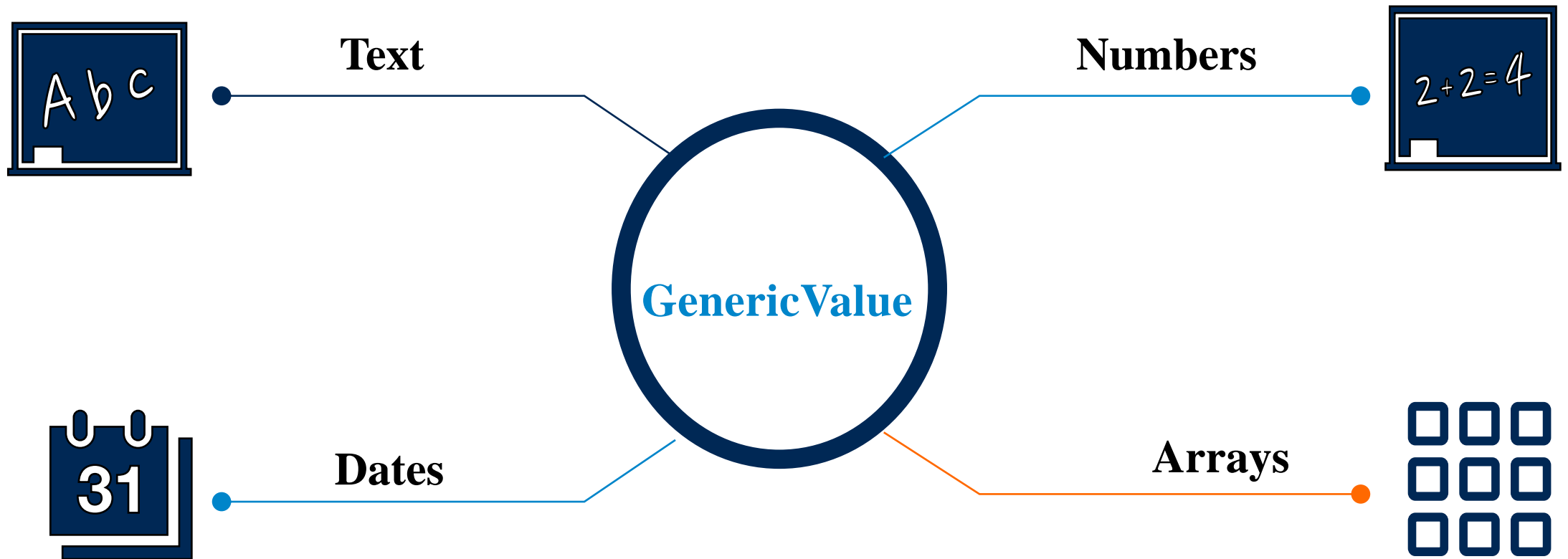
# UiPath Specific Variable - GenericValue

- The **GenericValue** is a type of variable that is particular to UiPath Studio and can store any kind of data, including text, numbers, dates, and arrays.
- **GenericValue** variables are automatically converted to other types, in order to perform certain actions.
- However, it is important to use these types of variables carefully, as the conversion may not always be the correct one for the project.

# UiPath Specific Variable - GenericValue

- UiPath Studio has an automatic conversion mechanism of GenericValue variables.
- First element in your expression takes precedence while deciding the outcome.
- For example, when you try to add two GenericValue variables, if the first one in the expression is defined as a string, the result is the concatenation of the two.
- If it is defined as an Integer, the result is their sum.

# UiPath Specific Variable - GenericValue



# Variables Panel in UiPath Studio

The screenshot displays the UiPath Studio interface with the 'DESIGN' tab selected. The main workspace shows a 'Main' sequence with a 'Sequence' container and an 'Is Match' activity. The 'Variables' panel is open at the bottom, showing a table of variables and a 'Create Variable' button. The 'Properties' panel on the right shows the 'UiPath.Core.Activities.IsMatch' activity properties.

**Activities Panel:** string

- Available
  - Programming
    - String
      - (.?) Is Match
      - (.?) Matches
      - (.?) Replace

**Variables Panel:**

Name	Variable type	Scope	Default
firstName	String	Sequence	Enter a VB expression
lastName	String	Sequence	Enter a VB expression
Create Variable			

**Properties Panel:** UiPath.Core.Activities.IsMatch

- Common
  - DisplayName: Is Match
- Input
  - Input: The string to ...
  - Pattern: The regular ...
  - RegexOption: IgnoreCase, ( ...
- Misc
  - Private: ☐
  - Result: A boolean v ...

# Variables Panel in UiPath Studio



## Name

Give a proper name to your variables



## Variable Type

Specify the data type for your variable



## Scope

Specify the scope of your variable



## Default






Specify if your variable should have a default value or not.

Name	Variable type	Scope	Default
firstName	String	Sequence	<i>Enter a VB expression</i>
lastName	String	Sequence	<i>Enter a VB expression</i>
<i>Create Variable</i>			

Variables

Arguments

Imports

 100% 

# Variables Panel in UiPath Studio

- The properties of Variables in UiPath are as follows:
  - 1. Name:** Here, You can give a name to your variables. If you rename a variable in the Variables pane, it will be renamed in all instances where it's used. This is useful in complex workflows.
  - 2. Variable Type:** Here you can specify the data type that you will store. These variables can be in the string, Array, GenericValue, Int32, and many more forms. Also you can change insert the variable type library by clicking browse types and search any variable library.

# Variables Panel in UiPath Studio

- The properties of Variables in UiPath are as follows:
- 3. Scope:** You can specify the scope of your variable in which it can be visible. It define the scope by public and private method, Where if you define any variable scope in the container then it is applicable only for those containers where you define it that is private scope. But if you want to publicly define the scope then you can define in the main class which is applicable for all containers that are called public scope.
  - 4. Default:** Specify Variable's default value here.

# Collections

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



# Arguments

- An Argument is simply a variable that can store a value.
- Create an argument in the Argument section of the main Designer panel.
- An argument has a larger scope than a variable and is used to pass values between different workflows.
- Suppose we have a big project to build; we break down the project into different workflows because smaller workflows can be easily tested separately.
- Easy to build smaller workflows and combine them.

# Arguments

- Arguments are used for interacting with different workflows by exchanging data between them. That is why the direction property is associated with Arguments.
- We can choose the direction on the basis of our requirement - either giving the value to some workflow or receiving the value from another workflow.
- We can easily create arguments in the Arguments panel.

# Arguments

- We can also specify the direction:
  1. **In:** When we have to receive the value from another workflow.
  2. **Out:** This is the current value if we have to send the value to a workflow.
  3. **In/Out:** This specifies both; it can take or receive the value.
  4. **Property:** This specifies that it is not being used currently.

# Arguments

Drop activity here

Name	Direction	Argument type	Default value
argument1	In	String	Enter a VB expression

Create Argument

Variables Arguments Imports

100%

# Variables vs. Arguments

## Variables

- only used inside a single workflow file(.xaml) in your project
- Types of Variable: Boolean, Array, Integer, text., Datatable
- It is used to pass it other activities.
- You can change the variable name multiple times

## Arguments

- used to pass data from one workflow file to another
- have a specific direction – In, Out, In/Out
- Type of Argument: Argument, Property, Direction, Default etc.

Name	Variable type	Scope	Default
ABC	Int32	Sequence	<i>Enter a VB expression</i>

Name	Direction	Argument type	Default value
XYZ	In	String	<i>Enter a VB expression</i>
ABC	Out	String	<i>Default value not supported</i>

# Arguments Panel in UiPath Studio



## Name

Give a proper name to your arguments



## Direction

Set up the direction of your argument



## Type

Specify the data type for your argument



## Default

Specify if your argument should have a default value or not.

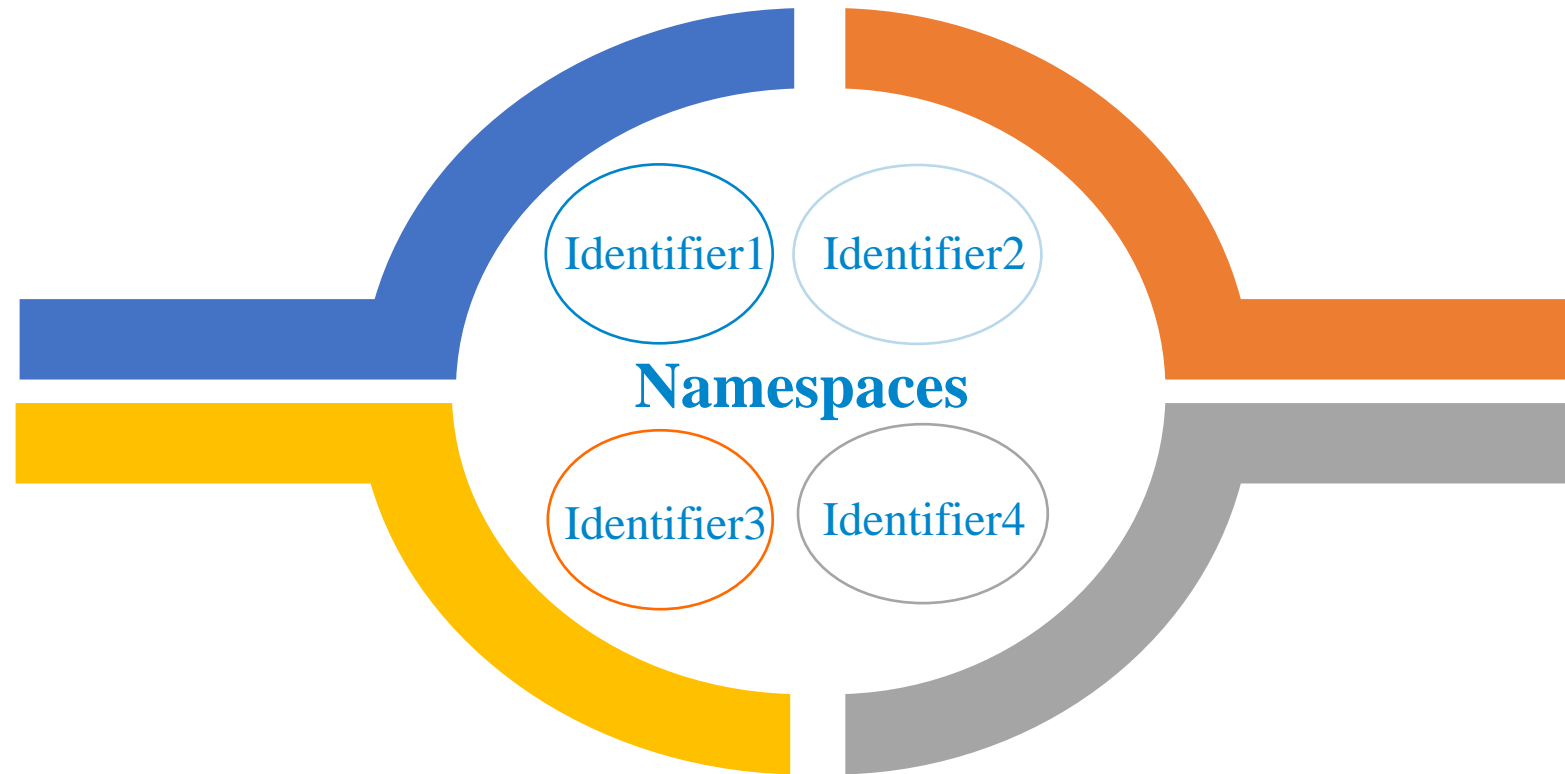
Name	Direction	Argument type	Default value
In_Values	In	Int32	Enter a VB expression
Out_Total	Out	String	Default value not supported
Create Argument			

Variables Arguments Imports

100%

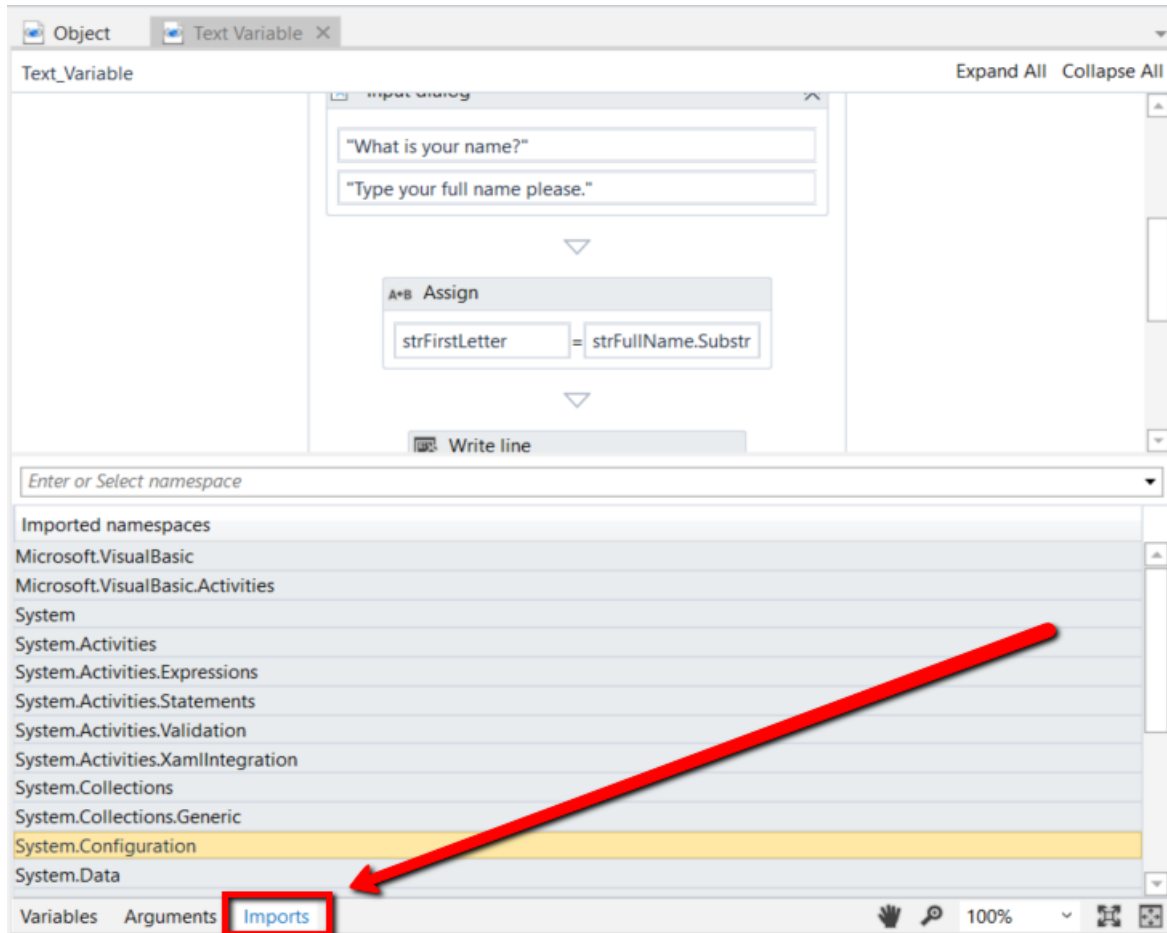
# Namespaces

**Namespaces** are containers created to hold a logical grouping of unique names (**identifier**).



# Namespaces

VB.NET namespaces in UiPath Studio represent containers that store different types of data.



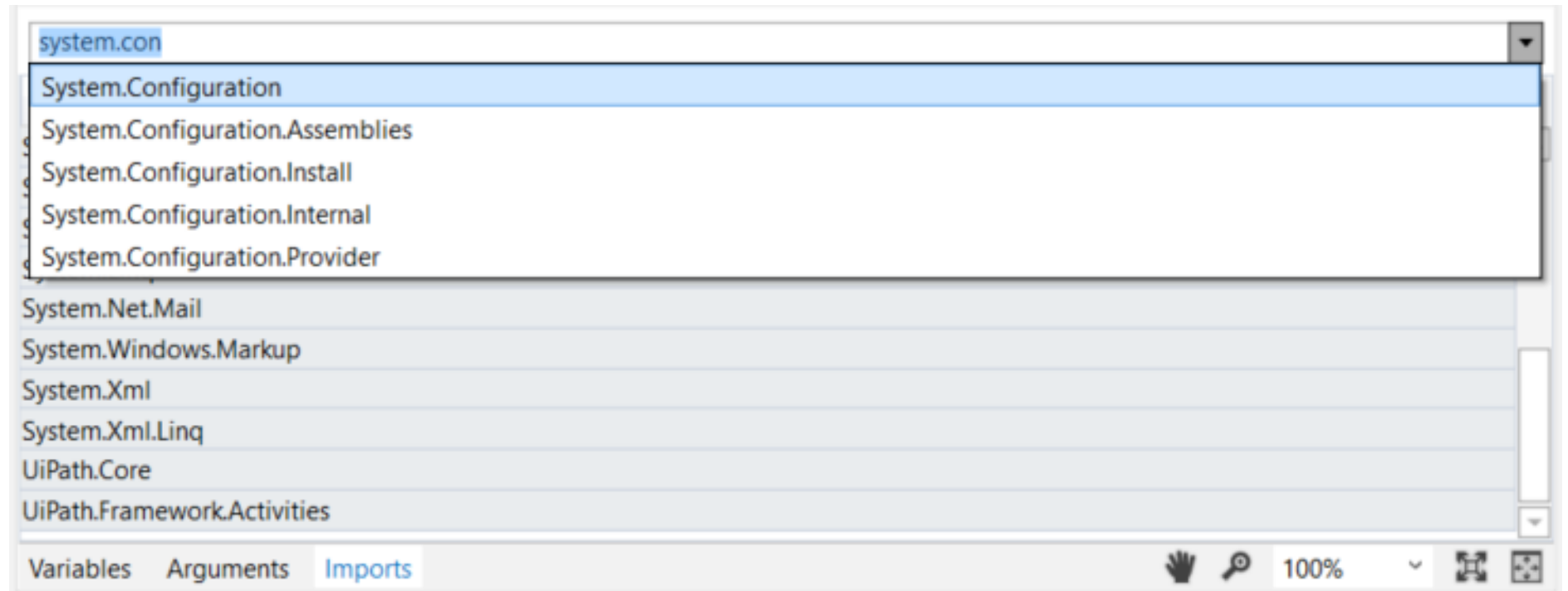
All imported namespaces are displayed in the Imports panel.



# Importing New Namespaces

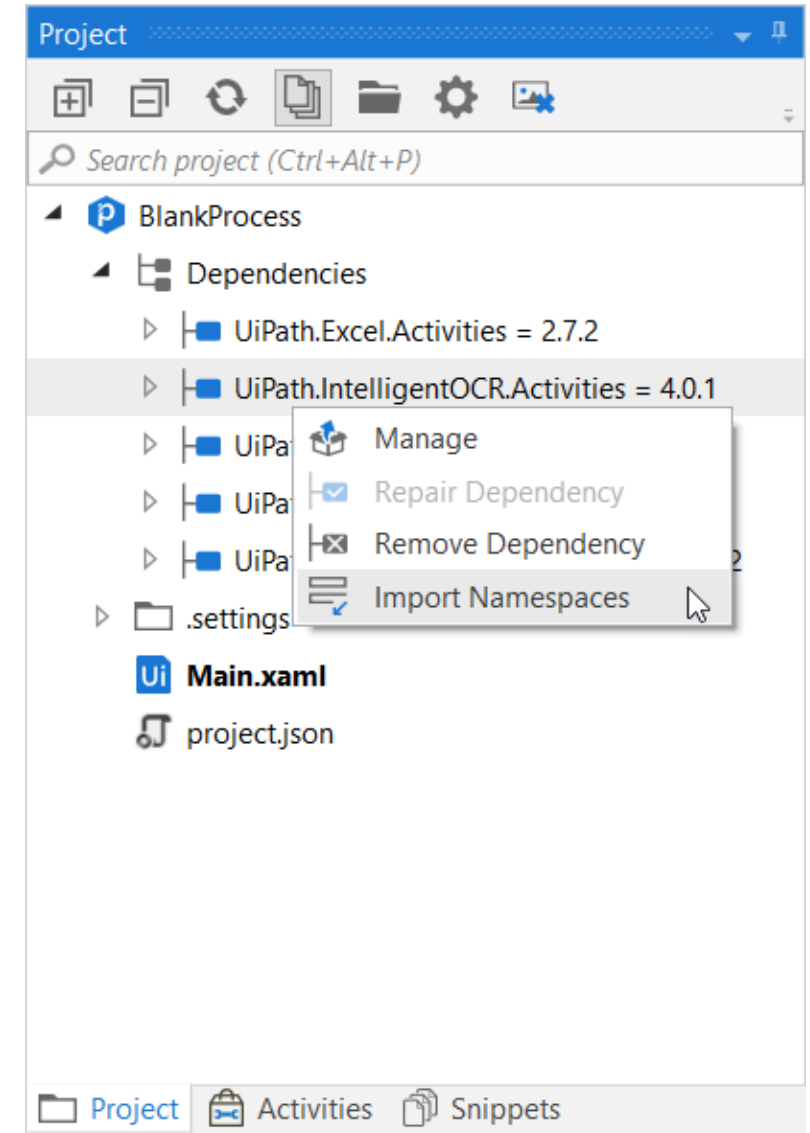
- **From the Imports Panel**

1. Open the Imports panel.
2. In the Enter or Select namespace field, start typing the namespace that interest you.
3. (Optional) Click the drop-down arrow to view and browse all available namespaces.
4. Select the desired namespace. The namespace is added to the Imported Namespaces list.



# Importing New Namespaces

- **From the Project Panel**
- Whenever you add a web service to a library project or install a new dependency to your project, either a .nupkg file from Manage Packages default feeds or a custom library that you've created, the namespace can be imported from the Project panel as well.
- Simply right-click the dependency or web service in the tree, and select Import Namespaces. The namespaces are automatically added to your project, you can check them in the Imports panel.



# Importing New Namespaces

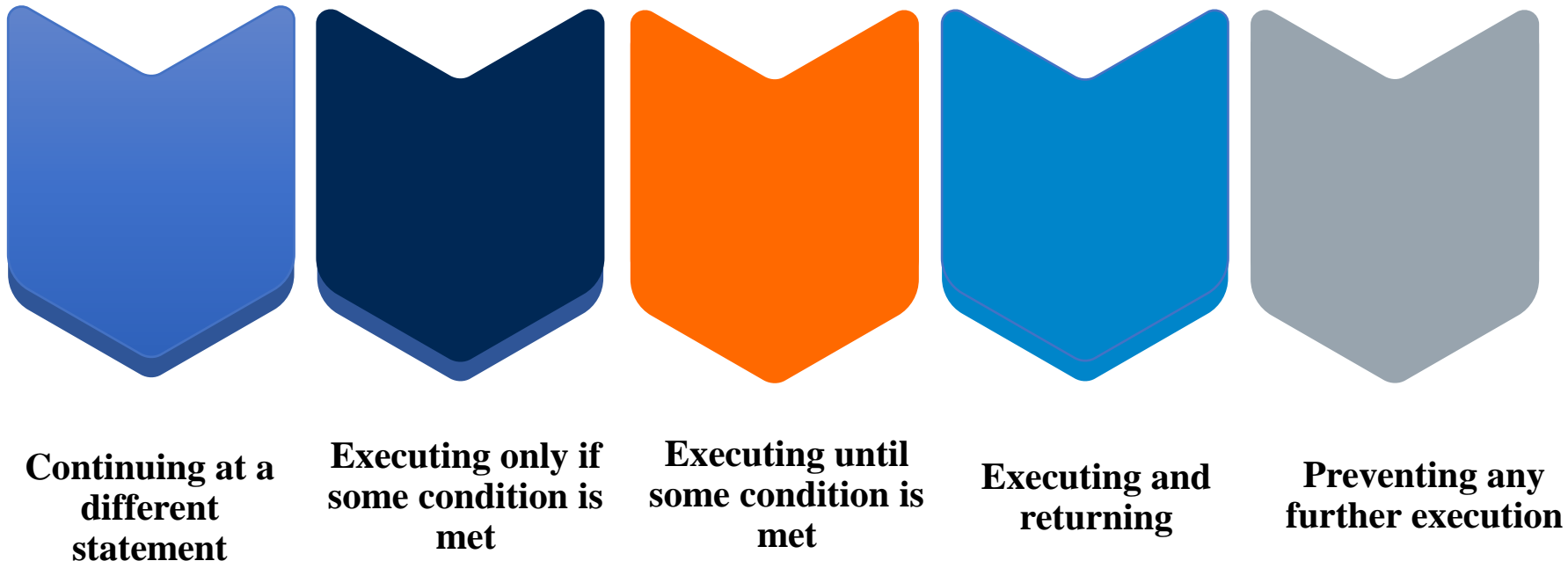
- **Removing Namespaces**

- To remove a namespace, in the Imports panel, either right-click the namespace and select Delete, or select the namespace and press the Delete key.
- To remove all the namespaces that are not used anywhere in the current file, select Remove Unused > Imports in the Studio ribbon.
- Note that namespaces can only be removed if they aren't used.
- For example, you can remove a namespace from the project if the assembly that contains it is no longer referenced by the project.

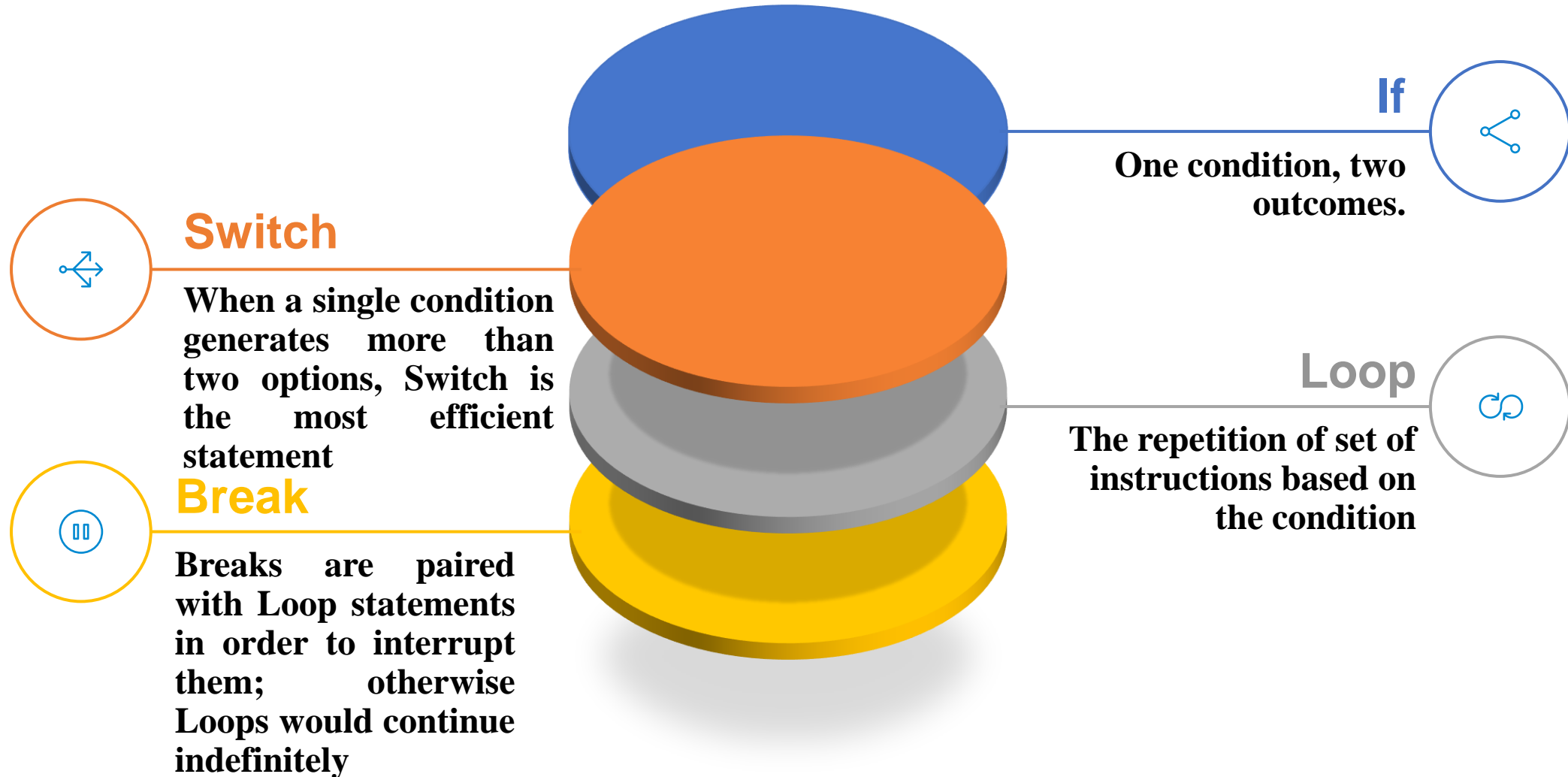
# Control Flow

# Control Flow

- The order in which individual statements, instructions or function calls are executed or evaluated in a software project. Control flow statements can be categorized by their effect:



# Basic Control Statements



If

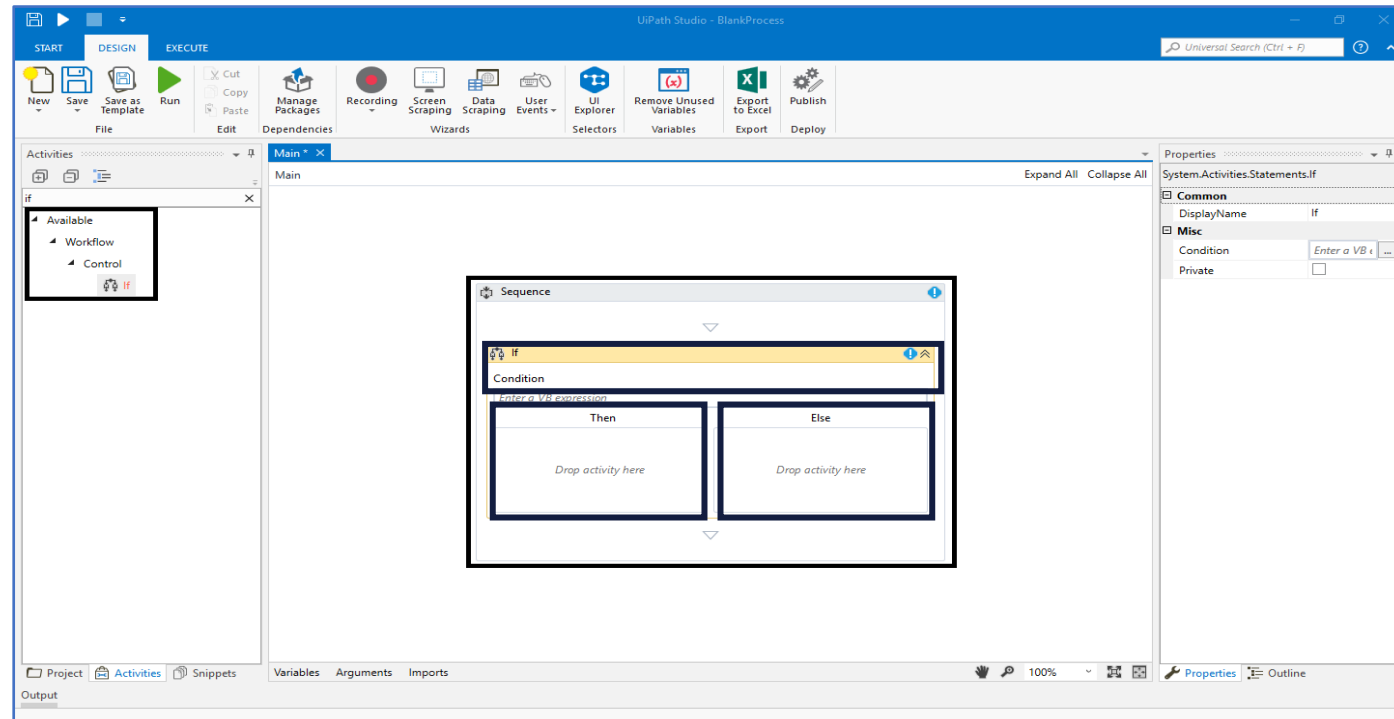
# The If Statement

The basic concept of If statement is a method of two activity (Then and Else) which contain two condition and one statement.

Switch

Loop

Break



If

# The Switch Statement

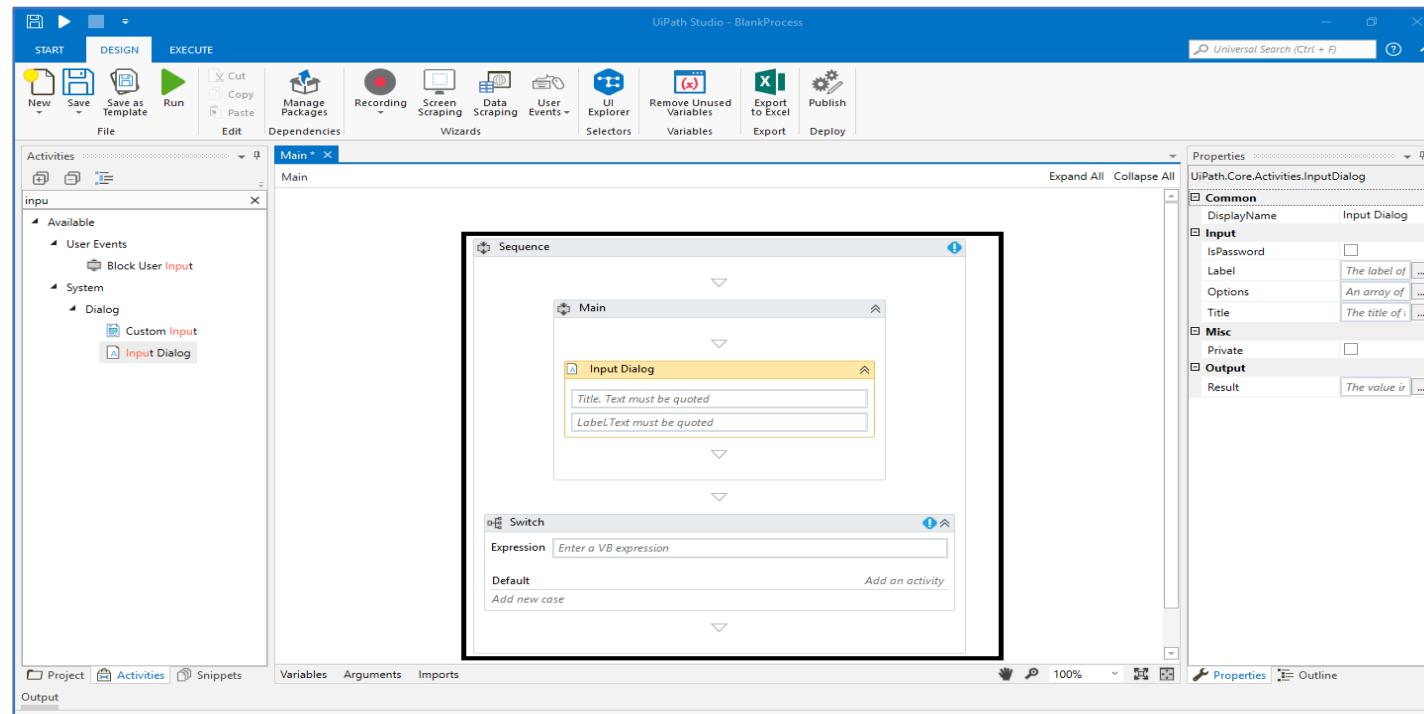
- The Switch statement allows one value out of multiple values by specified expression.

Switch

- Condition: It processes only integer argument values.
- Use: It is useful in the number of processes.
- Types of Switch statement: Structured and Unstructured

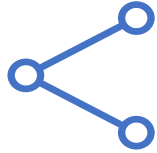
Loop

Break





# Comparison of If and Switch



**If**

```
IF V = 'Blue' THEN print 'You must be very brave'  
ELSE (IF V = 'Green' THEN print 'You must be  
very generous'  
ELSE (IF V = 'Gray' THEN print 'You must be very  
wise'  
ELSE print 'You must be a god, because you don't  
have human eyes'))
```



**Switch**

**SWITCH**

```
Case V = 'Blue' print 'You must be very brave'  
Case V = 'Green' print 'You must be very  
generous'  
Case V = 'Gray' print 'You must be very wise'  
Default Case print 'You must be a god, because  
you don't have human eyes'
```

If

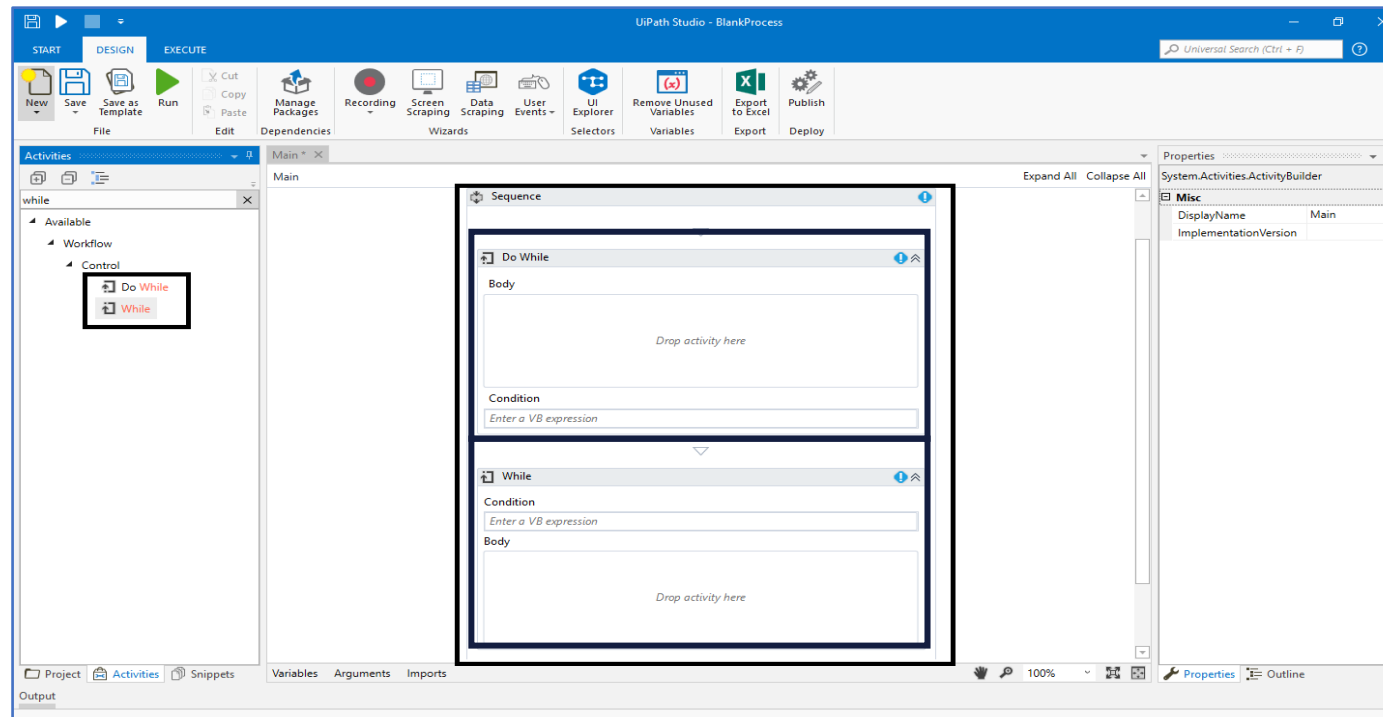
# The Loop Statement

- Loop is the structure that executes a repetitive set of operations with these low error activities.
  - Condition: It automates repeated tasks in UiPath through two loops statements.
  - Do While: If the condition is true, then in the execution process this activity runs first.
  - While: If the condition is false, then in the execution process this activity runs first.

Switch

Loop

Break



If

# The Break Statement

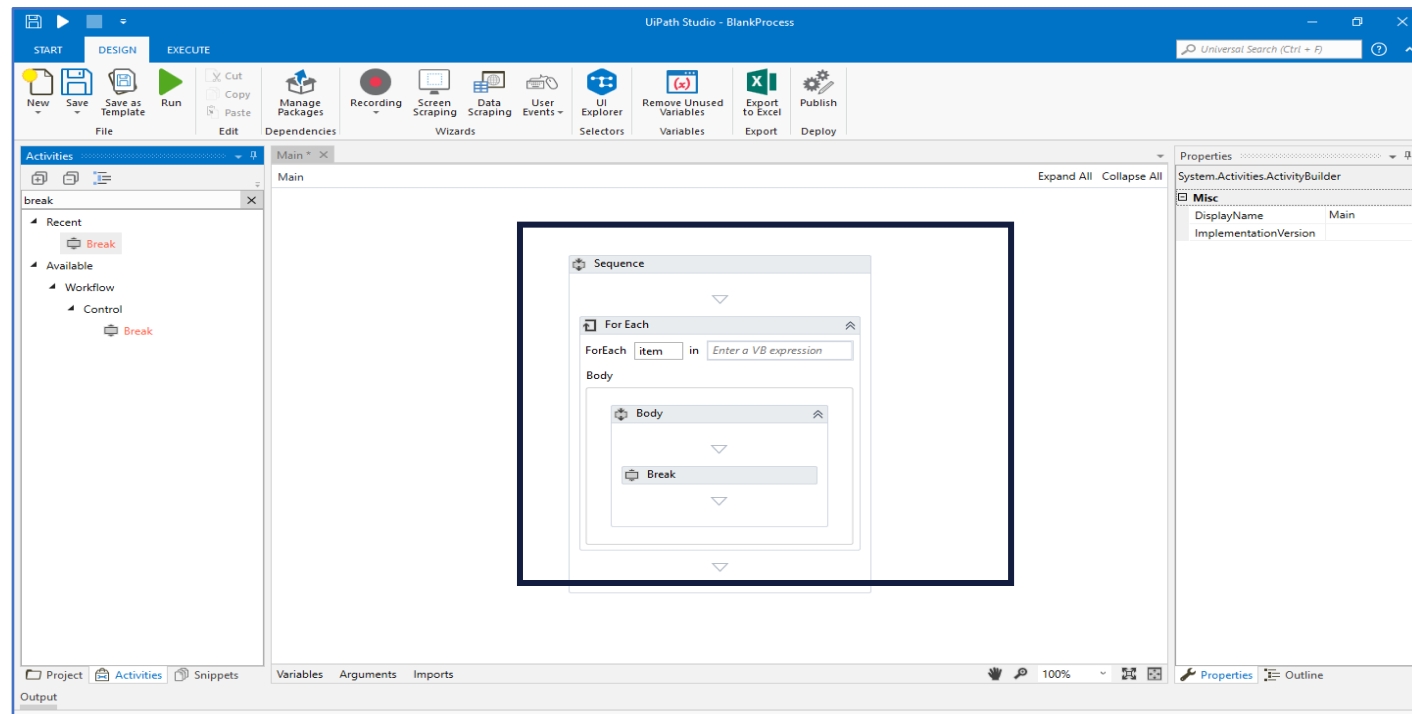
- It is the process in the UiPath Studio which allows to break an activity on the chosen or starting point.

Switch

- Condition: It exits each activity and continues the workflow process activity.
- Switch or Loop statement: Break statement used for loop termination and transfer the statement in the “Switch or Loop statement”.
- Use: It is used in relation with a Loop, to interrupt it and continue the execution outside it.

Loop

Break



# The Control Flow Statements in UiPath



Assign



Delay



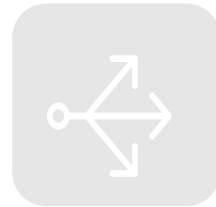
Do While



For Each



While



Switch



If



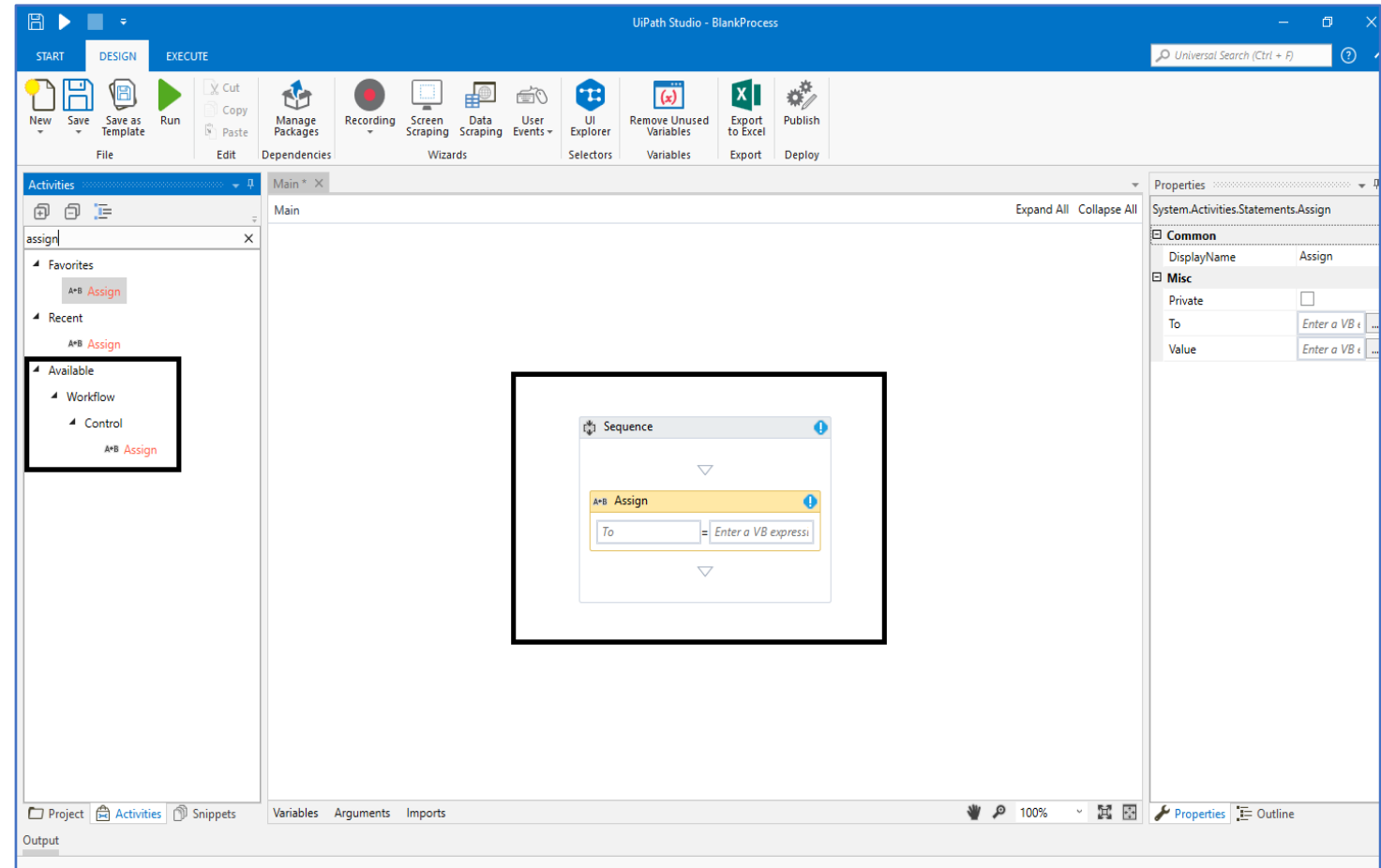
Break

# The Assign Statement

The Assign statement allocates a value to a variable or argument.

## What it can be used for?

- **Increment** the value of a variable in a loop
- **Sum** up two or more variables and assign the result to a different variable
- **Assign** values to an array



# The Delay Statement

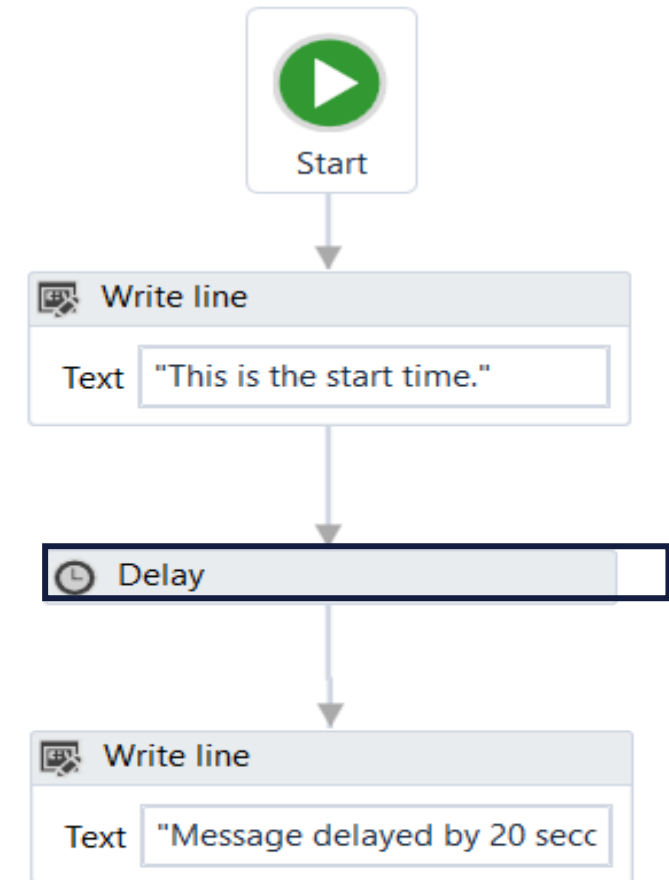
- The Delay statement pauses an automation for a period of time (hh:mm:ss).
- Useful in projects that require good timing, such as waiting for a specific application to start or waiting for some information to be processed.

## What it can be used for?

- **Machine Latency:** Delay is used to solve this issue which lead to error.

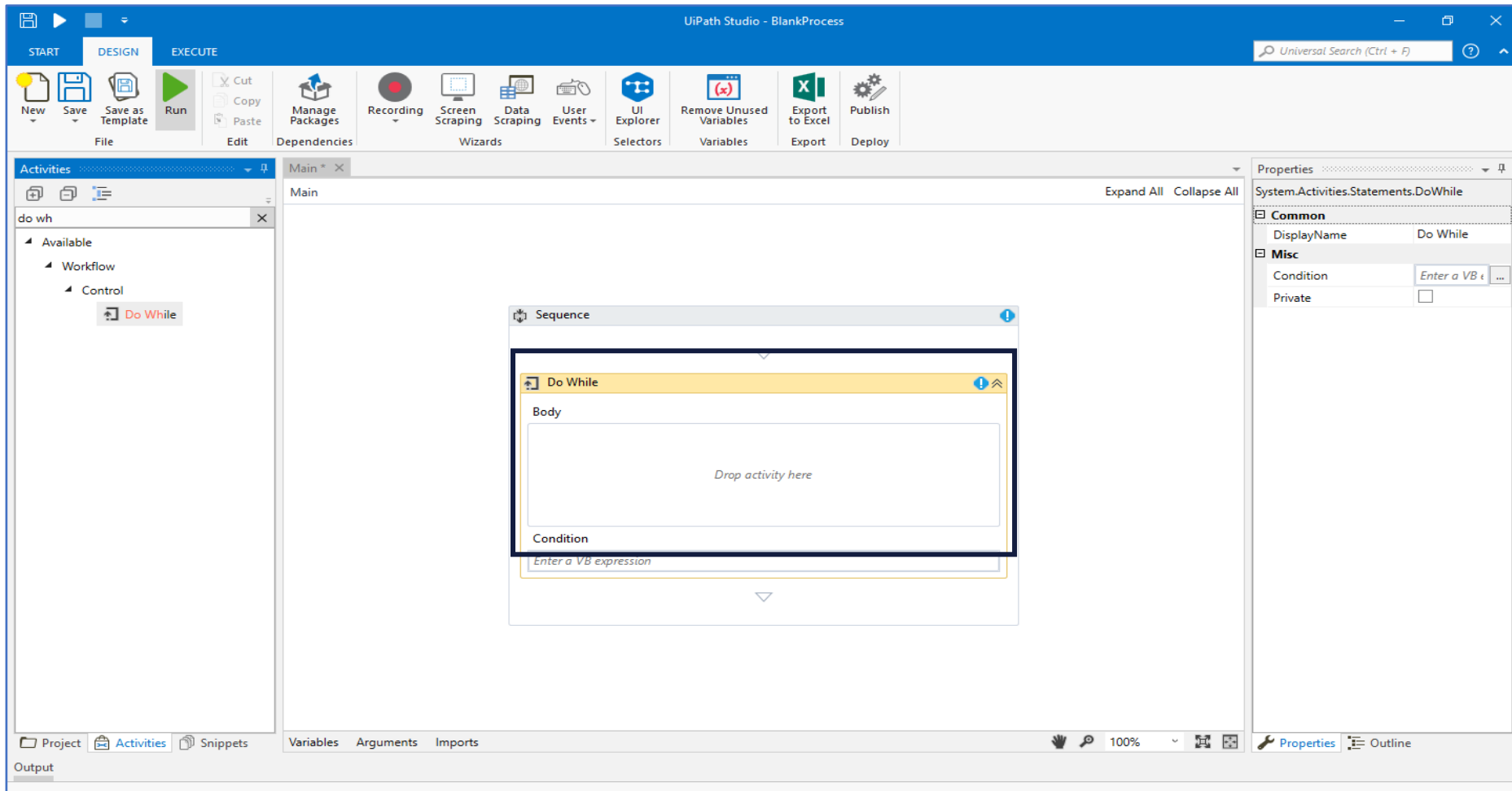
## What are its types?

- **Static Delay:** A pause which is fixed and has a tendency of failure.
- **Dynamic Delay:** Advanced form of static delay in which the conditions regulate the wait or pause time.



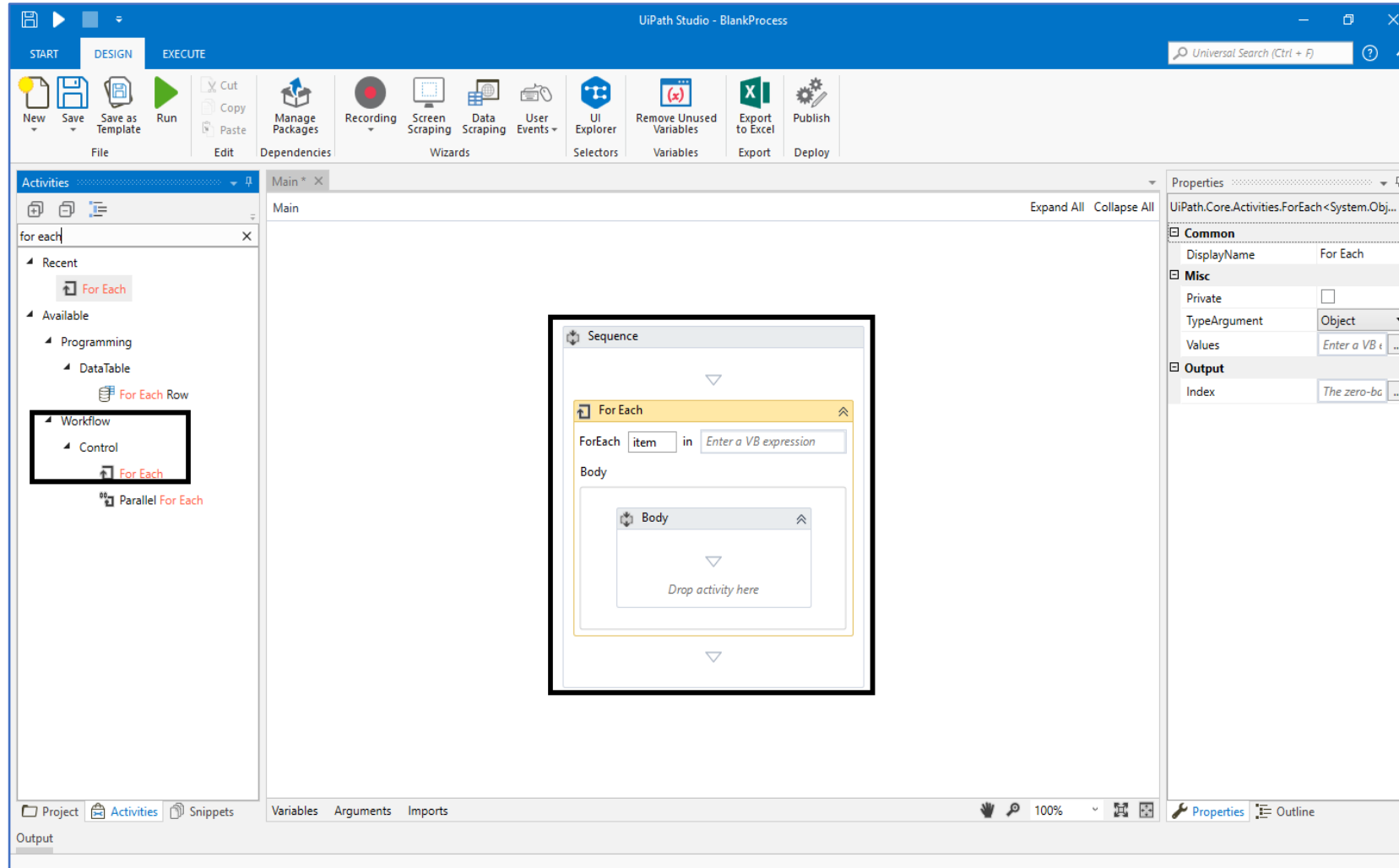
# The Do While Statement

- The Do While statement creates a loop that executes a specific sequence while a condition is met. The condition is evaluated after each execution of the statement



# The For Each Statement

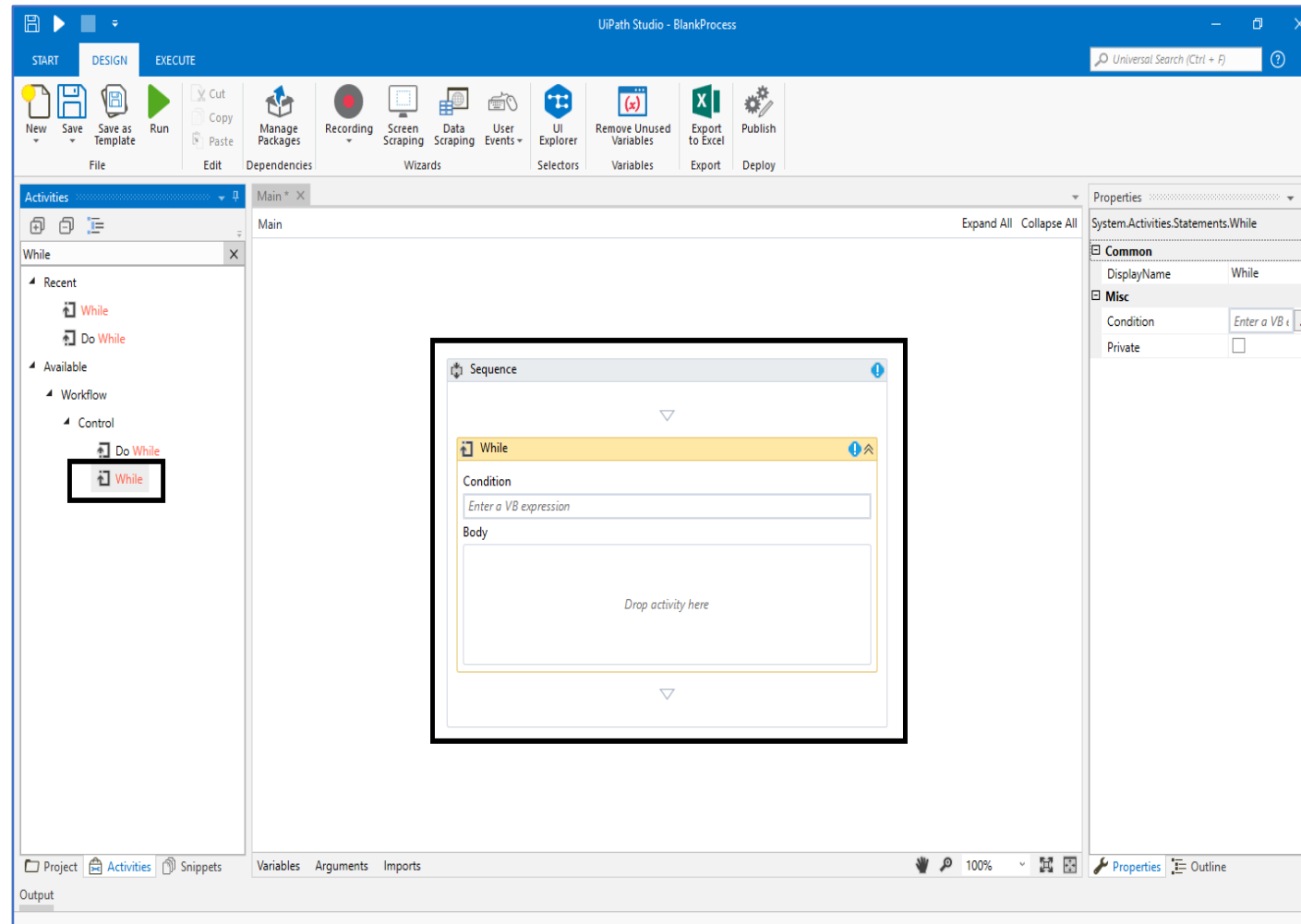
- The For Each statement performs an activity or a series of activities on each element of a collection.





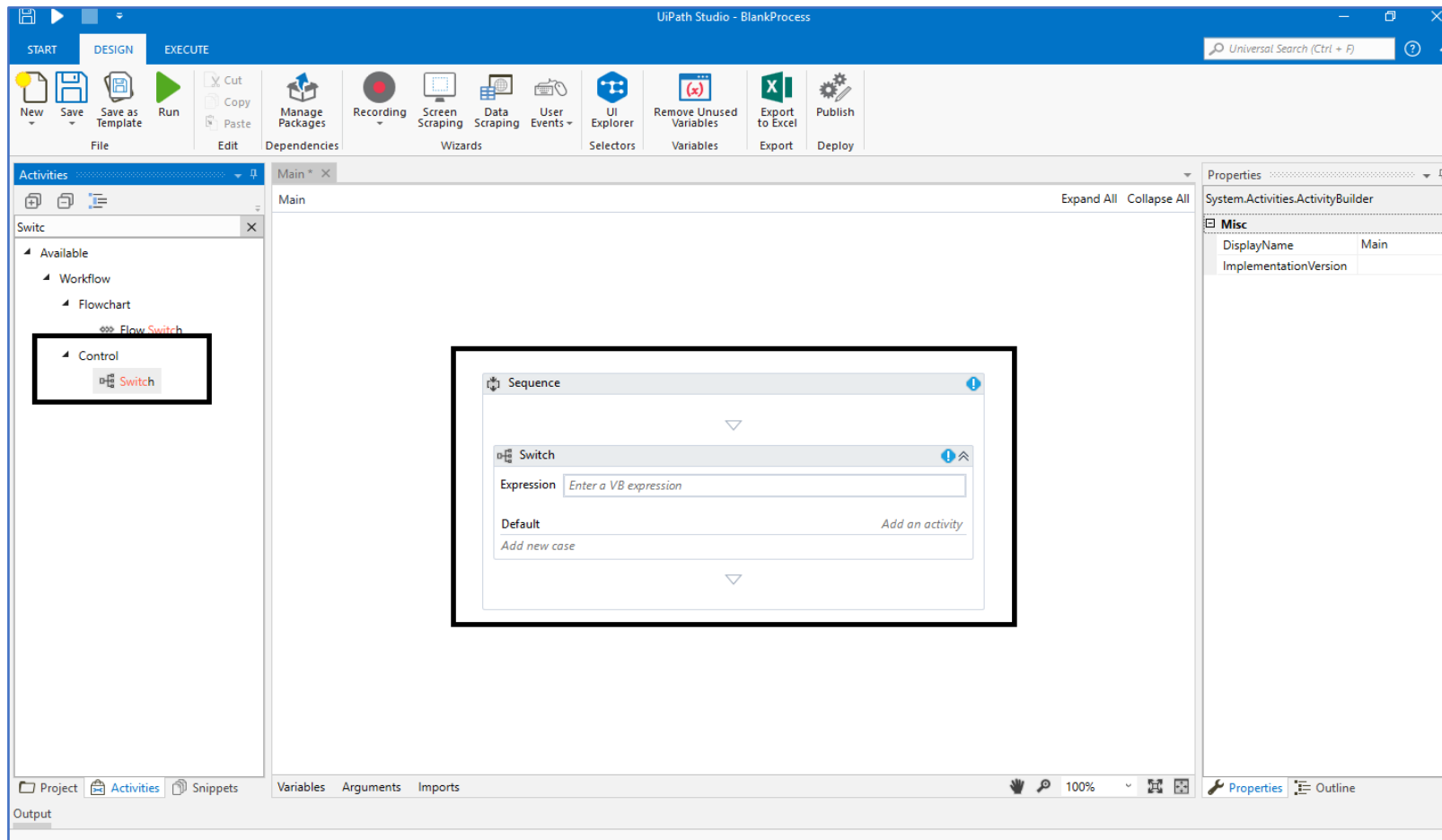
# The While Statement

- The **While** statement creates a loop that executes a specific sequence and the condition is evaluated before the execution of each statement.



# The Switch Statement

- The **Switch** statement executes a set of statements out of multiple statements, based on the value of a specific expression.

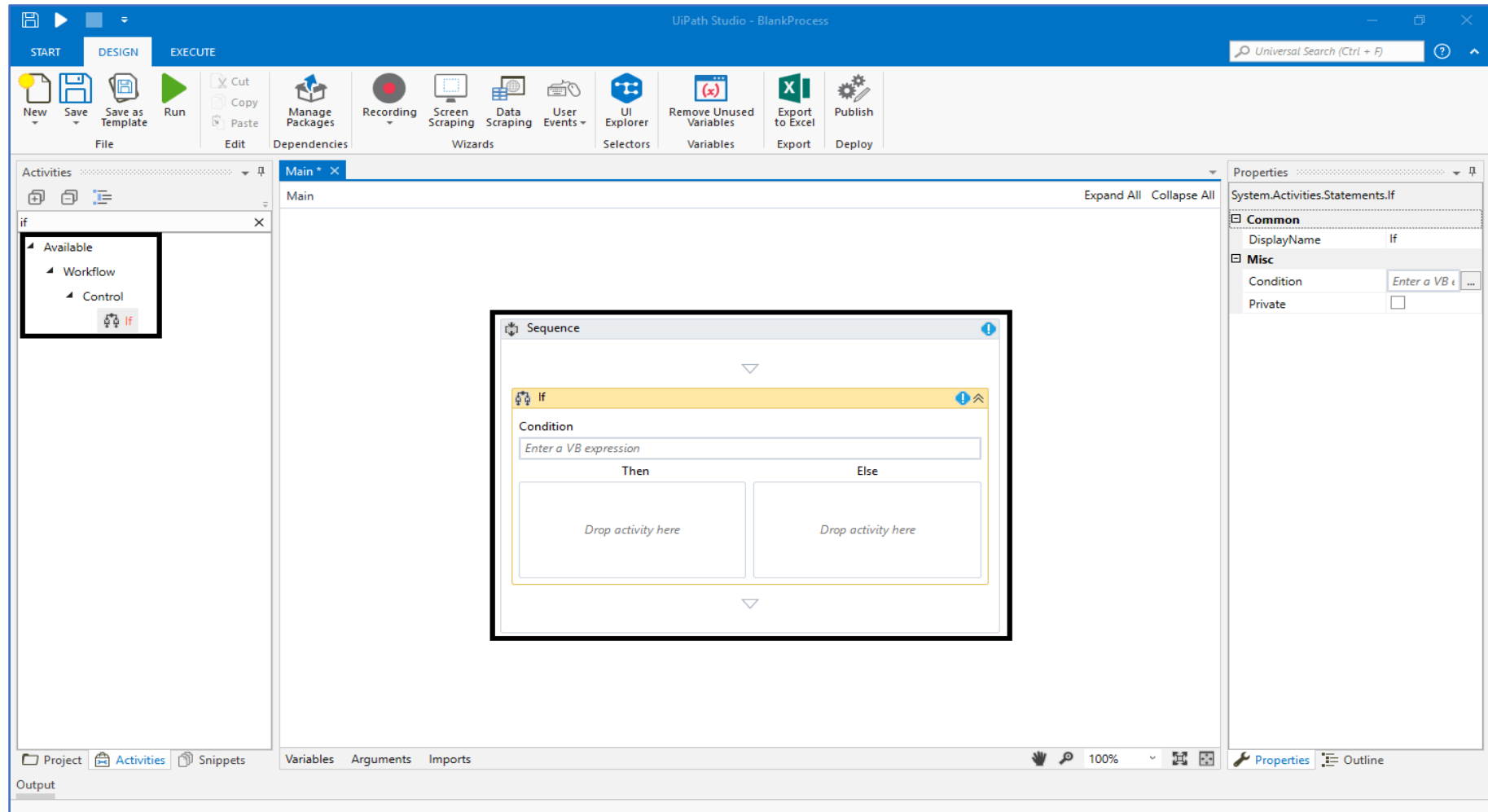


# The Switch Statement

- The Switch statement in UiPath is used when there are more than 2 cases that can be executed based on the matching of the condition (in UiPath, this is called Expression).
- A specific feature that can be used in UiPath is what we call Default Case – when none of the defined cases is matched against the expression, the default case is executed.
- This is optional and can be replaced by a specific set of instructions to be executed when the pre-defined cases are not matched, or can even be left blank.

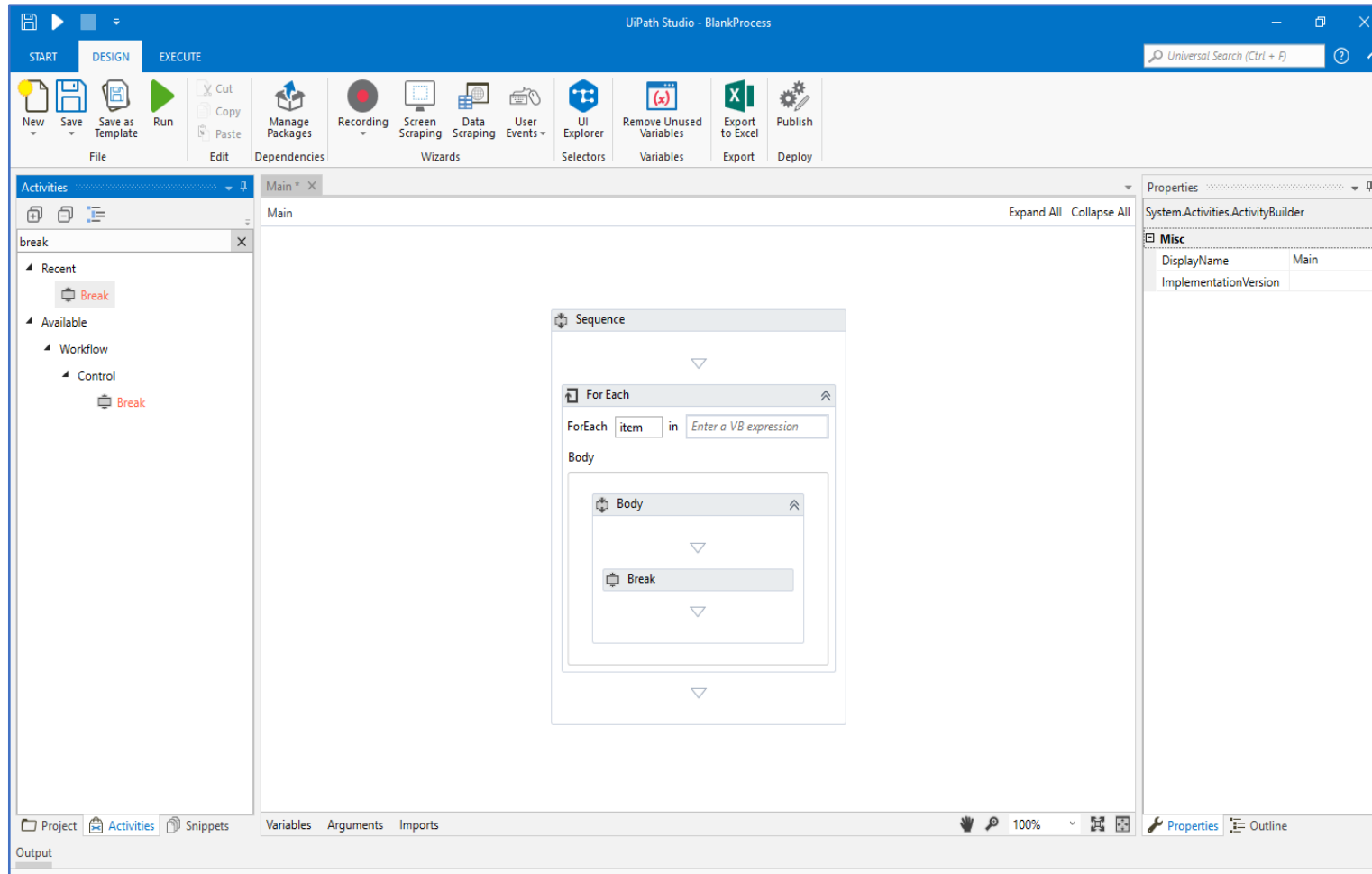
# The If Statement

- The If statement enables a project to take one of two different courses of action, depending on whether a specified condition is met.



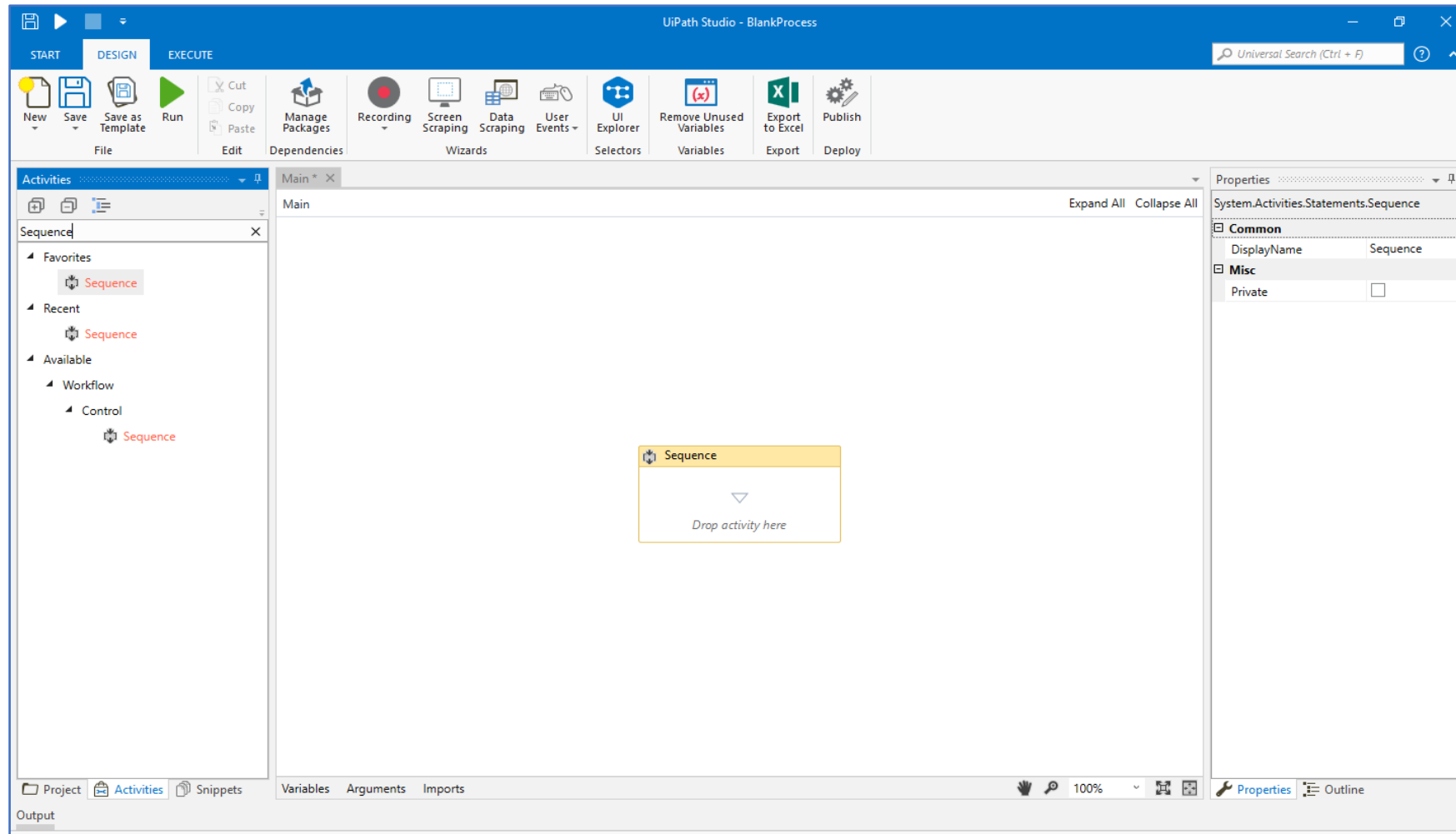
# The Break Statement

- The **Break** statement stops the loop at a chosen point and continues with the next activity.



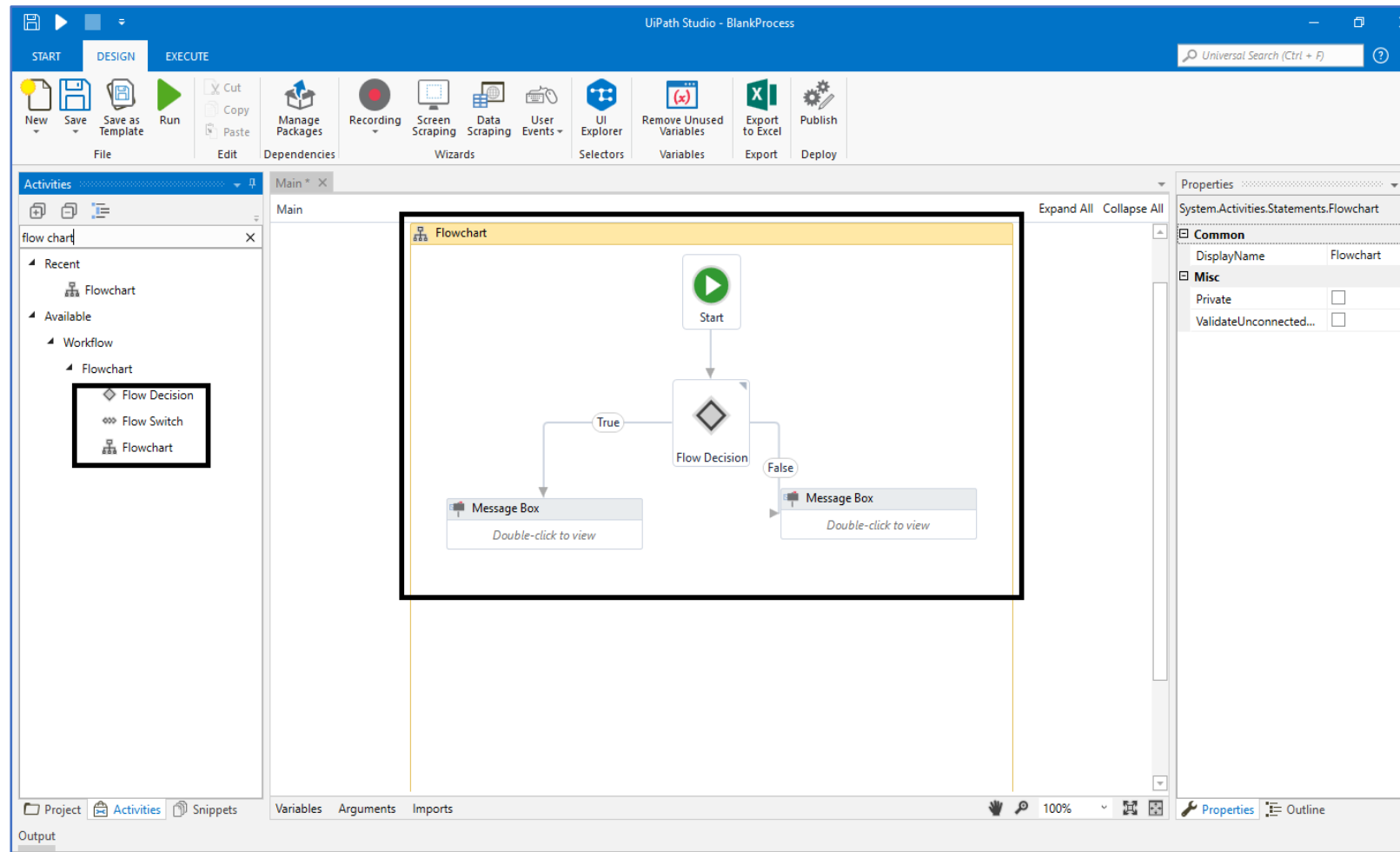
# Sequence

- **Sequence** is the smallest project to create a process in UiPath that enables to create a linear process in various activities and execute the sequential order.



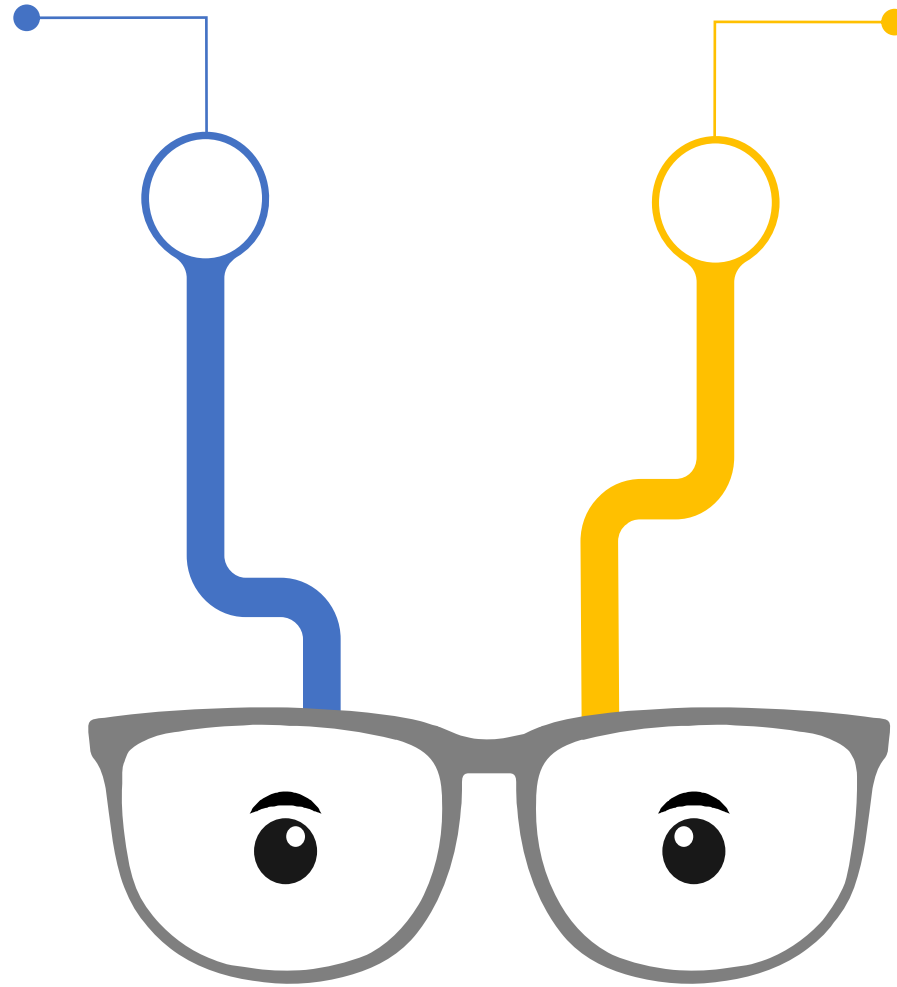
# Flowchart

The Flowchart represents various steps involved in completing activities, task, and process.



# Control Flow and Universal Statements

Control Flow is the order in which individual statements, instructions or function calls are executed or evaluated in a software project.



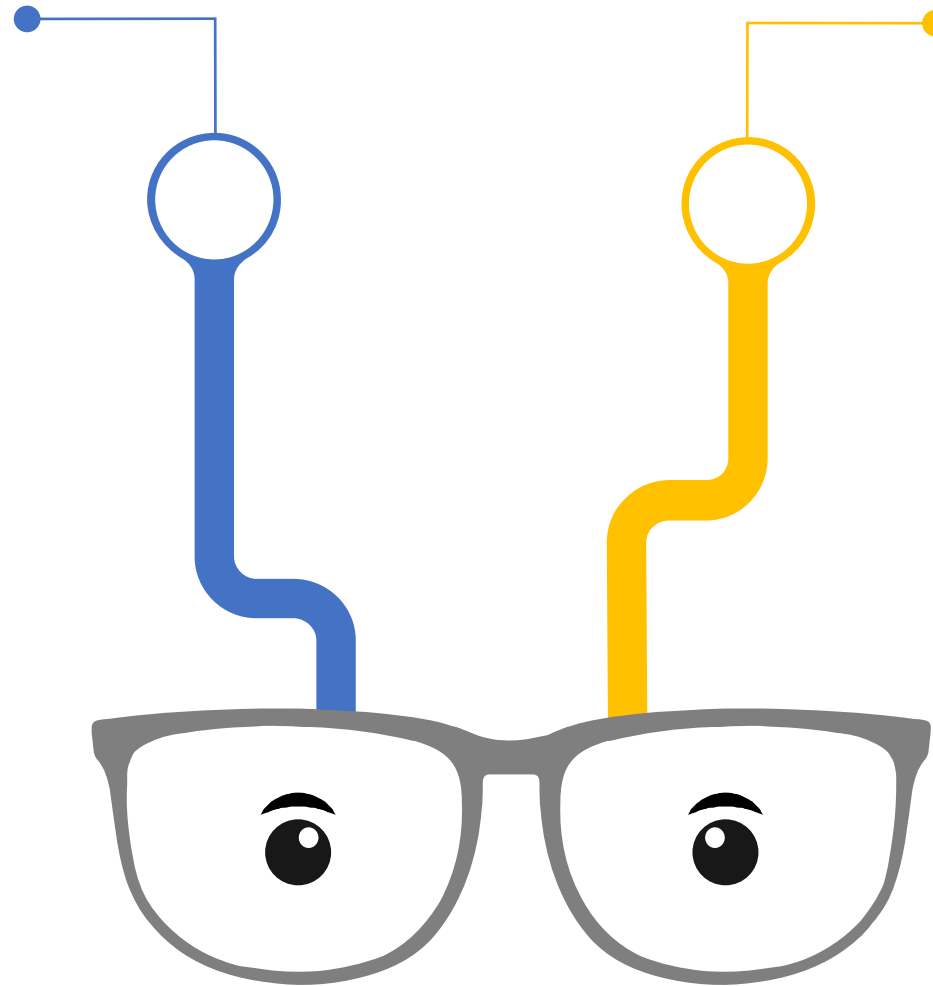
There are four basic control statements:

- If
- Switch
- Loop
- Break



# Control Flow Statements in UiPath

There are eight different types of control flow statement that are used in UiPath.



UiPath Control Flow Statements:

- Assign
- Break
- Delay
- Do While
- For Each
- If
- Switch
- While

**Thank You**