

# LECTURE 02B. INTRODUCTION TO UIPATH STUDIO

---

**Robotic Process Automation**  
**[10 October 2022]**

Elective Course, 2022-2023, Fall Semester

Camelia Chisăliță-Crețu, Lecturer PhD  
Babeș-Bolyai University

# Acknowledgements

This course is presented to our Faculty with the support of UiPath Romania.



# Contents

- Automation Project
  - Definition. Types. Structure
- User Interface
  - Ribbon. Panels
- Variables
  - Data Types: Integer, String, Boolean, Generic, Array of T
- Choices
  - If Activity, Flow Decision Activity, If Operator, Switch Activity, Flow Switch Activity
- Demo 5
- Control Flow Activities
  - For Each, While, Do While
- Demo 7
- References

# Automation Project. Definition

- An **activity** is
  - the smallest action in UiPath;
  - a **step** in a process workflow;
- An **automation project** is
  - A **set of steps** that allows to perform a meaningful task;
  - a graphical representation of the business **process**;
- it allows to automate a rule-based process, formed by custom set of steps;
- E.g.:
  - Click on a button;
  - Read a file;
  - Write to a log file.

# Automation Project. Types

- Types of supported projects
  - **Sequences** - for linear processes;
    - it connects one activity to another without cluttering the project;
    - *when to use*: simple scenarios, activities follows one after another;
    - easy to assemble and understand;
  - **Flowcharts** - for more complex processes;
    - it integrates decisions and connects activities in a more diverse manner through multiple branching and logic operators;
    - it provides a two dimensional view of the workflow;
    - *when to use*: to show decision points in a process, visual appealing;
    - *cons*: prone to chaotic interweaving of activities;
  - **State machine** - for very large projects;
    - it applies to projects that use a finite number of states during execution which are triggered by a condition or an activity;
    - *when to use*: to represent standard high-level process diagram of transactional business process templates.

see **Demo1 - FirstProject**

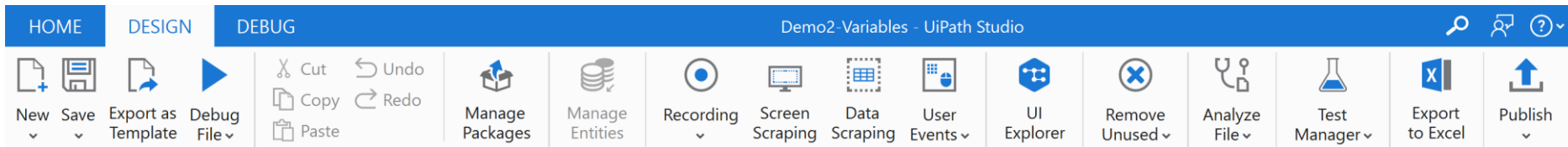
# Automation Project. Structure

- by default:
  - **Main.xaml file** – it consists of the main flow;
    - a sequence or a flowchart can be initially added;
    - other .xaml files may be added;
    - at run time this file will be executed only ==> all other .xaml files should be connected in **Main.xaml** through the **Invoke Workflow File** activity;
    - an .xaml file can be set as *main module* by choosing the **Set as Main** option in the *pop-up menu*;
  - **.screenshots folder** – it is generated if the project uses UI automation;
    - to save the screenshot;
  - **project.json** – it contains details of the automation project.

see **Demo1 - FirstProject**

# The User Interface. Top Ribbon

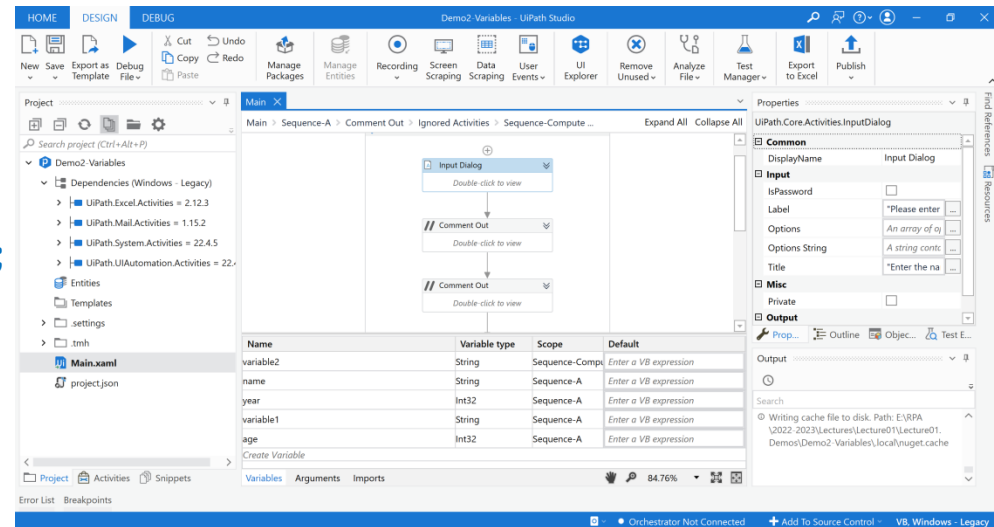
- There are 3 menus on the top ribbon:
  - **Start/Home** - to create a new project, i.e., a new process;
    - It Connects one activity to another without cluttering the project;
  - **Design** - to design the process;
    - Actions allowed: add activities (sequences, flowcharts, state machine), UI interaction, export to Excel, publish to Orchestrator;
  - **Execute/Debug** - debug related actions;
    - Actions allowed: validate, run, debug, monitor the execution step by step;



see **Demo1 - FirstProject**

# The User Interface. Panels

- Main areas (panels) in UiPath Studio:
  - design time:
    - **Project, Activities, Snippets;**
    - **Designer;**
    - **Variables, Arguments, Imports;**
    - **Properties Panel, Outline Panel;**
  - run/debug time:
    - **Output Panel, Locals Panel;**
    - **Error List, Breakpoints.**

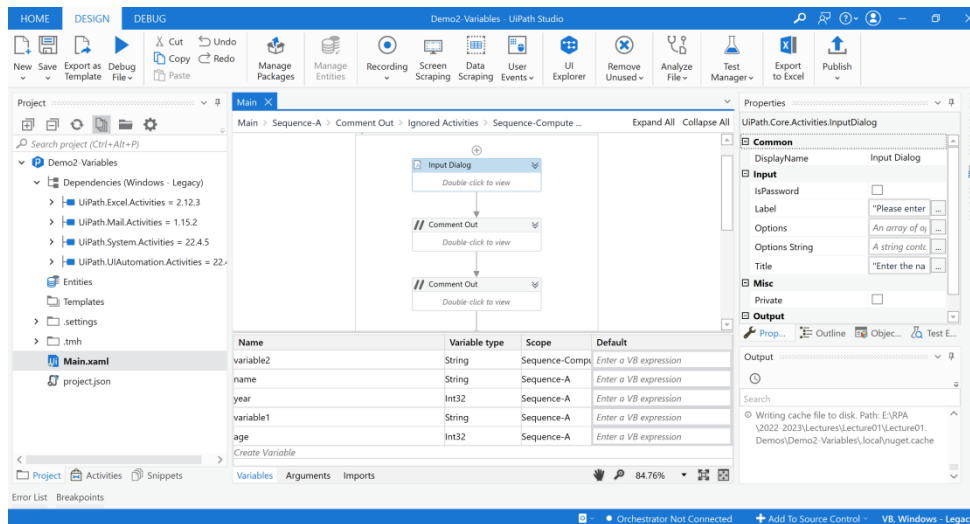


see Demo1 – FirstProject, Demo2 – Variables



# Demo 1 - FirstProject

- Tasks:
  - Create a simple project of type **Process**;
  - Work with **Sequence** and **Flowchart** containers.
  - Read and write some data (numeric, text);
    - work with **Input Dialog** and **Write Line** and **Message Box** activities;
    - **Input Dialog** activity versatility.



see Demo1 – FirstProject

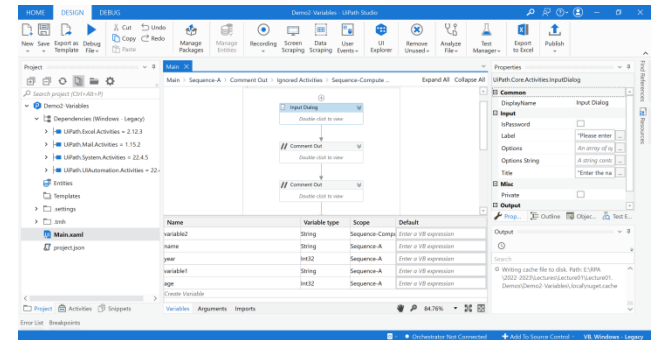
# Variables. Data Types

- **Variables** are used to store different types of data: numeric, text, image, file, colour;
- main types of variables:
  - **Int32**;
  - **String** – with quotes, e.g., “abc”, “123”;
  - **Boolean** = {True, False};
  - **GenericValue** – almost any data type;
  - **Array of [T]** – all values have the same type;
- a variable defined within a scope (e.g., **Sequence**, **Flowchart**) is available in all scopes included in it;
- **Variable Panel** shows the properties of the defined variables:
  - **Name, Type, Scope, Default value**;
  - it presents the variables available in the selected activity.
- VB. Net operators used: **Not** for !; **<>** for !=; **And** for &; **Or** for |; **=** for ==; **Mod** for % (see [https://en.wikipedia.org/wiki/Comparison\\_of\\_C\\_Sharp\\_and\\_Visual\\_Basic\\_.NET\\_Comparers](https://en.wikipedia.org/wiki/Comparison_of_C_Sharp_and_Visual_Basic_.NET_Comparers)).

see **Demo2 - Variables**

# Demo 2 - Variables

- Tasks:
  - Compute the age in years;
    - variables of type String and Int32;
    - work predefined objects and properties, e.g., property Now from class **System**;
- Identifiers
  - duplicated identifiers
  - not case sensitiveness in UiPath;
  - work with **Sequence** and **Flowchart** containers.
- Generic variables
  - **GenericValue** data type, converted to **String** or **Int32**
    - $c=a+b$
    - $c=b+a$
    - different meanings for the + operator, according to the closest type of the first operand.



see **Demo2 – Variables**

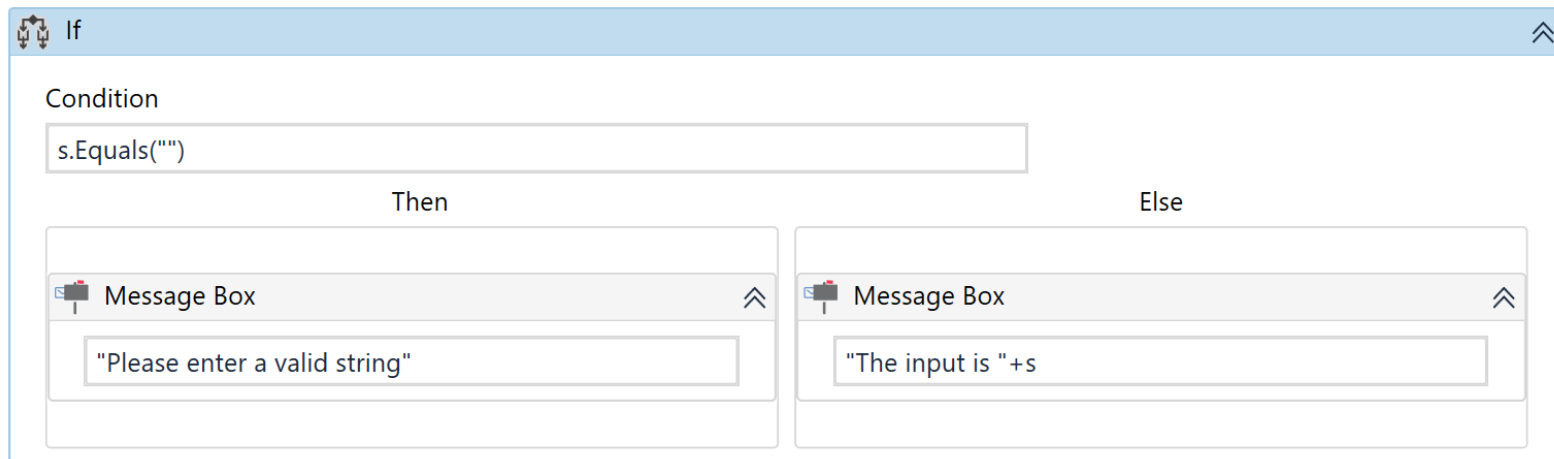
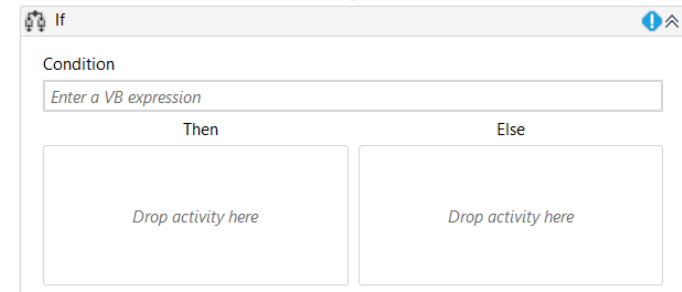
# Choices

Activity	Sequence	Flowchart	Assign
<b>If</b> activity	Yes	Yes	No
<b>Else If</b>	Yes	Yes	No
<b>Flow Decision</b>	No	Yes, similar to <b>Else If</b> activity in a <b>Sequence</b>	No
<b>If</b> operator (VB)	No	No	Yes, similar to <b>If</b> activity in a <b>Sequence</b>
<b>?:</b> operator (C#)	No	No	Yes, similar to <b>If</b> activity in a <b>Sequence</b>
<b>Switch</b>		Yes	No
<b>Flow Switch</b>	No	Yes	No

see Demo3 – Choices (VB, C#), Demo4 – IfOperator (VB)

# Choices. If Activity

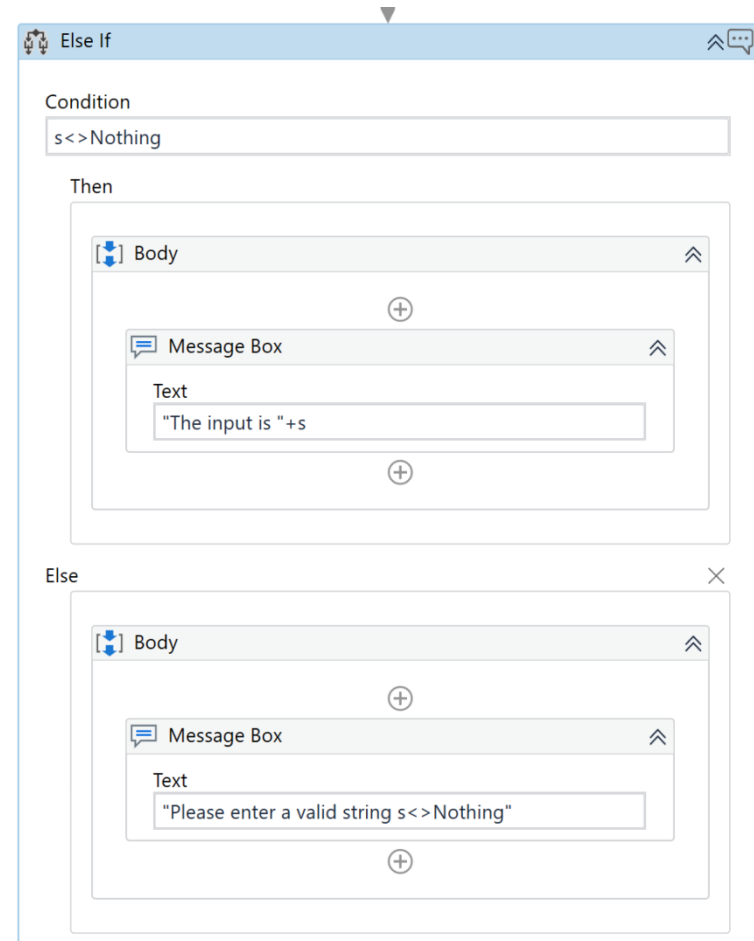
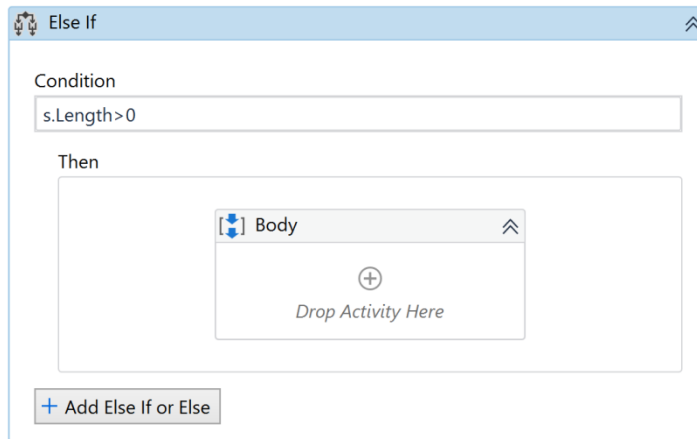
- **if** activity:
  - it splits the sequence vertically;
  - adequate for short linear branches;
- *cons*:
  - more than one *if else if* chained affects perception on the screen;



see Demo3 – Choices (VB, C#)

# Choices. Else If Activity

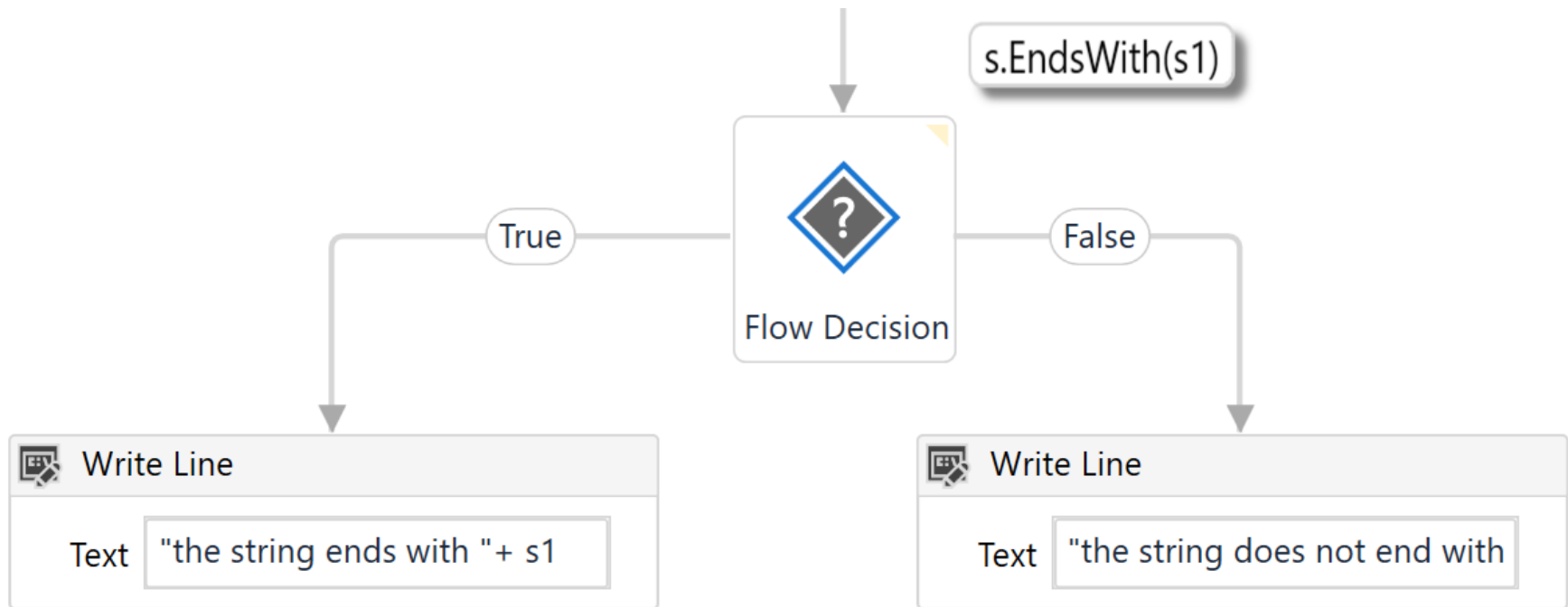
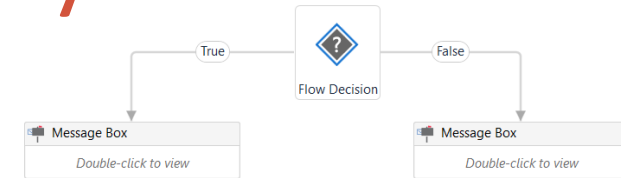
- **Else If** activity:
  - suitable for cases when the project takes different courses of action, depending on whether a *series of specific conditions are met*;
  - the **Else** or **Else If** condition is **optional**;



see Demo3 – Choices (VB, C#)

# Choices. Flow Decision Activity

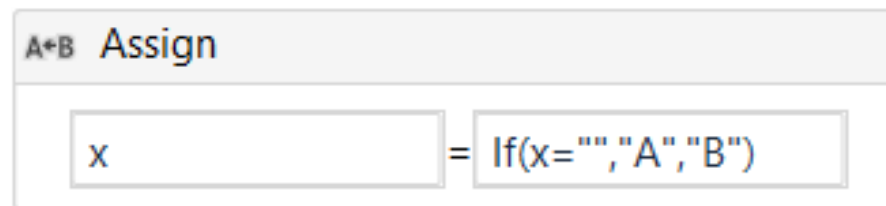
- **Flow Decision** activity:
  - it shows important decision logic and related conditions;
  - **Flow Decision** activity in *flowcharts* = **If** or **Else If** activity in *sequences*;



see Demo3 – Choices (VB, C#)

# Choices. If Operator

- **If** operator:
  - this is a VB operator;
  - useful for **small local conditions** or **data computations**;
  - it reduces the block to a single **Assign** activity;

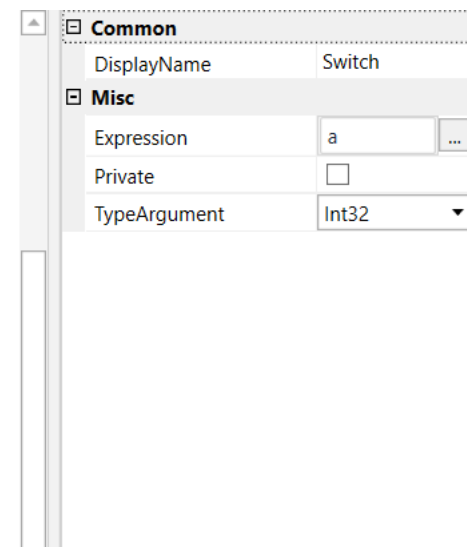
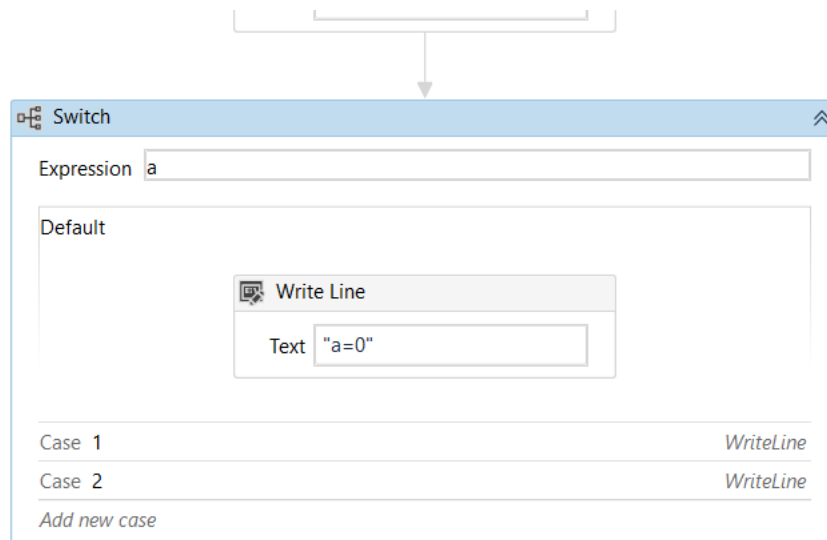


see Demo3 – Choices (VB), Demo4 – IfOperator (VB)



# Choices. Switch Activity

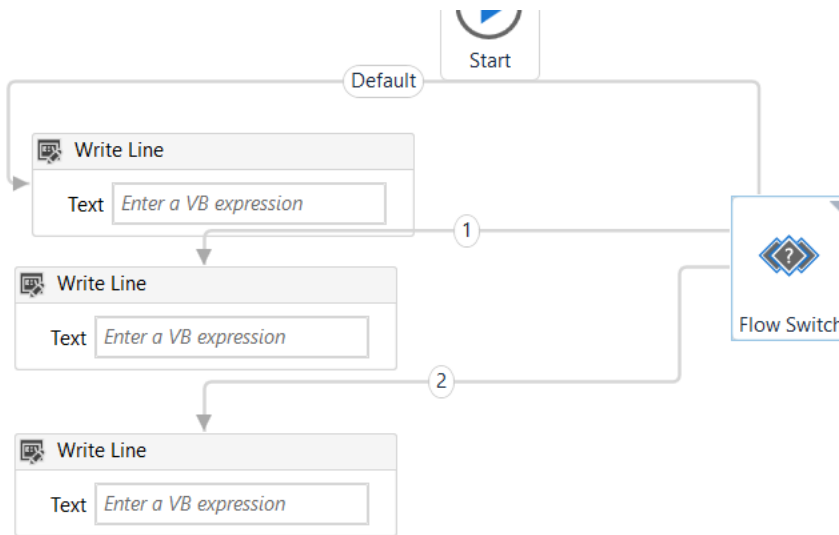
- **Switch** activity:
  - it can be used together with **If** operator; to streamline and compact *if else if* cascade, with distinct conditions and activities per branch;



see Demo3 – Choices (VB, C#)

# Choices. Flow Switch Activity

- **Flow Switch** activity:
  - an **If** activity that selects the next node depending on the value of expression;
  - **Flow Switch** activity in *flowcharts* = **Switch** activity in *sequences*;

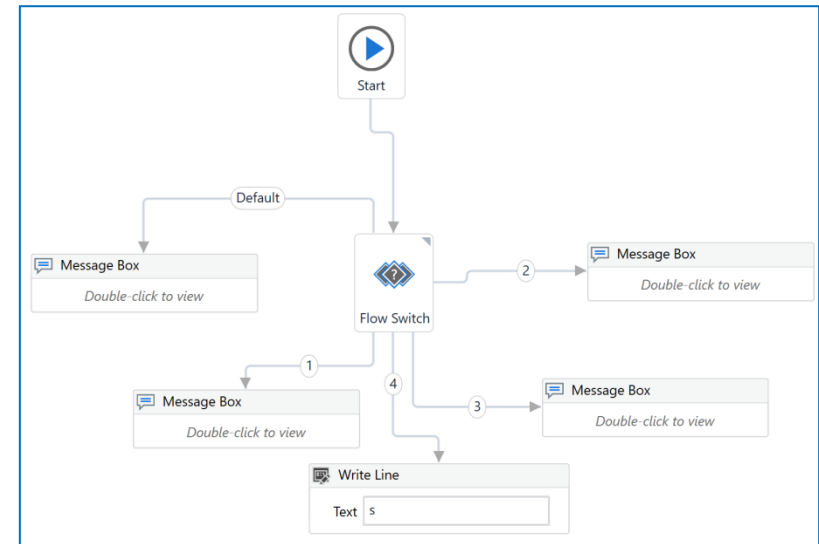


DisplayName	Flow Switch
Expression	a
TypeArgument	Int32

see Demo3 – Choices (VB, C#)

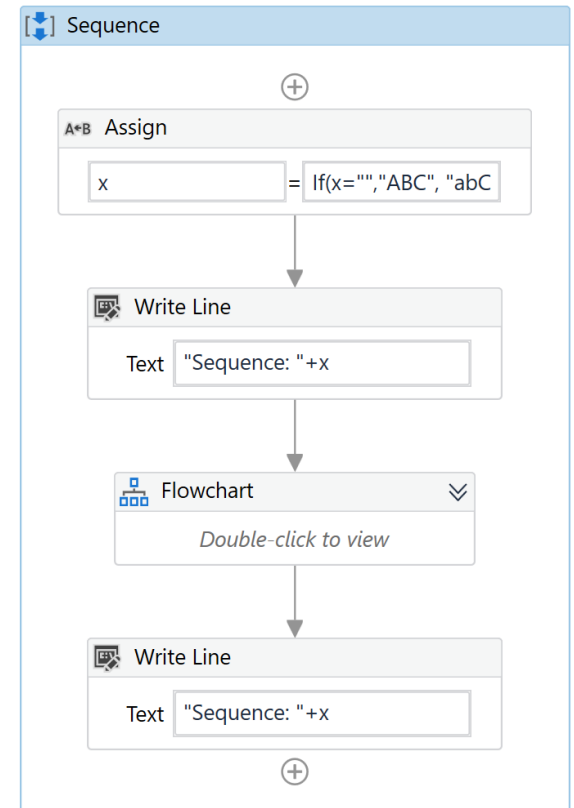
# Demo 3 - Choices

- Tasks:
  - **If** activity
    - checks whether a string equals to "";
  - **Else If** activity
    - checks whether a string is null;
  - **Flow Decision** activity
    - checks if a string ends with "T";
  - **If** operator (VB) or **?:** ternary operator (C#)
    - checks if a string ends with "T";
  - **Switch** activity
    - checks the value of some expression and acts accordingly
  - **Flow Switch** activity
    - checks the value of some a string related expression and acts accordingly



# Demo 4 – If Operator (VB)

- Tasks:
  - **If** operator (VB)
    - `x=If (x="", "ABC", "abC")`
  - working with duplicated identifiers in different scopes



Name	Variable type	Scope	Default
x	String	Flowchart	"XYZ"
y	String	Flowchart	Enter a VB expression
x	String	Sequence	"abc"

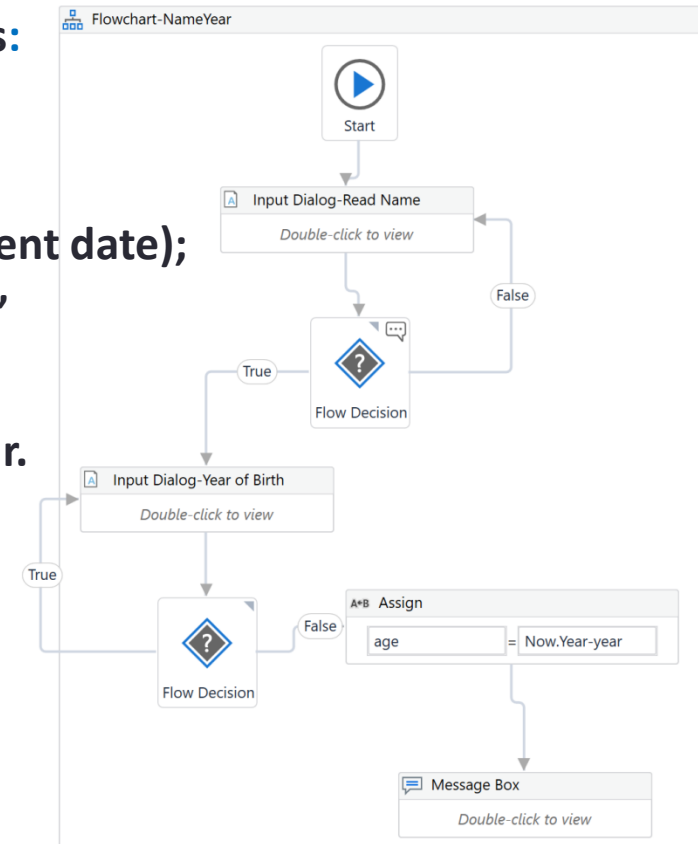
Create Variable

- ① Debug started for file: Main
- ① Demo4-IfOperator execution started
- ① Sequence: abC
- ① Flowchart: XYZ
- ① Sequence: abC
- ① Demo4-IfOperator execution ended in: 00:00:01

see Demo4 – IfOperator (VB)

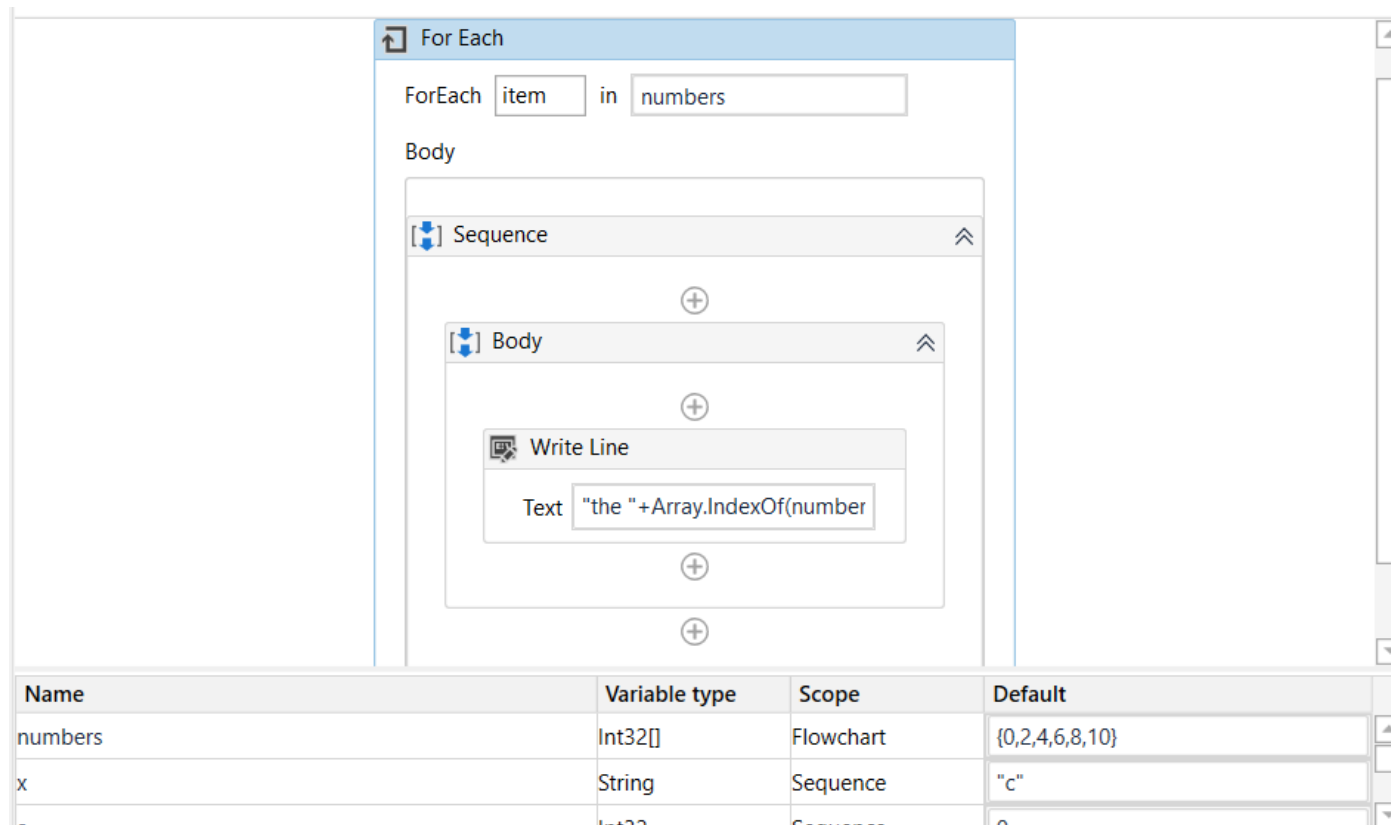
# Demo 5 – Name Age

- Create a process that performs the following actions:
    - 1. *read* the name of the person;
    - 2. *read* the birth year;
    - 3. *compute* the age in years (considering the current date);
    - 4. *print* “Congratulations, Z! You are x years old!”
  - Do not allow empty name and/or negative birth year.
    - *What is the easiest way to achieve it?*
- sequence/flowchart.*



# Control Flow. For Each Activity

- **For Each** activity:



Name	Variable type	Scope	Default
numbers	Int32[]	Flowchart	{0,2,4,6,8,10}
x	String	Sequence	"c"
	Int32	Sequence	0

see **Demo6 - ControlFlow**

# Control Flow. While Activity

- **While** activity:

**While**

Condition: `index < z.Length`

Body:

- Sequence**
  - Write Line**  
Text: `"item"+z(index).ToString`
  - Assign**  
`index = index+1`

**Common**

DisplayName	Assign
-------------	--------

**Misc**

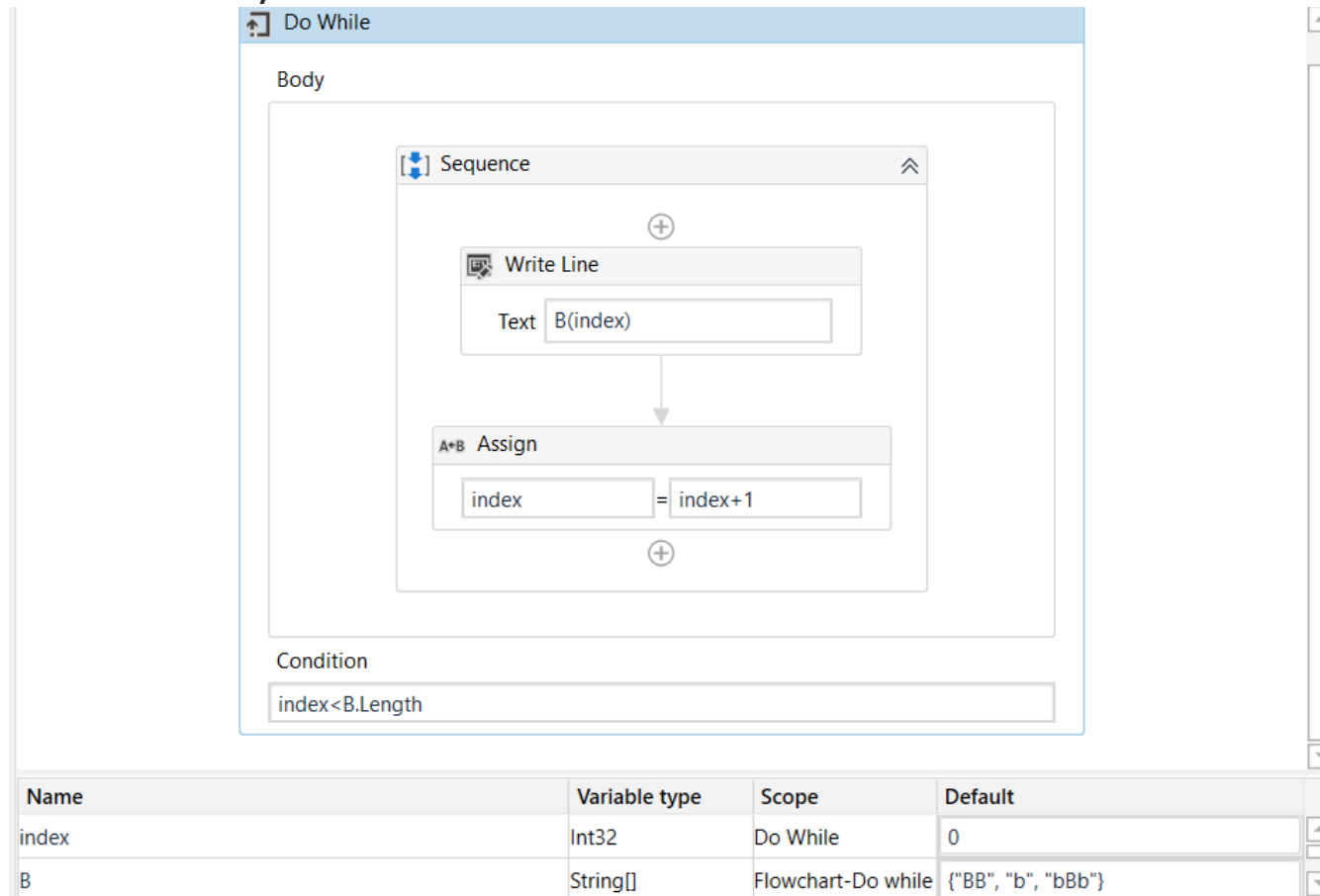
Private	<input type="checkbox"/>
To	index
Value	index+1

Name	Variable type	Scope	Default
index	Int32	While	0
z	Int32[]	Flowchart	{1,3,5,7,9}

see **Demo6 - ControlFlow**

# Control Flow. Do While Activity

- **Do While** activity:



see Demo6 - ControlFlow



# Demo 6 – Control Flow

- Tasks:

- **For Each**

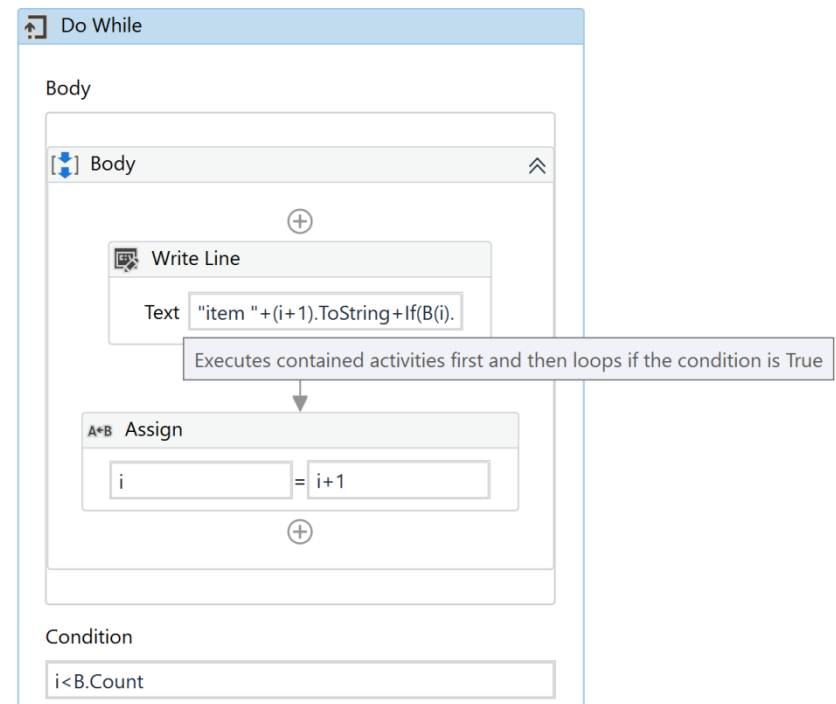
- navigating over an array of integers {0,0,2,4,6,8,10}
    - accessing the item and index from some specific array position

- **While**

- navigating over an array of integers {1,3,5,7,9}
    - accessing the array items based on the index

- **Do While**

- navigating over an array of strings {"B", "bBb", "BB"}
    - checking whether some condition is kept for each item in the array



see Demo4 – IfOperator (VB)

# Demo 7

- Create a process that performs the following actions:
  - 1. *generate* an integer number from 1 to 7;
  - 2. *read* a number to guess the generated number;
  - 3. *compare* the generated value
    - 3.1. print the message “Enter a smaller number!” or
    - 3.2. print the message “Enter a bigger number!”;
  - 4. *repeat* steps 2 and 3 until you succeed to find the number;
  - 5. *show* the message “Well done!!!”
- Is there a way to design the workflow without **ForEach/While/DoWhile** activities?

# References

- UiPath Docs - <https://docs.uipath.com/studio/docs>
- UiPath Forum - <https://forum.uipath.com/>
- UiPath Academy - <https://academy.uipath.com/>