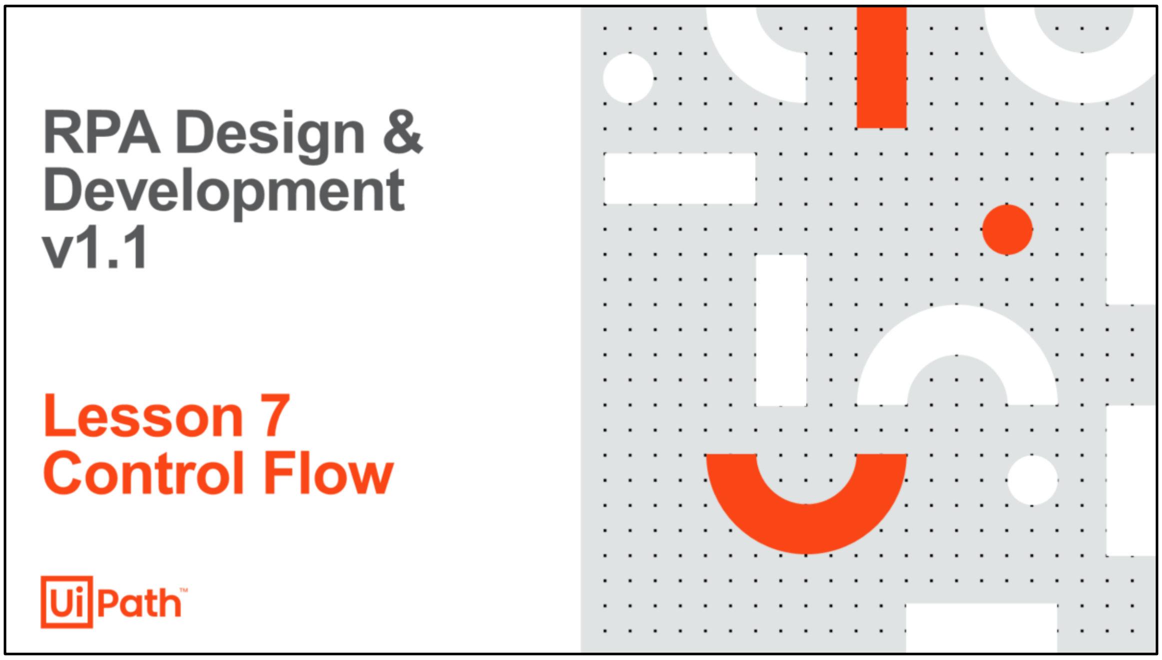
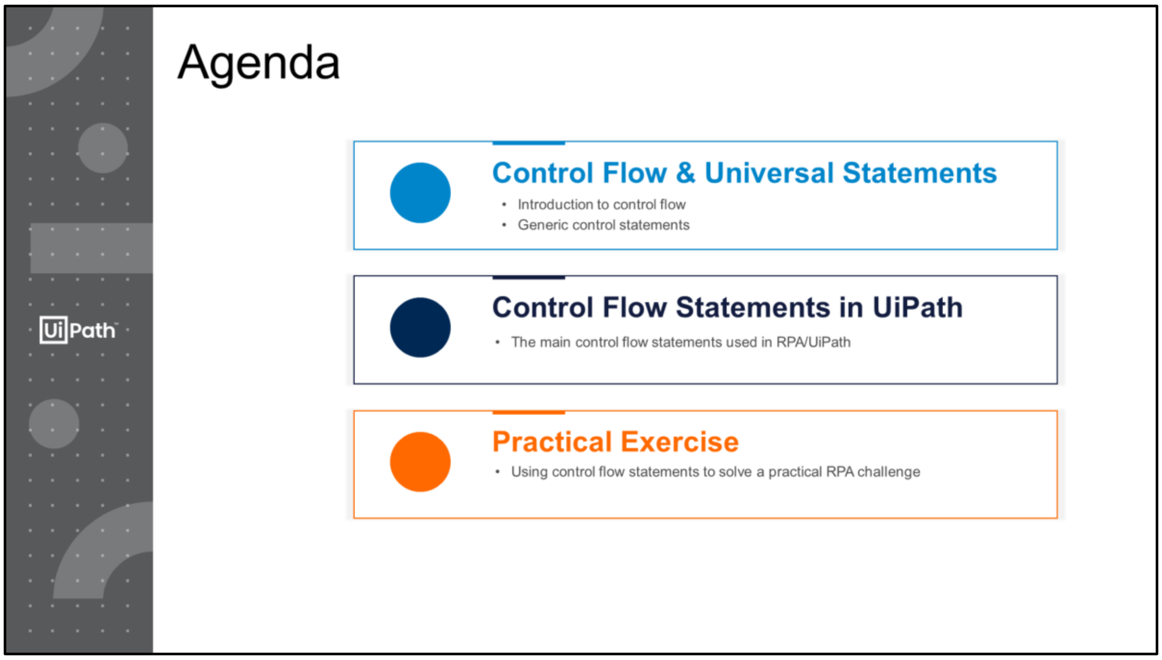


RPA Design & Development v1.1

Lesson 7 Control Flow

UiPath™

In this lesson, we will study and understand Control Flow and its importance in RPA Automation.


Agenda

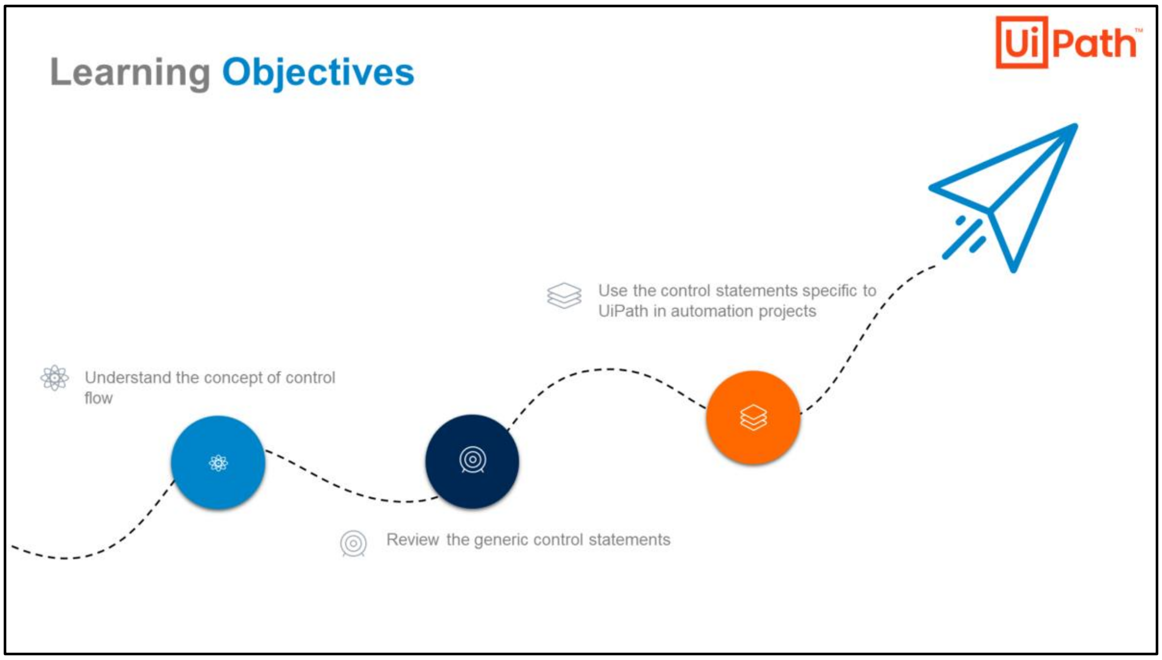
Control Flow & Universal Statements
- Introduction to control flow
- Generic control statements

Control Flow Statements in UiPath
- The main control flow statements used in RPA/UiPath

Practical Exercise
- Using control flow statements to solve a practical RPA challenge

In this lesson, we will cover the following topics:

- Control Flow & Universal Statements
  - Introduction to control flow
  - Generic control statements
- Control Flow Statements in UiPath
  - The main control flow statements used in RPA/UiPath
- Practical Exercise
  - Using control flow statements to solve a practical RPA challenge



Learning Objectives

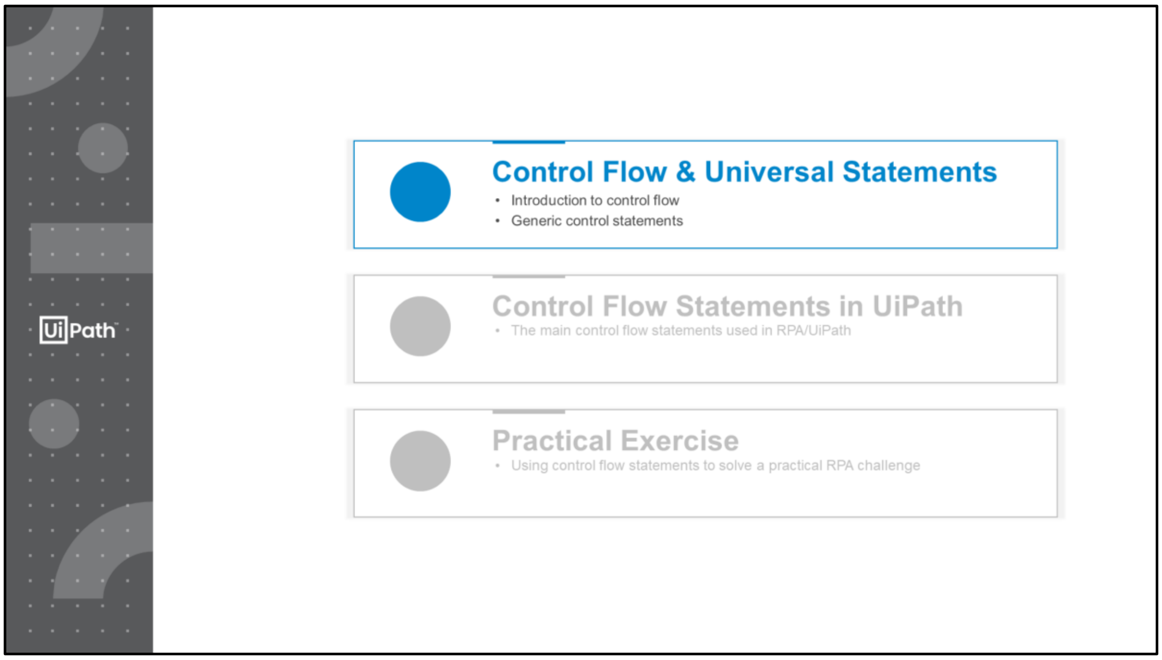
Understand the concept of control flow

Review the generic control statements

Use the control statements specific to UiPath in automation projects

At the end of this lesson, you will be able to:

- Understand the concept of control flow
- Review the generic control flow statements, applicable in all software and automation projects
- Use various control statements and workflow specific to UiPath in automation projects

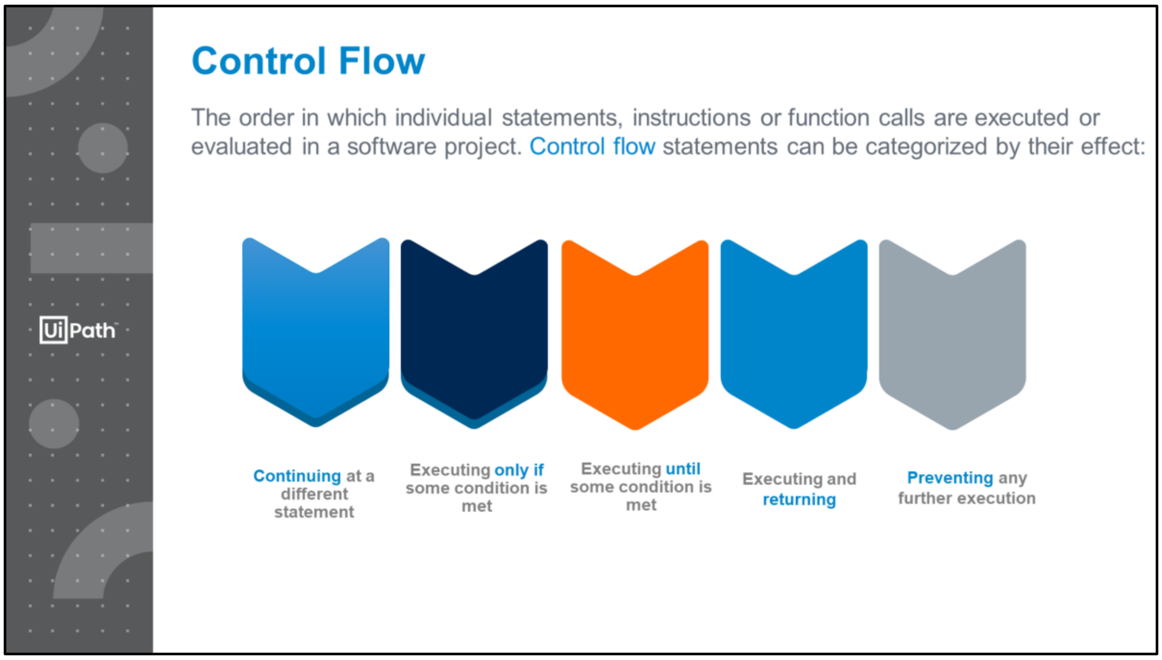


Control Flow & Universal Statements
- Introduction to control flow
- Generic control statements

Control Flow Statements in UiPath
- The main control flow statements used in RPA/UiPath

Practical Exercise
- Using control flow statements to solve a practical RPA challenge

In this topic, we will learn about the concept of Control Flow.

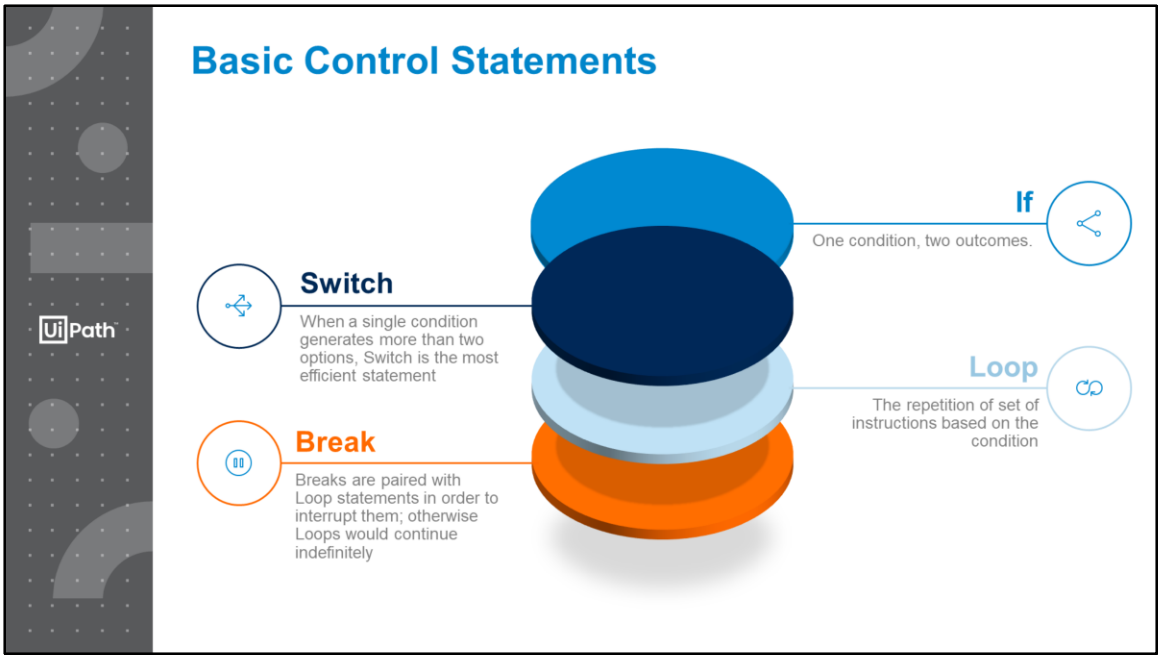

Control Flow

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:

Continuing at a different statement
Executing only if some condition is met
Executing until some condition is met
Executing and returning
Preventing any further execution

In UiPath, Control Flow is a programming concept that indicates the order in which individual statements, instructions or function calls are executed or performed. Control flow statements can be categorized by their effect:

- Continuing at a different statement (unconditional branch or jump)
- Executing a set of statements only if some condition is met (choice - i.e., conditional branch)
- Executing a set of statements zero or more times, until some condition is met (i.e., loop - the same as conditional branch)
- Executing a set of distant statements, after which the flow of control usually returns (subroutines, coroutines, and continuations)
- Stopping the program by preventing any further execution (unconditional halt)



Basic Control Statements

If

One condition, two outcomes.

Switch

When a single condition generates more than two options, Switch is the most efficient statement

Loop

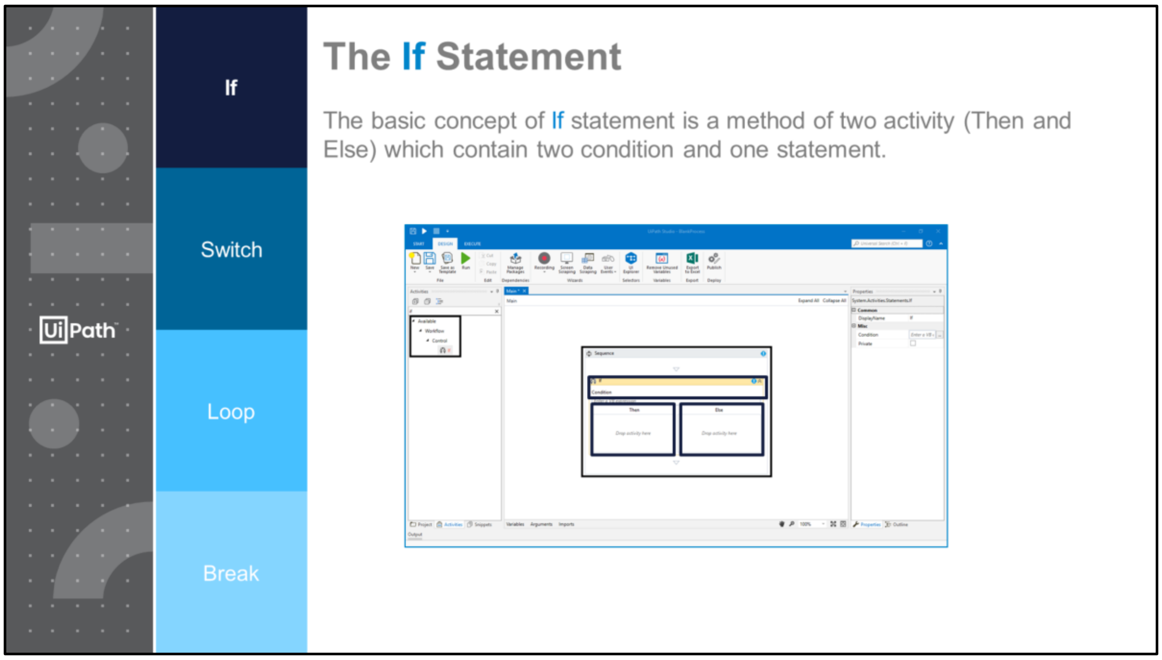
The repetition of set of instructions based on the condition

Break

Breaks are paired with Loop statements in order to interrupt them; otherwise Loops would continue indefinitely

There are four basic control statements that are the foundation of the control flow in all programming languages and software projects:

- **If**: the decision point with 2 branches
- **Switch**: the decision point with more than 2 branches
- **Loop**: the repetition of a set of instructions, based on a condition
- **Break:** the interruption of a loop, also based on a condition



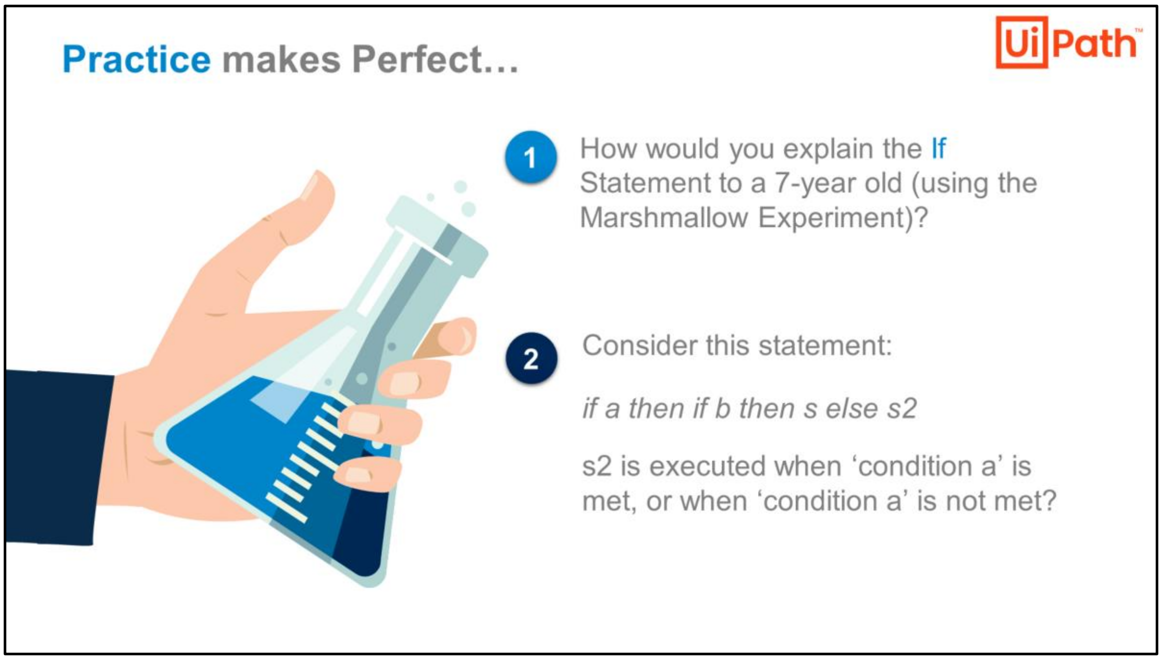
The If Statement

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

The basic concept of **If** statement is based on two statements (Then and Else). The **If** statement can be illustrated by an example where initially the two outcomes are fixed and "Then" activity is guided by the initial result.

A. The first condition is “Then” and it executes the process when the condition is “True.”
B. The second is “Else” and it is executed when the condition is “false.” Apart from this, the Condition statement takes the expression or variable declaration process executed in “Then and Else statement.” Hence, the **If** statements are controlled by conditions.

- **Condition:** It contains Boolean and argument expression that is executed in the “Then and Else” Statement.
- **Then**: If the condition is true then it comprises the data or activities.
- **Else:** If the condition is False then it comprises the data or activities.



Practice makes Perfect…

1. How would you explain the If Statement to a 7-year old (using the Marshmallow Experiment)?

2. Consider this statement:

   if a then if b then s else s2

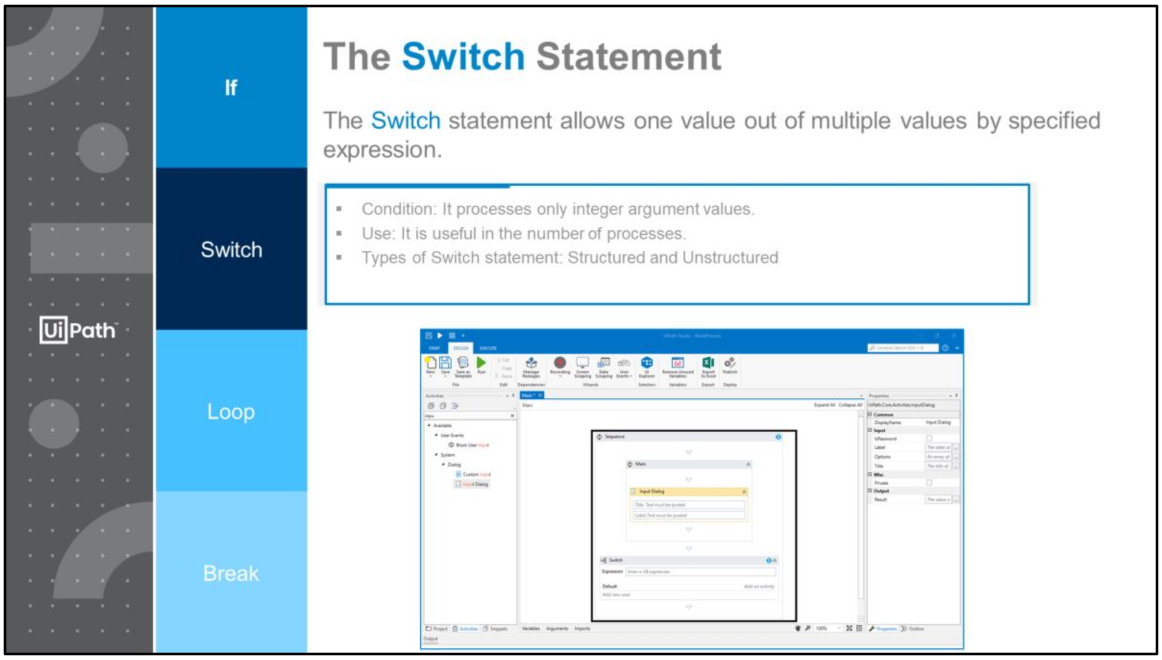
   s2 is executed when 'condition a' is met, or when 'condition a' is not met?

1. How would you explain the **If** statement to a 7-year old? To make it easier, let's use the Marshmallow Experiment as an example. In this experiment, a child would be let alone in a room with a marshmallow on a plate. If she could wait 5 minutes without eating the marshmallow, she would receive an extra marshmallow; otherwise, she would only have the first marshmallow (already eaten). The experiment was said to predict the success obtained later in life: those who could delay the gratification were considered grittier.
2. We covered nested **If** statements. We can theoretically nest **If** statements to recreate a much more complex process. But, this comes at a price. Analyze the steps below and answer – would s2 be executed when condition a is met or when it isn't?

Answer: it depends on the language. It could be both. This is why each language has additional signs to isolate the attributes:

- if a then (if b then s else s2) – if we want the s2 to be executed only if a is met
- If a then (if b then s) else s2 – if we want the s2 to be executed

only if a is not met



The Switch Statement

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

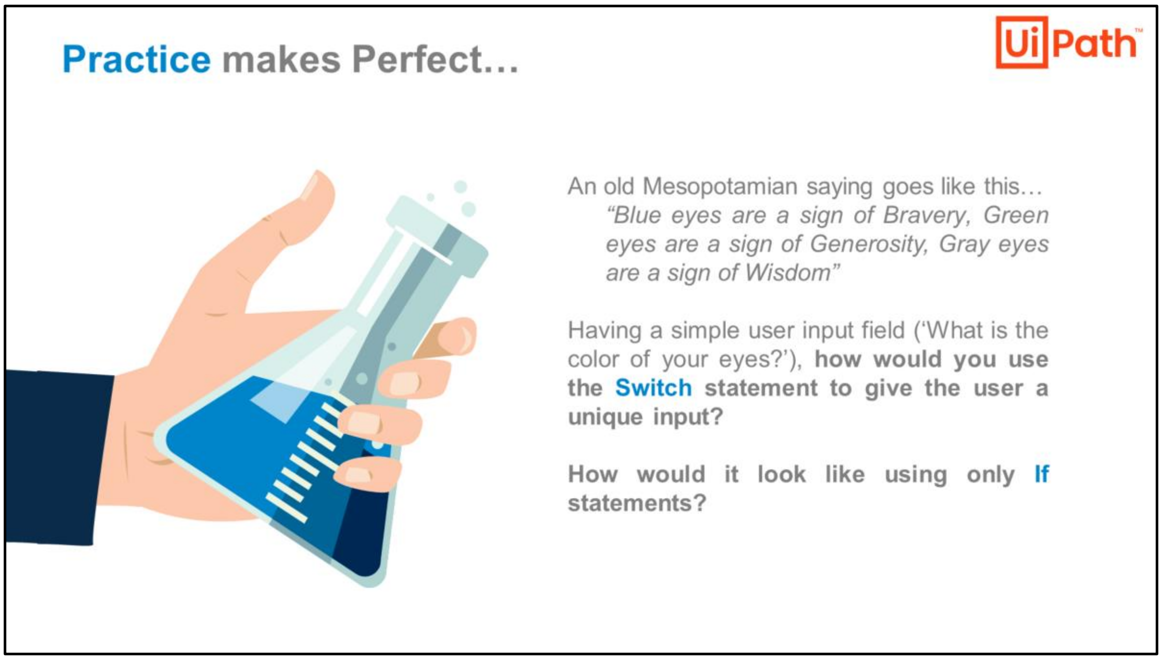
- Condition: It processes only integer argument values.
- Use: It is useful in the number of processes.
- Types of Switch statement: Structured and Unstructured

The **Switch statement** is allowing the one value out of multiple values by specified expression where it processes only integer argument values. You can change the properties also in it by the "TypeArgument" list. It is used in UiPath to categorize the number of processes. The use of the Switch statement is considered superior to an equivalent series of **If** statements because it is:

- Easier to debug
- Easier to read
- Statement fixed in-depth

These are the types of Switch statement which you can allow in the project development

- **Structured** – Only one branch is taken, then the execution continues at the end of the statement;
- **Unstructured** – Cases are treated as well as labels, so all of them can be executed



Practice makes Perfect…

An old Mesopotamian saying goes like this…

"Blue eyes are a sign of Bravery, Green eyes are a sign of Generosity, Gray eyes are a sign of Wisdom"

Having a simple user input field ('What is the color of your eyes?'), **how would you use the Switch statement to give the user a unique input?**

How would it look like using only If statements?

The point of this exercise is to facilitate a comparison between the Switch statement and nested If statements.

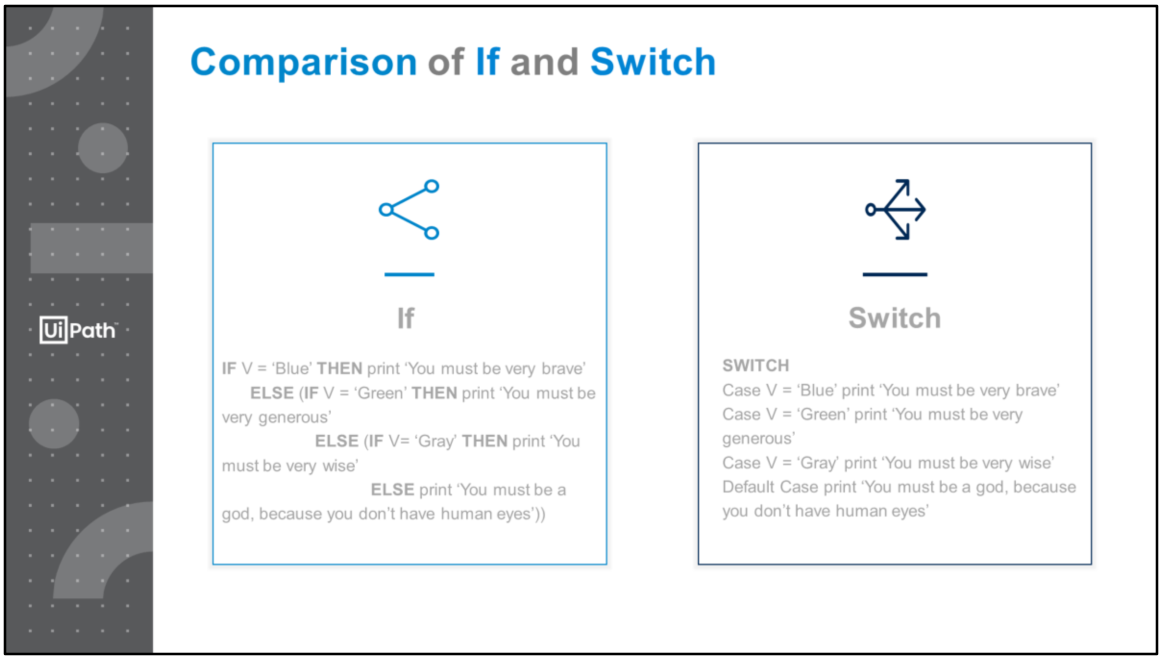
Using the Switch statement, the solution is very simple:

- Define a variable to store the input of the user (it can be predefined with 3 outcomes: blue, green, gray (dropdown list), or left as a string).
- Define a control variable with the 3 potential values.
- For each match between the variable and the control variable, the outcome message should be the appropriate one.
- Optionally, create a fourth statement for the case in which the user input doesn't match the 3 values.

Using the nested If statements, it becomes more complicated (but not impossible)

- Define a variable (V) to store the input of the user (it can be predefined with 3 outcomes: blue, green, gray (dropdown list), or left as a string).

```
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’))
```


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

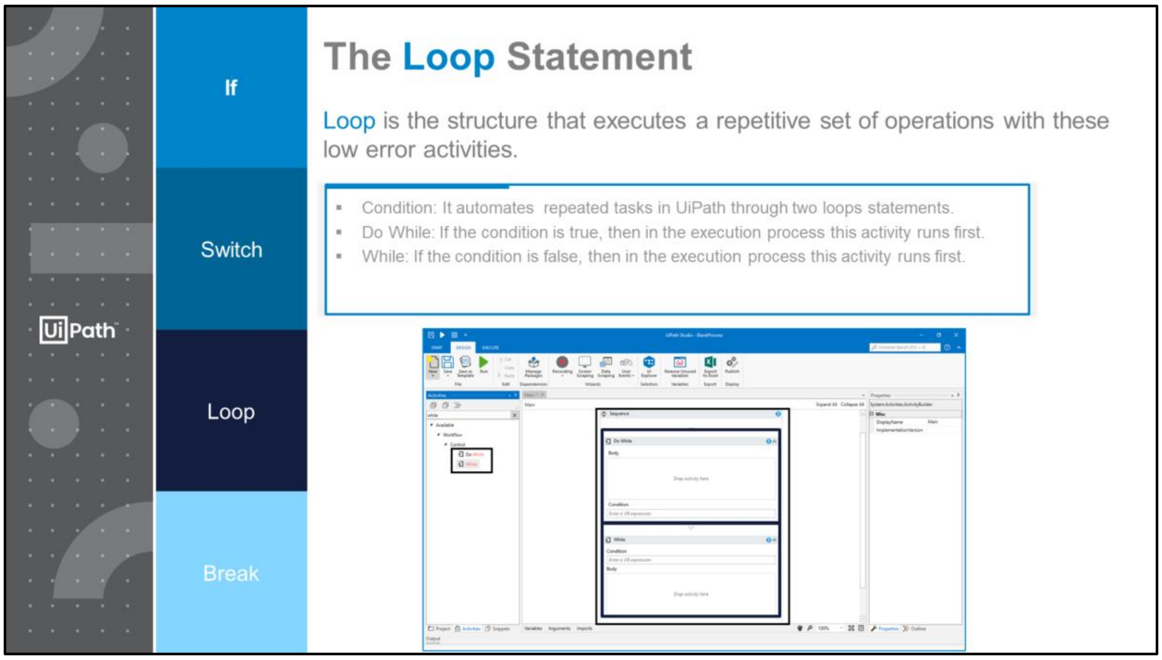
```
SWITCH
Case V = 'Blue' print 'You must be very brave'
Case V = 'Green' print 'You must be very generous'
Case V = 'Gray' print 'You must be very wise'
Default Case print 'You must be a god, because you don't have human eyes'
```

Comparison of If & Switch

- Switch is much easier to configure, while If statement requires a bigger effort and a higher attention.
- Whenever a new case appears (black eyes), it's enough to simply add a line; in the If solution, an entire If statement needs to be added, and the code has to be verified again down to the last line.

Note* Both conditional statements are used in programming language.

The **If** statement evaluates integer, character, pointer or floating-point type or boolean type.
On the other hand, switch statement evaluates only character or an integer datatype.



The Loop Statement

Loop is the structure that executes a repetitive set of operations with these low error activities.

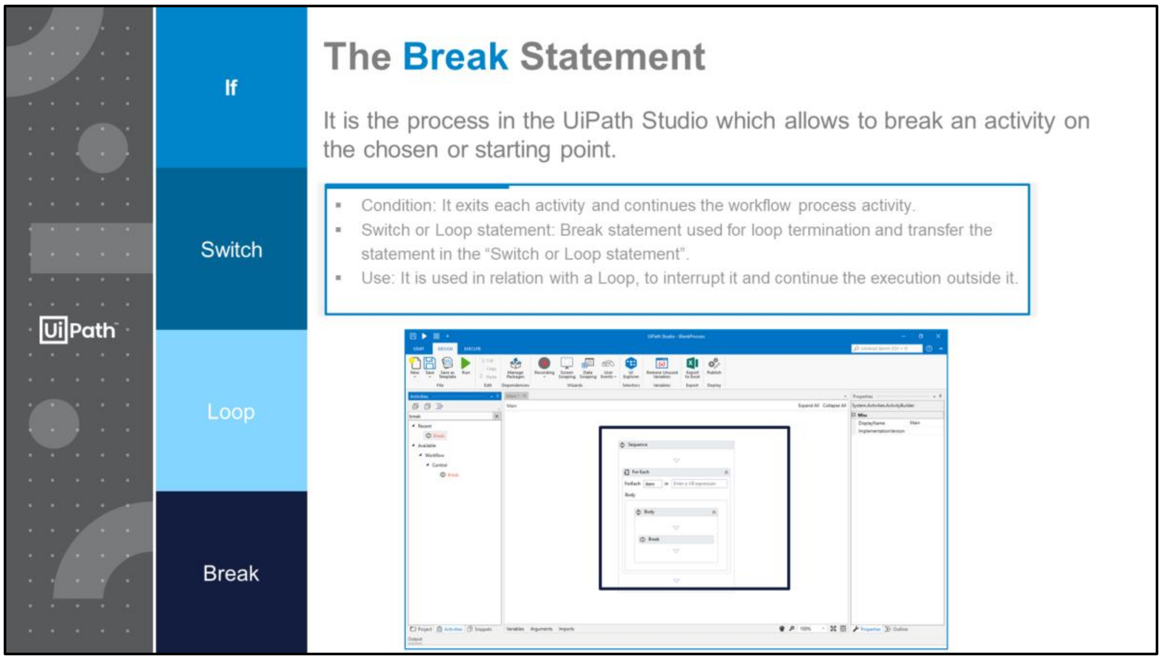
- Condition: It automates repeated tasks in UiPath through two loops statements.
- Do While: If the condition is true, then in the execution process this activity runs first.
- While: If the condition is false, then in the execution process this activity runs first.

Loop is a structure that is used to automates repetitive tasks in UiPath. It executes sets of operations repetitively, faster, with low error rates through loops. It executes the process within the loop and the essential element is the control in the loop. It could be:

- Count-controlled: In such a Loop, the number of execution of the loop is predefined.
- Indefinite Case: Such a Loop executes for an unlimited number of times.
- Conditional: This is widely used in programming. It contains a validation mechanism and executes when / until a certain condition is met.

The Loop Statement is of two types:

- Do While: In the Execution process, this activity runs when the condition is true.
- While: In the Execution process, this activity runs when the condition is False.

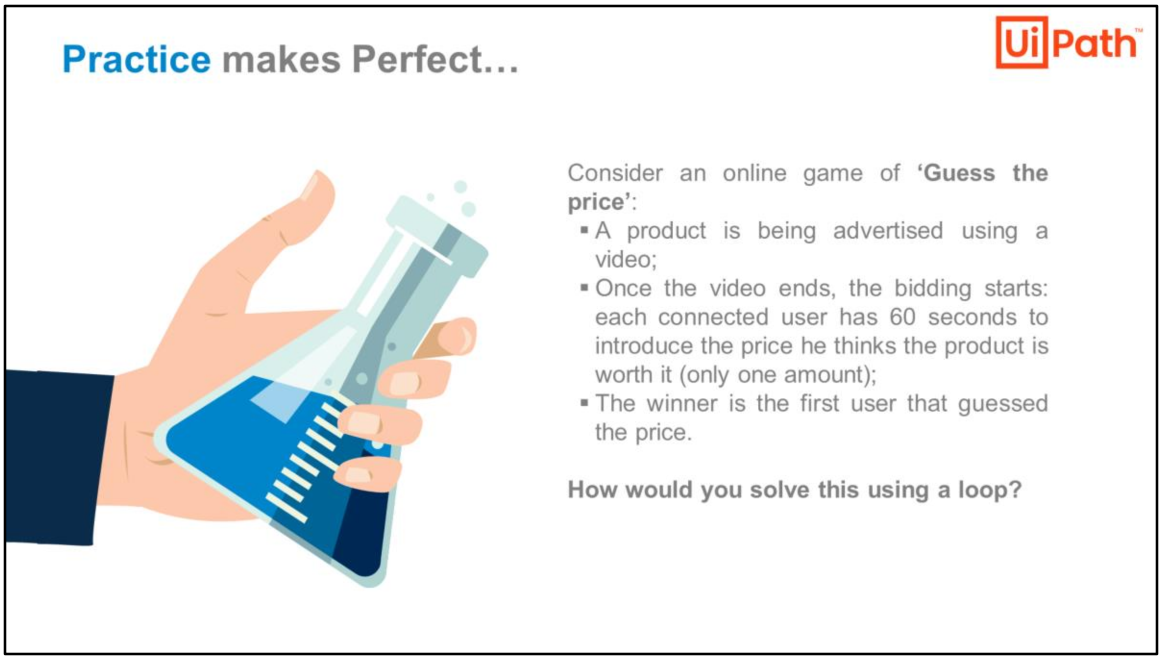


The Break Statement

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

- 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.

The Break statement allows you to break an activity on the chosen or starting point and enables the process to continue in the next activity. When you build a project with the help of this statement, it exits each activity and continues the workflow process activity.
In project development, this statement is used for loop termination and transfer the statement in the "Switch or Loop statement." For example, to end a branch and move to a different sequence. The most common use is in relation with a Loop, where it is used to interrupt it and continue the execution outside it.


Practice makes Perfect…

Consider an online game of **‘Guess the price’**:

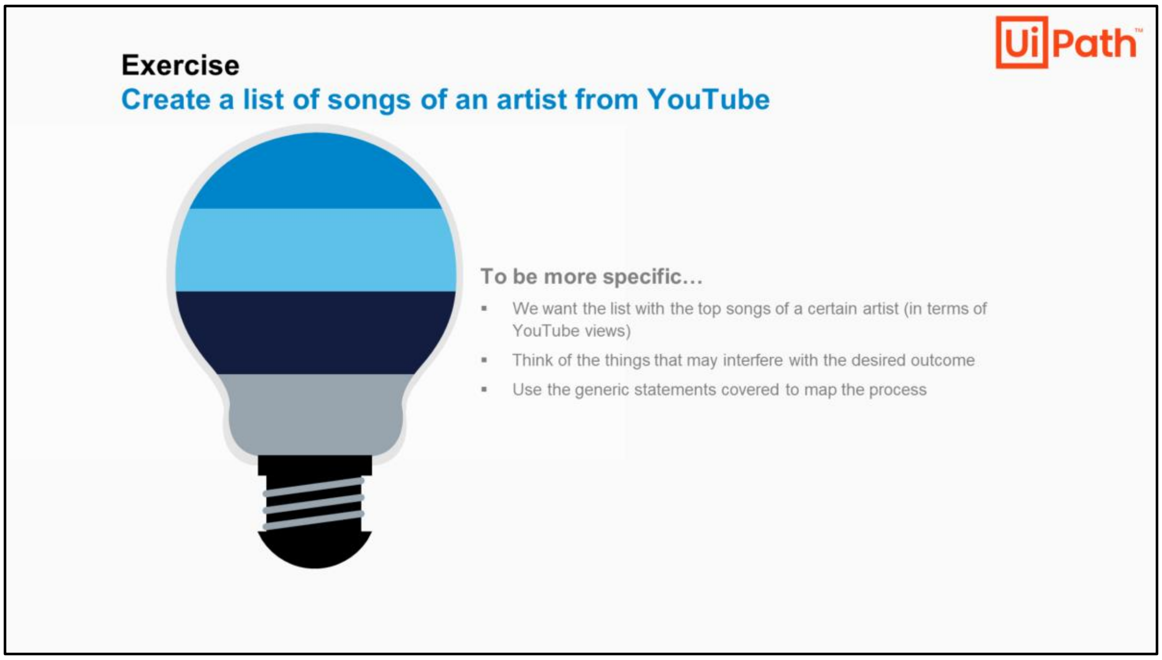
- A product is being advertised using a video;
- Once the video ends, the bidding starts: each connected user has 60 seconds to introduce the price he thinks the product is worth it (only one amount);
- The winner is the first user that guessed the price.

How would you solve this using a loop?

This is not the only solution to understand the challenge of Control Flow. However, all solutions should have these common features.
Two variables: the actual price of the product (P, integer/float), the prices indicated by the users (UP, dataTable)

- The prices indicated by the users should be assigned in UP as they are received (so that the first value should be the value sent by the first user)
- LOOP conditional (values in UP not verified) / count-controlled (values in UP)

```
    P = UP? YES Write “We have a winner”; BREAK
            NO Continue
```


Exercise

Create a list of songs of an artist from YouTube

To be more specific…

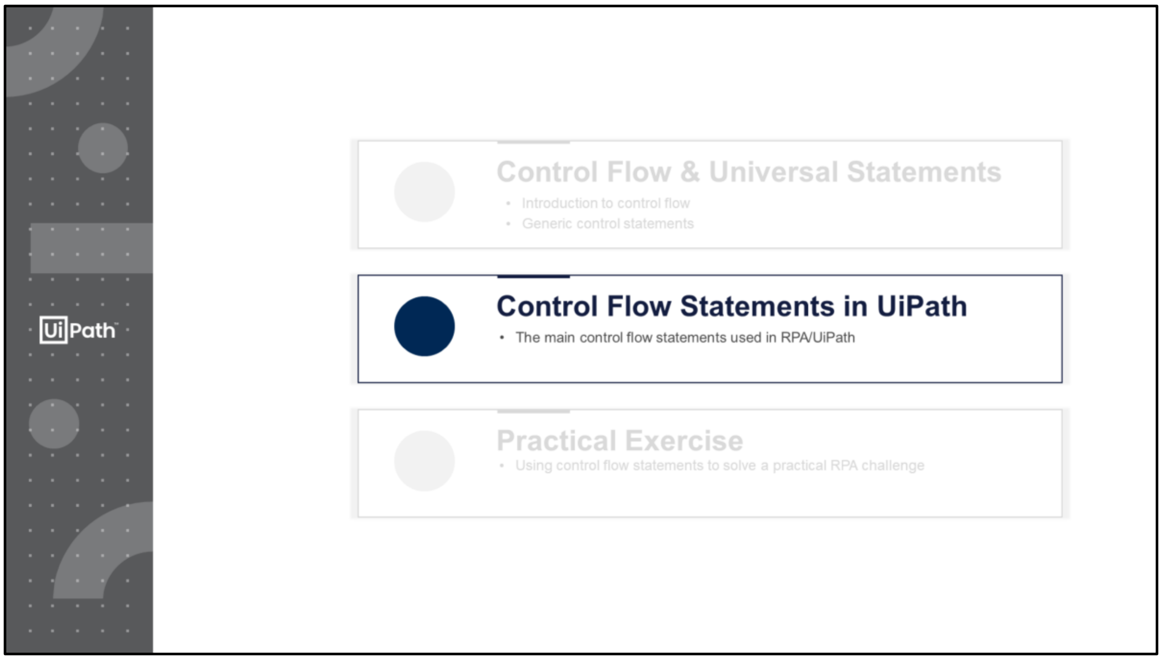
- We want the list with the top songs of a certain artist (in terms of YouTube views)
- Think of the things that may interfere with the desired outcome
- Use the generic statements covered to map the process

Let's say we want the top songs in terms of YouTube views. We want to exclude live songs because of their quality; and make sure that the songs are of that artist (sometimes, YouTube uses its 'similar artists' feature in displaying search results).

So, how can we use the generic statements to address the request?

```
0. Variables needed: integer N = 0, integer NV = 1, string ArtistName (can be requested using a dialog box), string SongTitle
1. Desktop/Folder in Explorer > Right-click > New… > Microsoft Excel document
2. Change title to 'List of songs' (optional)
3. Double-click to open document
4. Open web browser
5. Type in www.youtube.com
6. Search ArtistName > Enter
7. Filter > Sort by > View count
8. IF N < 20 THEN
9. Go to video NV
10. IF SongTitle contains 'live' THEN
                        NV=NV+1
                        Break [Go to 9]
11. ELSE IF title contains ArtistName THEN
                  copy link
                                  go to 'List of songs'
```

```
                              paste link
12. ELSE NV=NV+1
              Break [Go to 9]
13. Copy songtitle
     go to ‘List of songs’
     paste songtitle
14. N=N+1
15. Break [go to 8]
```



Control Flow & Universal Statements

- Introduction to control flow
- Generic control statements

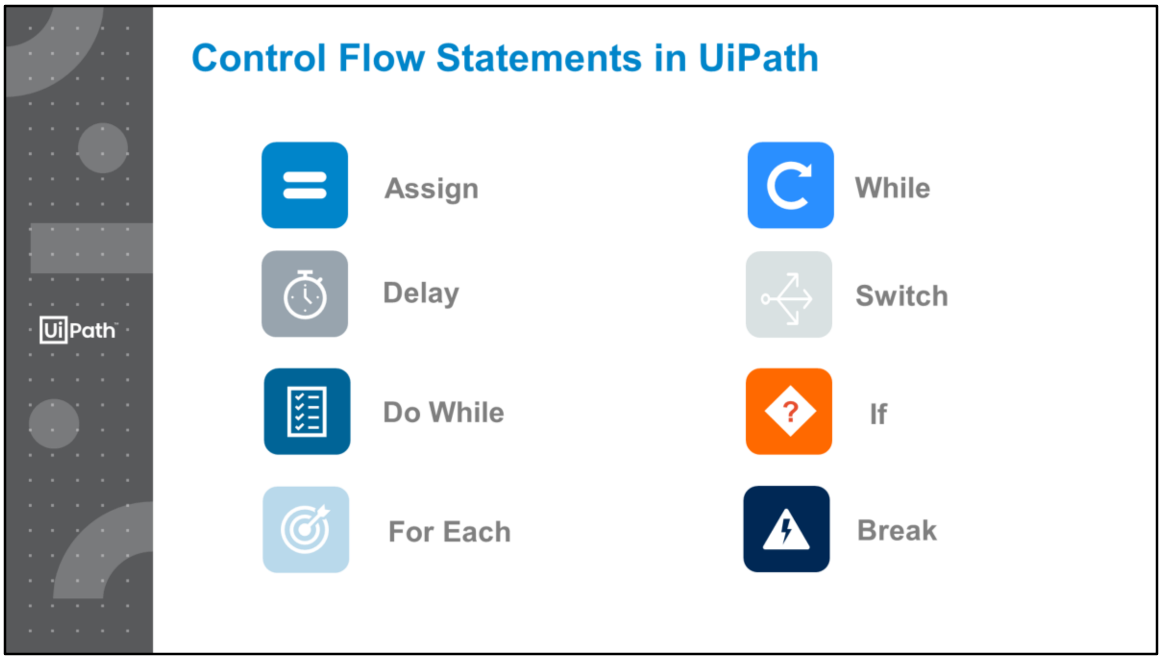
Control Flow Statements in UiPath

- The main control flow statements used in RPA/UiPath

Practical Exercise

- Using control flow statements to solve a practical RPA challenge

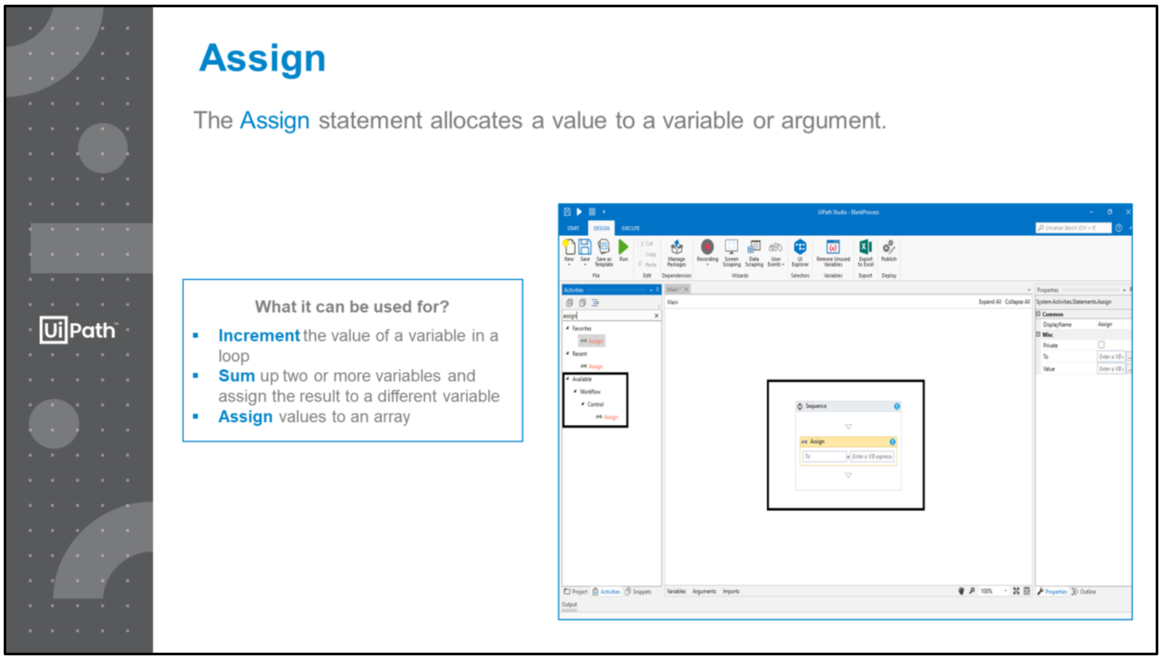
In this topic, we will learn about Control Flow Statements in UiPath.



Control Flow Statements in UiPath

Assign	While
Delay	Switch
Do While	If
For Each	Break

How is the control flow concept applied in UiPath?
There are basically eight different types of control flow statements that we use in UiPath. Some are generic control flow statements like **If** and **Switch** where as others are specific like **Do While**, **For Each** and **While** are loops. **Break** in UiPath is not the generic break that we use in programming, but it serves the same purpose.



Assign

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

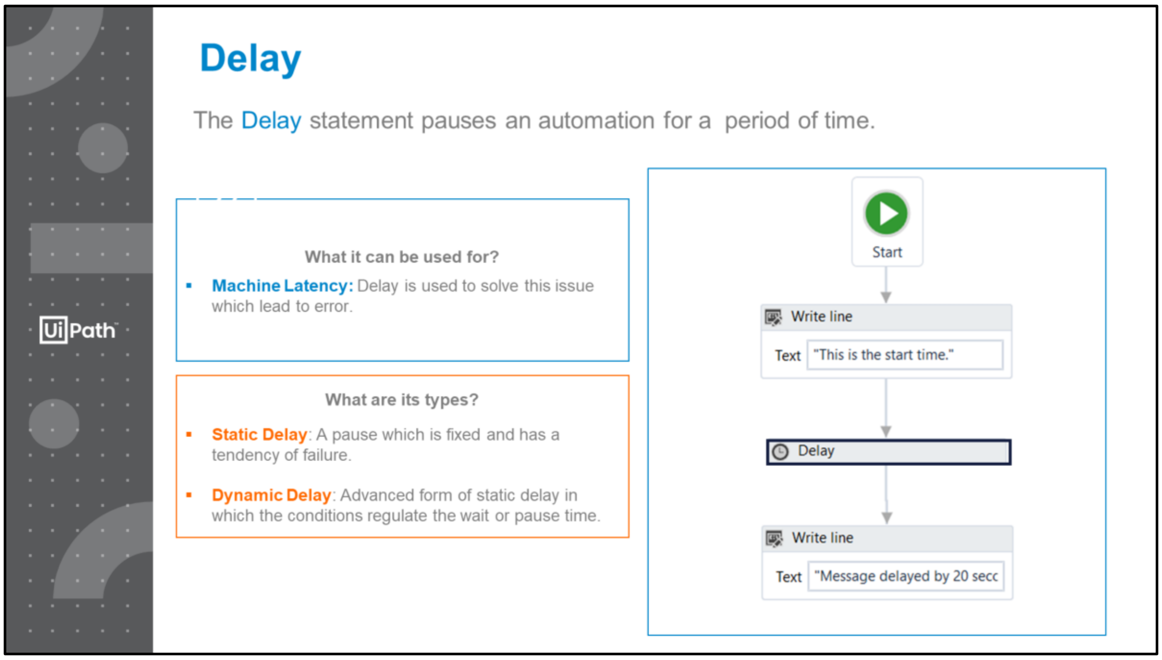
What it can be used for?

- **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

Primarily, **Assign** is used for allocating values but in reality, it is used in a much broader sense.
Assign is used to:

- Increase the value of a variable(this is very useful while using a limited loop)
- Perform the mathematical calculations or operations and assign the result or outcome to a new variable.
- Allocate value to an entire Array

In programming languages, where variables are used a lot, Assign is an essential function both in the beginning, where initial values are assigned, throughout the project, when the exchange of information is done through variables, or at the end, when end values are being assigned (for example, outputs).


Delay

The Delay statement pauses an automation for a period of time.

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.

Start
Write line
Text "This is the start time."
Delay
Write line
Text "Message delayed by 20 secc

The purpose of the **Delay** statement is to pause an automation for a given period of time. It's the simplest way:

- To address a common issue in programming, and especially in automation – the loading time of a specific application or website
- As a generic approach, when two operations that need to be in a sequence are not ready at the same time

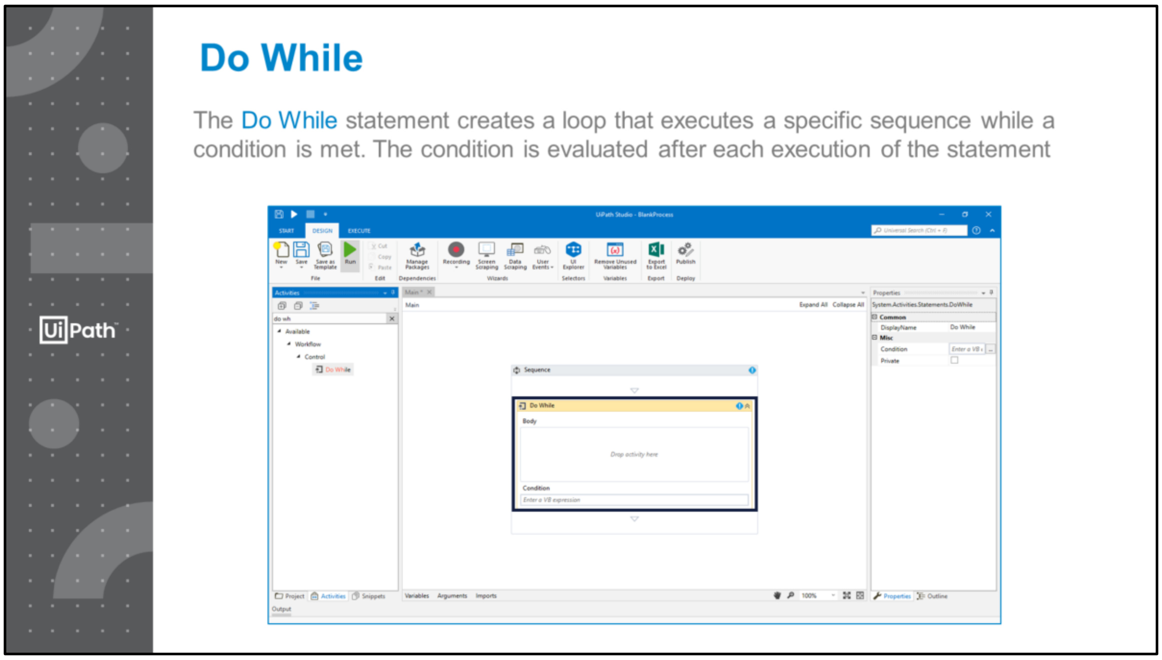
The concept of delay has solved many issues such as the machine latency which leads to system slowness and when a specific application is not able to load or sync at that particular time therefore leading to the automation throwing error or failure.

Although it is the easiest way to pause a set of operations, it is neither efficient, nor failproof (as the loading time may vary for the same operation). There are better ways to replace the Delay statement (for example, by seeking a certain element on the screen that appears only after the application/page has loaded).

There are two types of delay that we use in UiPath:

- Static Delay**:** It is defined as that pause which is fixed and has a tendency of failure.

- Dynamic delay: It is the advanced form of static delay in which the wait or pause time is regulated by the conditions.



Do While

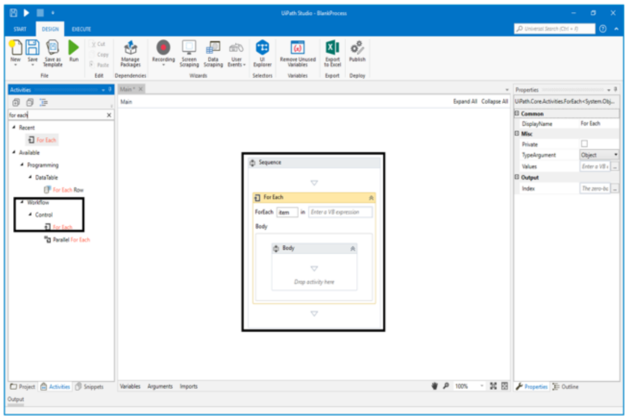
The Do While statement creates a loop that executes a specific sequence while a condition is met. The condition is evaluated after each execution of the statement

In general, different loops are being defined based on the type of condition and the verification mechanism.
In UiPath, **Do While** is one of the loops, and the condition is verified after the execution of the instructions.
Just like in the case of the If statement, the condition to be verified needs to be Boolean.

For example, if you would want a sequence to be executed 20 times, you could define a variable, assign the initial value 0, increase by 1 every time the sequence is executed. In this case, the condition verification should be the variable < 20.

For Each

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

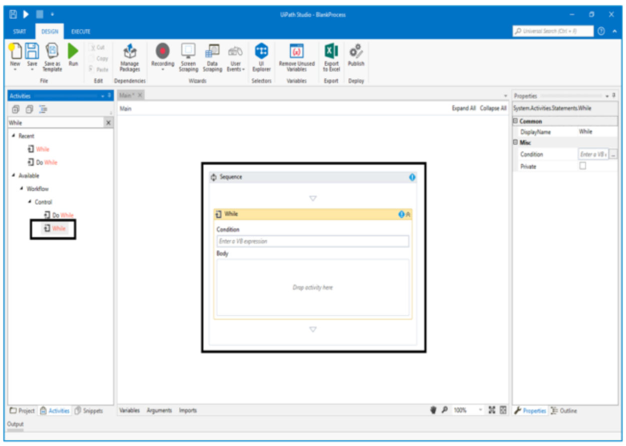


For Each is another example of a Loop, used to go through a series of elements. This may come in useful when the same actions need to be performed on the elements from a list – user-generated data or from other sources.

The **Break** statement is used mandatorily to interrupt the **For Each** statement.

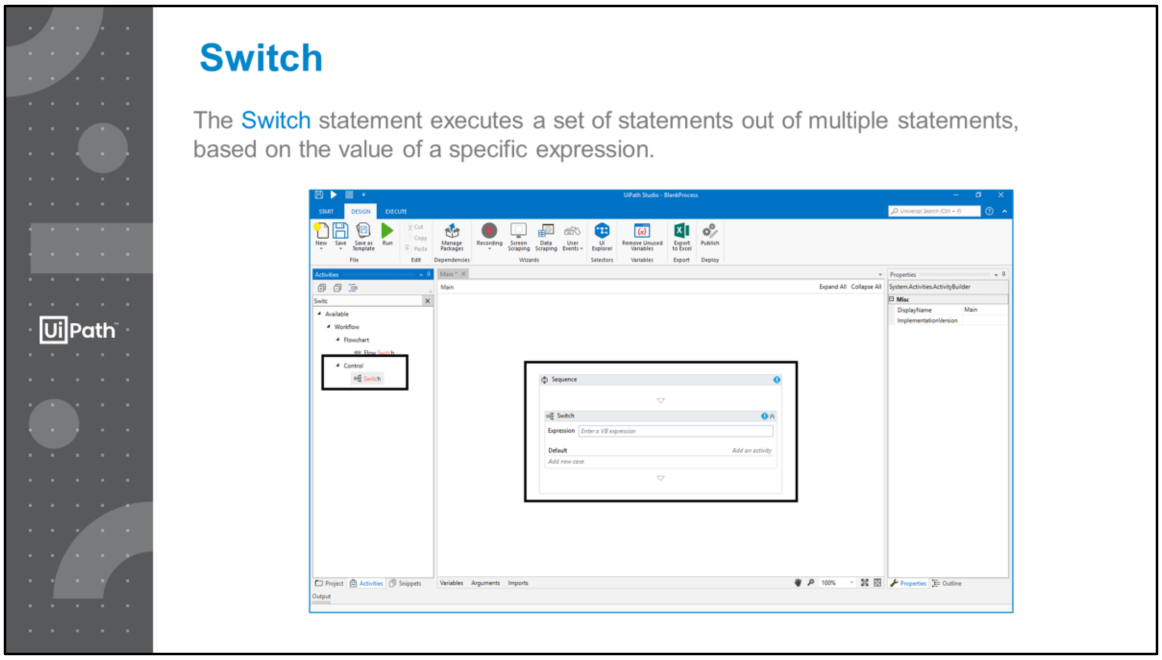
While

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



While is very similar to **Do While**. The most important aspect is that the condition is verified before the execution of the instructions. Again, just like in the case of the **If** and **Do While** statements, the condition to be verified needs to be Boolean.

Using the same example that we had for **Do While**, i.e. the sequence to be executed 20 times, you still define a variable, assign the initial value of 0, and increase by 1 every time the sequence is executed. But this time, since the verification is done before, the condition verification should be the variable < 21.

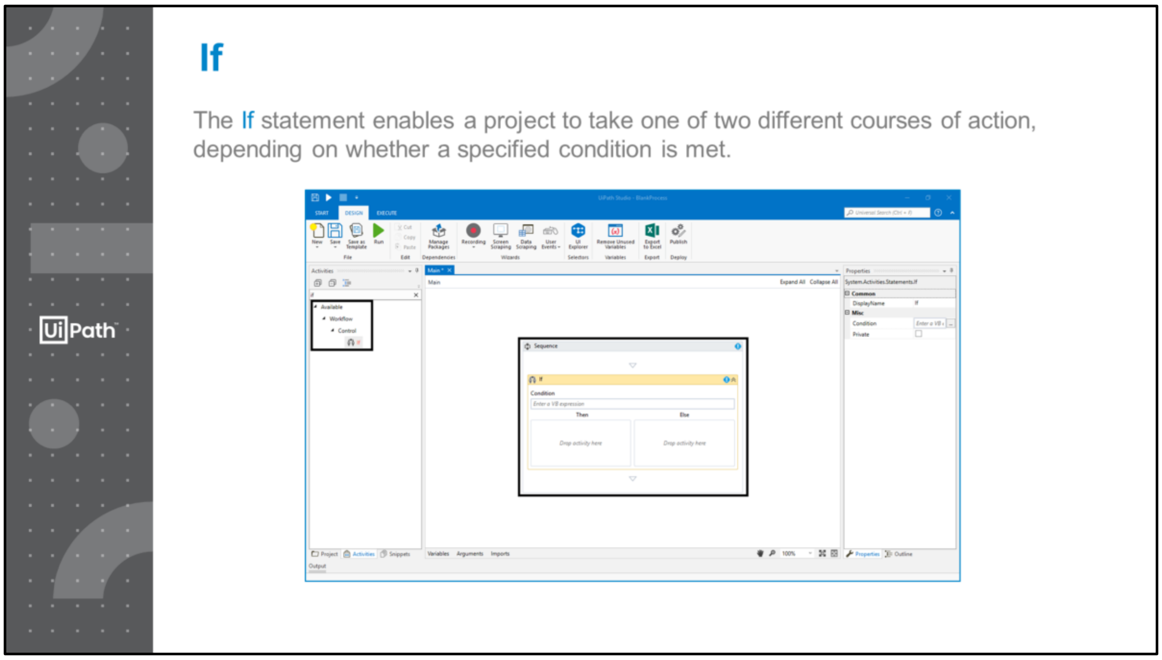


Switch

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

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).

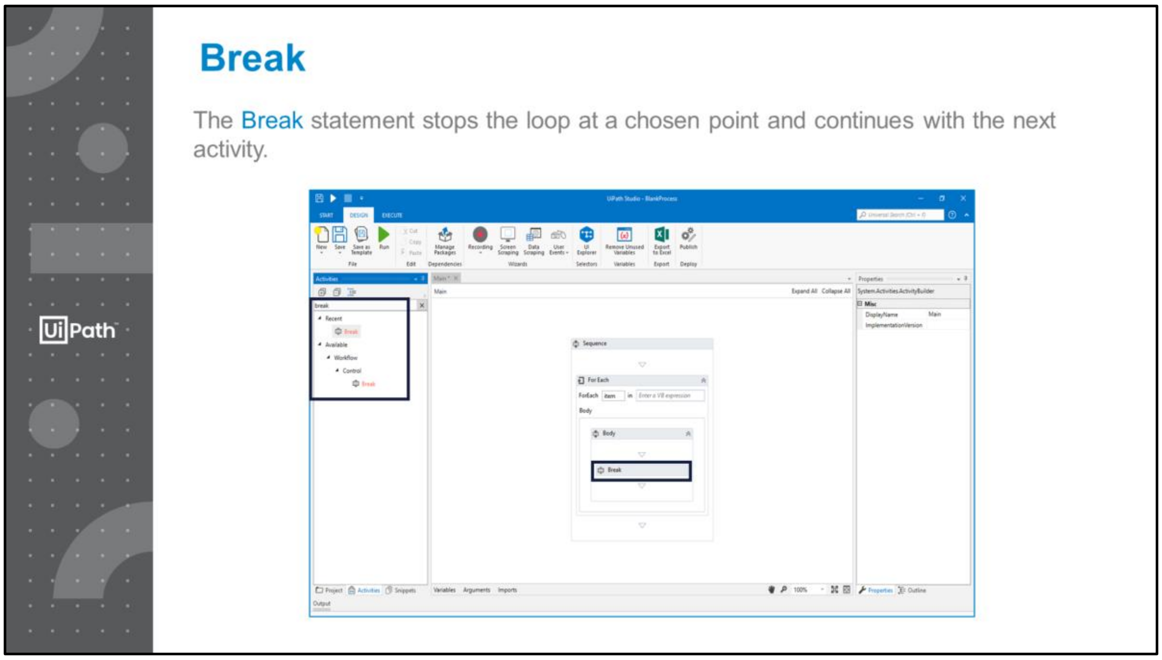
A specific feature that can be used in UiPath is what we call Default Case – when none of the defined cases is matched against the expression, the default case is executed. This is optional and can be replaced by a specific set of instructions to be executed when the pre-defined cases are not matched, or can even be left blank.



If

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

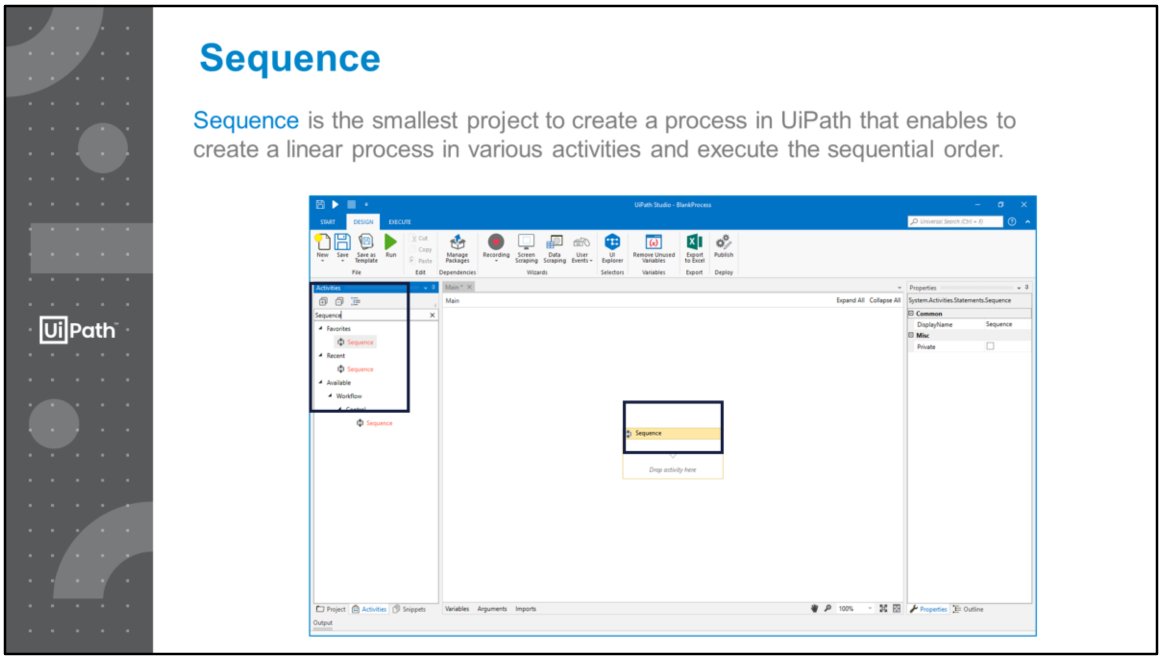
If command hep us in deciding the flow of any automation. This is mainly associated with taking decision like what should be followed on.
If command helps in taking decision regarding the execution of the further process.



Break

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

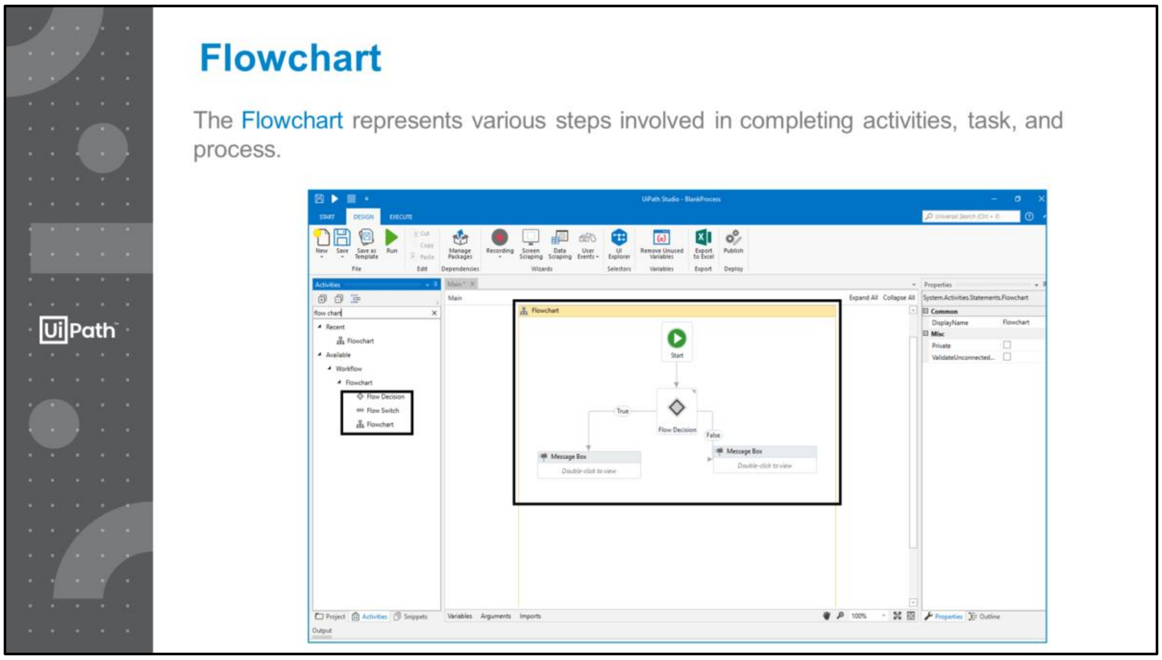
Although we discussed the **Break** statement as being the statement universally used to stop a loop, whereas in UiPath we use it only to interrupt the Each activity (which is a form of the loop). In UiPath we don’t use the broken concept for a **While** and **Do While** as there they are already included as conditions.



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.

It 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. Apart from this, it can be reused again and again in the UiPath project development.

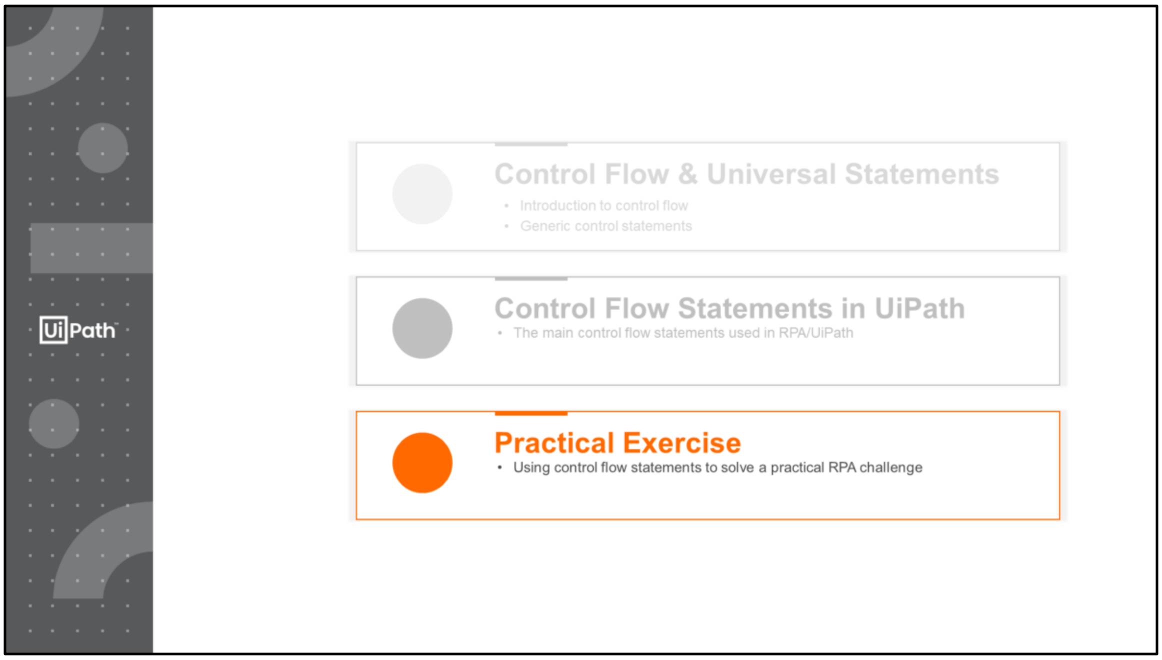


Flowchart

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

A flowchart is a diagrammatical approach which represents various steps involved in completing activities, task, and process. It is based on three types of activities:

- Flowchart
- Flow Decision
- Flow Switch



Control Flow & Universal Statements

- Introduction to control flow
- Generic control statements

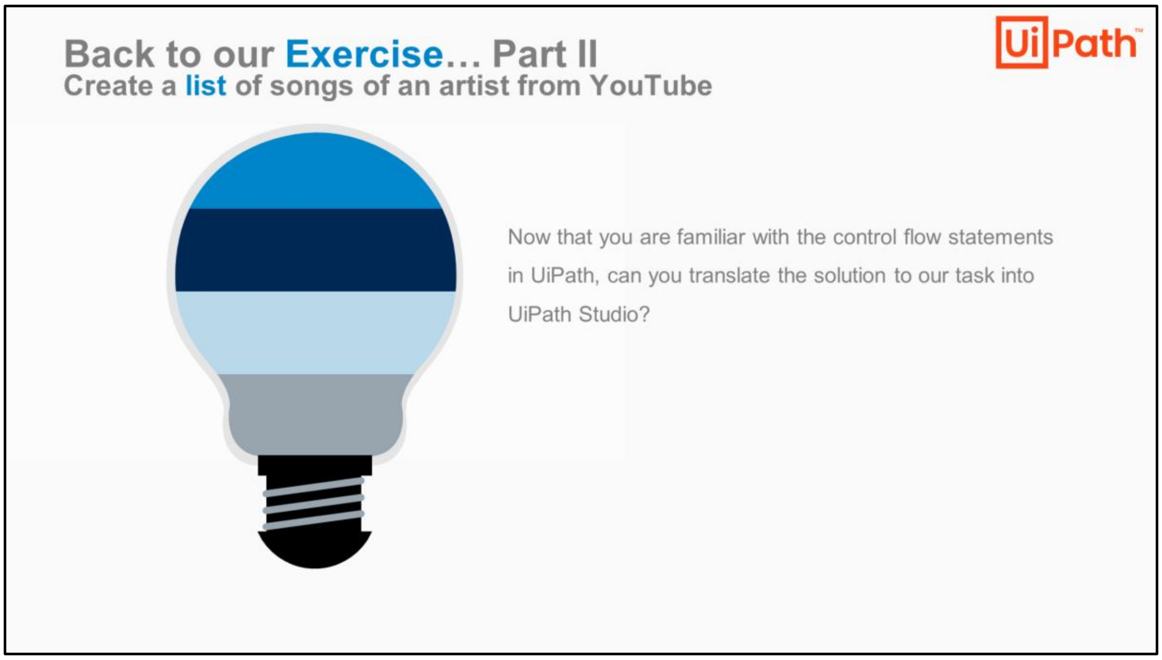
Control Flow Statements in UiPath

- The main control flow statements used in RPA/UiPath

Practical Exercise

- Using control flow statements to solve a practical RPA challenge

In this topic, we will use control flow statements to solve a practical RPA challenge.

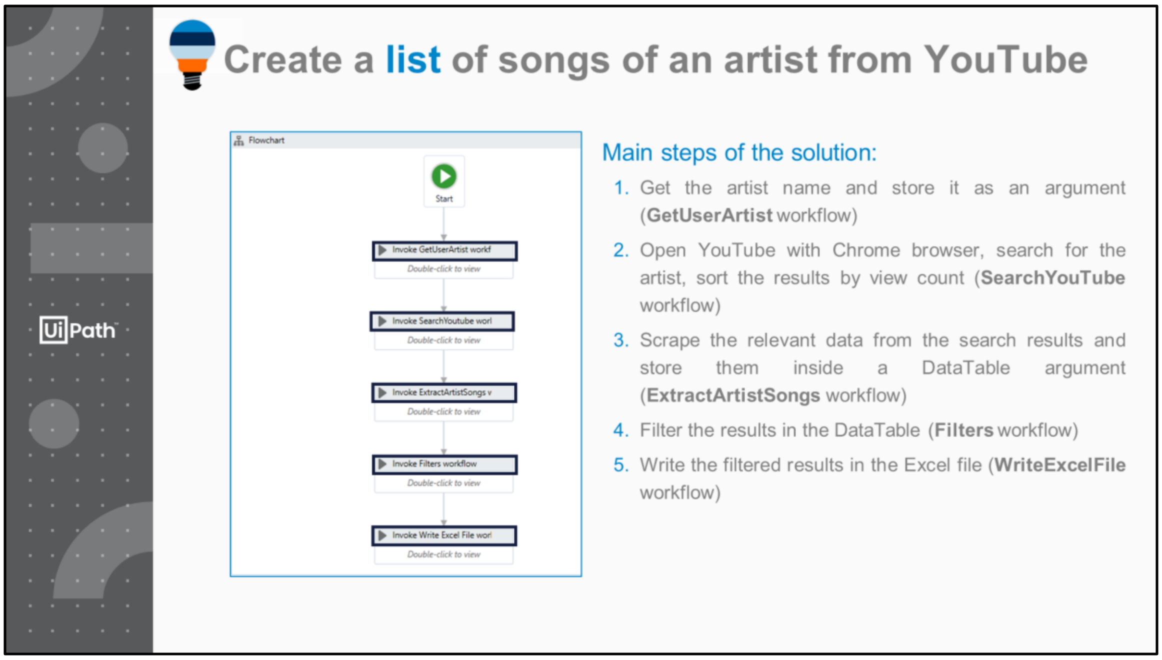


Back to our Exercise… Part II

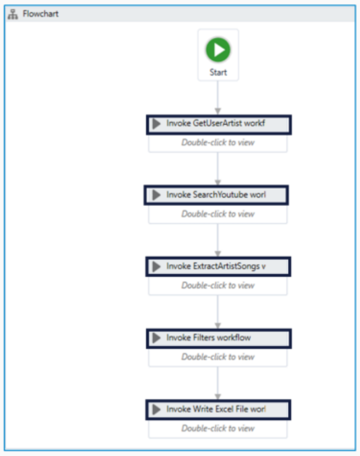
Create a list of songs of an artist from YouTube

Now that you are familiar with the control flow statements in UiPath, can you translate the solution to our task into UiPath Studio?

Here we will see practical implementations of how the control flow work.



Create a list of songs of an artist from YouTube



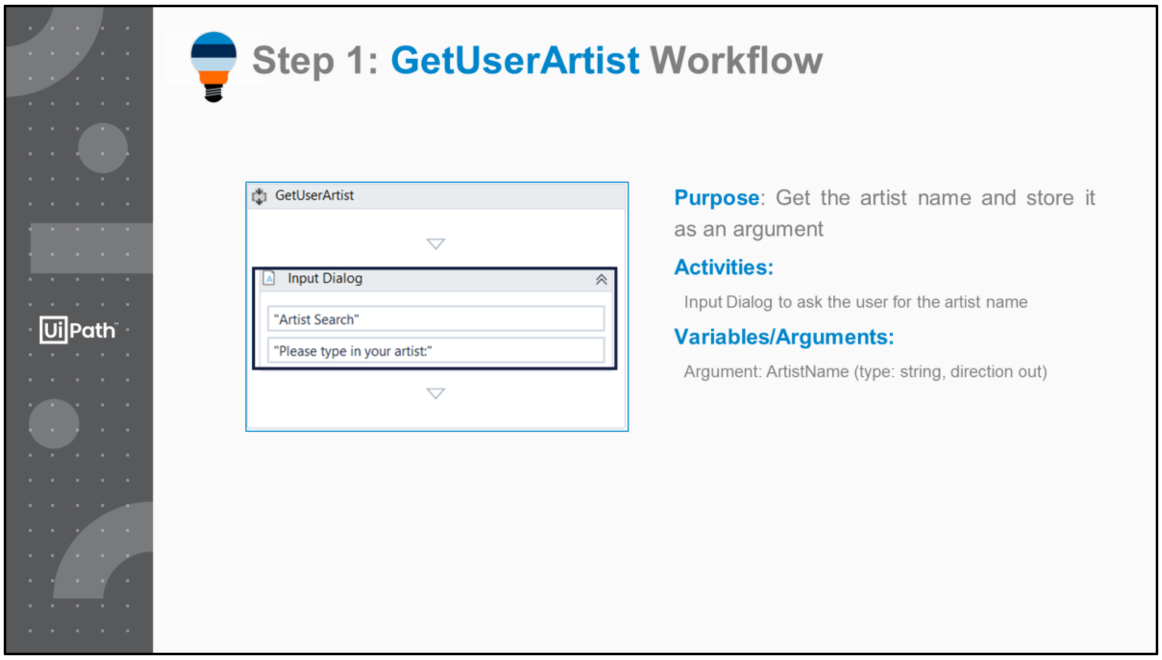
Main steps of the solution:

1. Get the artist name and store it as an argument (**GetUserArtist** workflow)
2. Open YouTube with Chrome browser, search for the artist, sort the results by view count (**SearchYouTube** workflow)
3. Scrape the relevant data from the search results and store them inside a DataTable argument (**ExtractArtistSongs** workflow)
4. Filter the results in the DataTable (**Filters** workflow)
5. Write the filtered results in the Excel file (**WriteExcelFile** workflow)

There are basically five workflows that we have created for this simple task:

- GetUserArtist
- SearchYouTube
- ExtractArtistSongs
- Filters
- WriteExcelFile

These workflows are responsible for performing their own activity. For example, in this case GetUserArtist will get the artist name and store as an argument and WriteExcelFile will put the filtered result in an excel file.



Step 1: GetUserArtist Workflow

GetUserArtist

Input Dialog

"Artist Search"

"Please type in your artist:"

Purpose: Get the artist name and store it as an argument

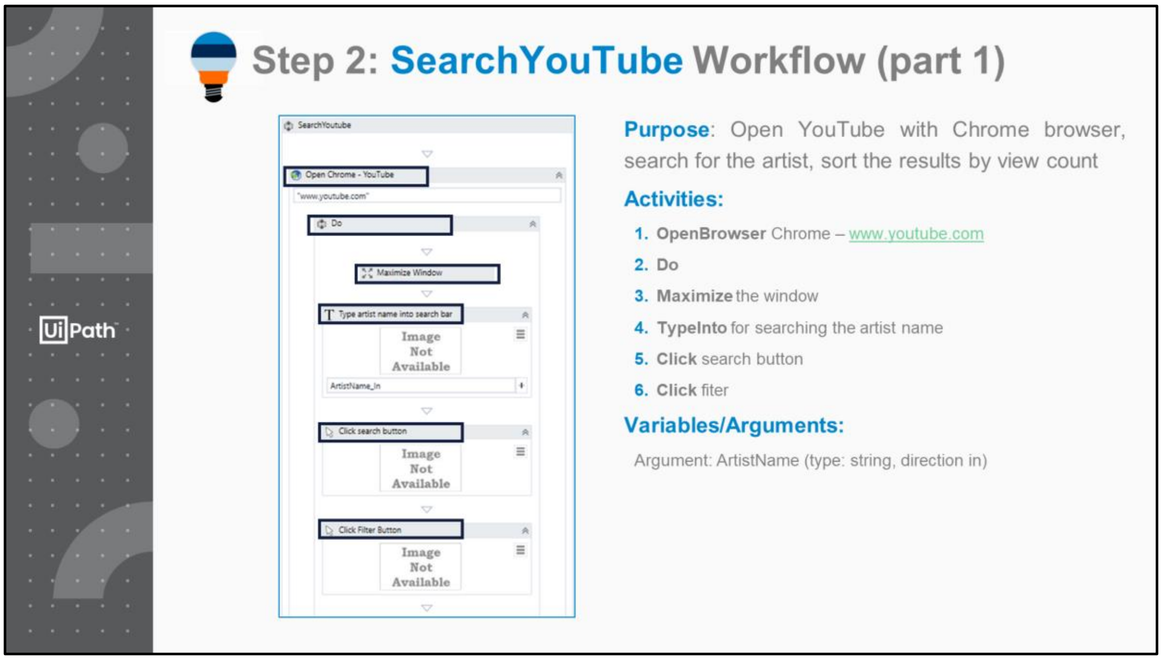
Activities:

Input Dialog to ask the user for the artist name

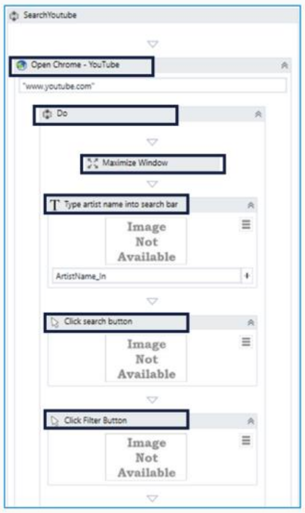
Variables/Arguments:

Argument: ArtistName (type: string, direction out)

The Input Dialog activity is added in the workflow, with a title for the pop-up window and a text.
In order to use the name given by the user, it has to be stored in an Argument (that will be used in a different workflow). The direction needs to be set 'Out', and the type is string.



Step 2: SearchYouTube Workflow (part 1)



Purpose: Open YouTube with Chrome browser, search for the artist, sort the results by view count

Activities:

1. **OpenBrowser** Chrome – www.youtube.com
2. **Do**
3. **Maximize** the window
4. **TypeInto** for searching the artist name
5. **Click** search button
6. **Click** fiter

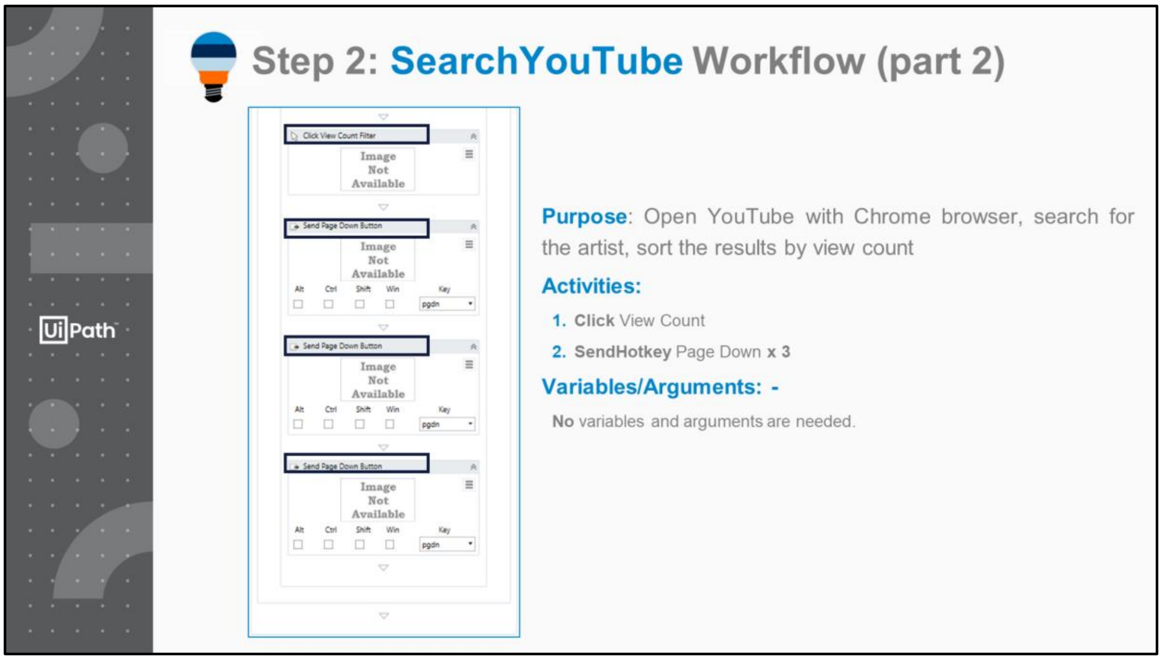
Variables/Arguments:

Argument: ArtistName (type: string, direction in)

The activities used in this first part of the workflow:

1. OpenBrowser for opening YouTube (in our case using Chrome). All the activities will be executed in the browser window.
2. Do, which is the loop statement in our case.
3. Maximize the window (this step is optional, but it will definitely improve the scraping)
4. TypeInto – type the artist name into the search bar. The search bar is indicated on the screen.
5. Click the search button indicated on the screen.
6. Click on filter (the button is indicated on the screen).

For this part of the workflow, only one argument is needed, the name of the artist, taken from the first workflow. The type is also string, but the direction is in.



Step 2: SearchYouTube Workflow (part 2)

Purpose: Open YouTube with Chrome browser, search for the artist, sort the results by view count

Activities:

1. **Click** View Count
2. **SendHotkey** Page Down **x 3**

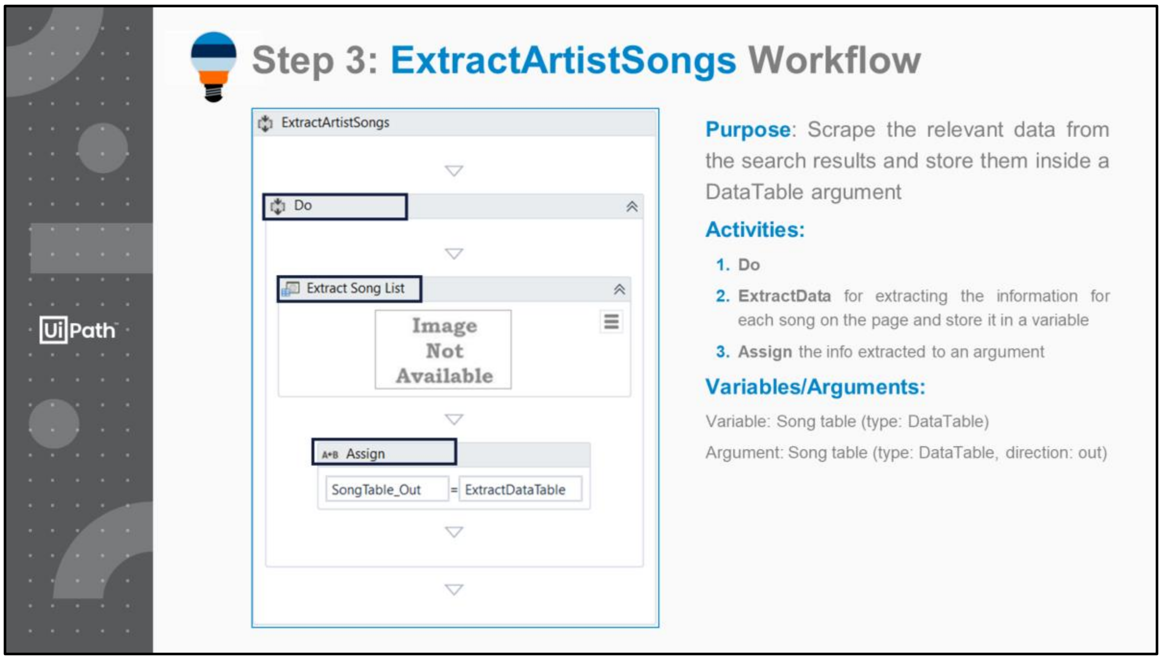
Variables/Arguments: -

No variables and arguments are needed.

The activities used in the second part of the workflow:

1. Click on view count (as filter)
2. SendHotkey to press Page Down 3 times, to increase the number of songs displayed, subject to scraping.

For this part of the workflow, no variables and arguments are needed.



Step 3: ExtractArtistSongs Workflow

Purpose: Scrape the relevant data from the search results and store them inside a DataTable argument

Activities:

1. **Do**
2. **ExtractData** for extracting the information for each song on the page and store it in a variable
3. **Assign** the info extracted to an argument

Variables/Arguments:

Variable: Song table (type: DataTable)

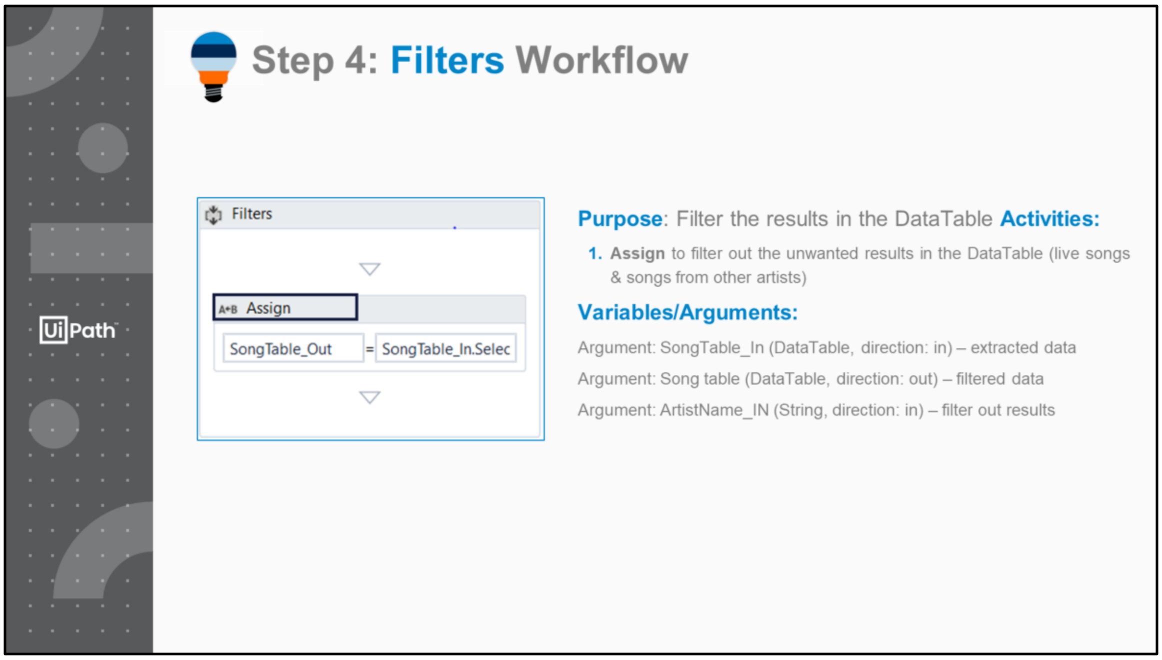
Argument: Song table (type: DataTable, direction: out)

The activities used in this workflow:

1. Do (loop executed for all the results displayed).
2. ExtractData for extracting the song information for the results displayed and storing in a data table variable.
3. Assign the song information in the variable to an argument of same type (DataTable) to be able to move it in the next workflow.

For this part of the workflow, the following variables and arguments are needed:

1. Song table (DataTable Variable) to assign what is extracted using ExtractData.
2. Song table (DataTable Argument, direction: out) to assign what is stored in the variable in order to move to the next workflow.



Step 4: **Filters Workflow**

Filters

Assign

SongTable_Out = SongTable_In.Selec

Purpose: Filter the results in the DataTable **Activities:**

1. **Assign** to filter out the unwanted results in the DataTable (live songs & songs from other artists)

Variables/Arguments:

Argument: SongTable_In (DataTable, direction: in) – extracted data

Argument: Song table (DataTable, direction: out) – filtered data

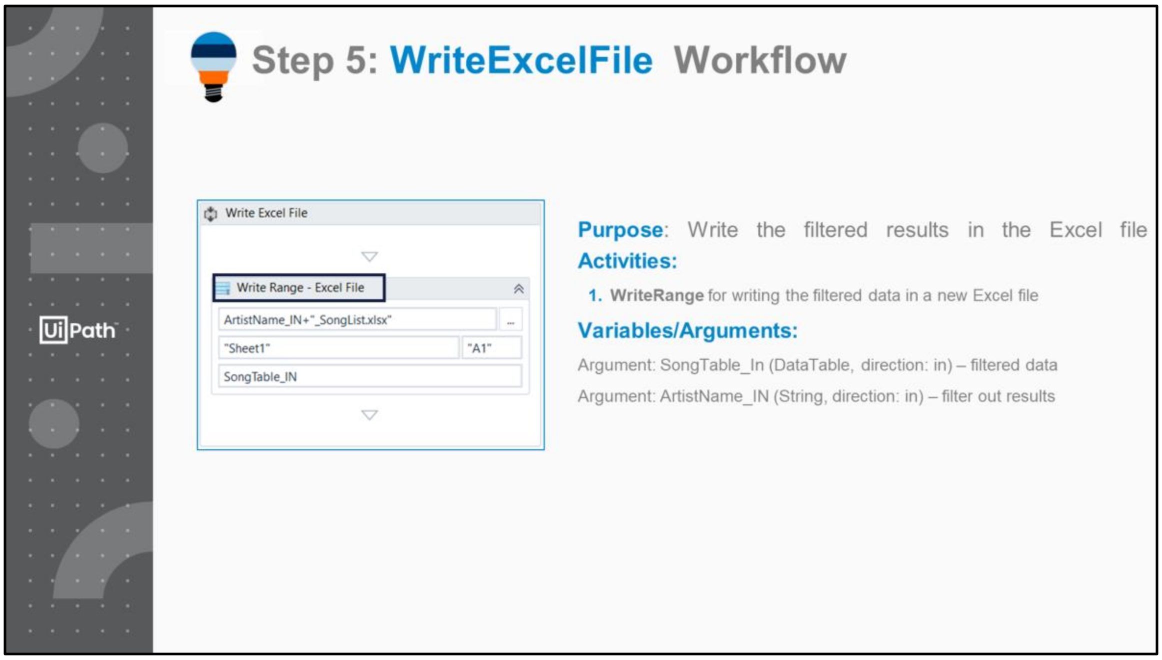
Argument: ArtistName_IN (String, direction: in) – filter out results

The activities used in the Filters workflow:

1. Assign – used to take the data extracted in the previous workflow and stored in the DataTable argument, apply filters to remove live songs and songs of other artists, and assign the filtered data to a new Argument
   The filter used is: SongTable_In.Select("([Song Name] not like '%live%' OR [Song Name] not like '%Live%') and [Song Name] like '%"+ArtistName_IN+"%'").CopyToDataTable

For this workflow, the following variables and arguments are needed:

1. SongTable_In (DataTable argument, direction in) to import the data extracted in the previous workflow.
2. Song table (DataTable argument, direction: out) to assign the filtered data.
3. ArtistName_IN (String argument, direction in) to apply the filter for excluding songs of other artists.



Step 5: WriteExcelFile Workflow

ArtistName_IN+"_SongList.xlsx"
"Sheet1" "A1"
SongTable_IN

Purpose: Write the filtered results in the Excel file

Activities:

1. **WriteRange** for writing the filtered data in a new Excel file

Variables/Arguments:

Argument: SongTable_In (DataTable, direction: in) – filtered data

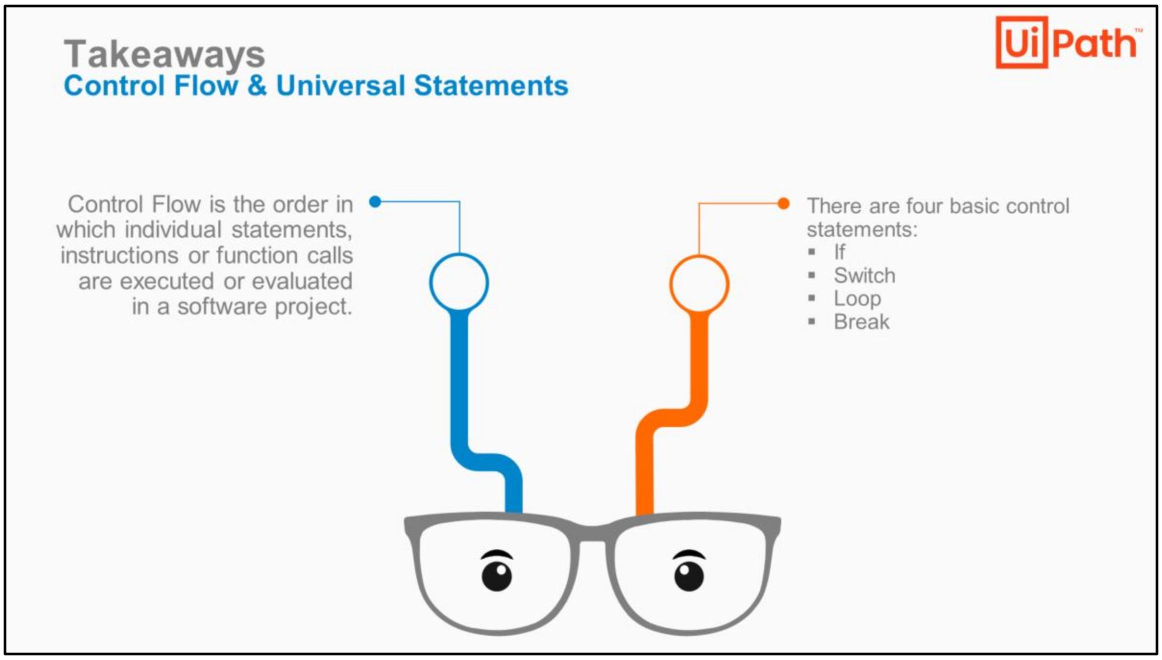
Argument: ArtistName_IN (String, direction: in) – filter out results

The activities used in the Write Excel File workflow:

1. WriteRange – used to write the filtered data stored in the Argument in an excel file.

For this workflow, the following variables and arguments are needed:

1. Song table (DataTable argument, direction: out) to assign the filtered data
2. ArtistName_IN (String argument, direction in) to name the excel file



Takeaways
Control Flow & 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:

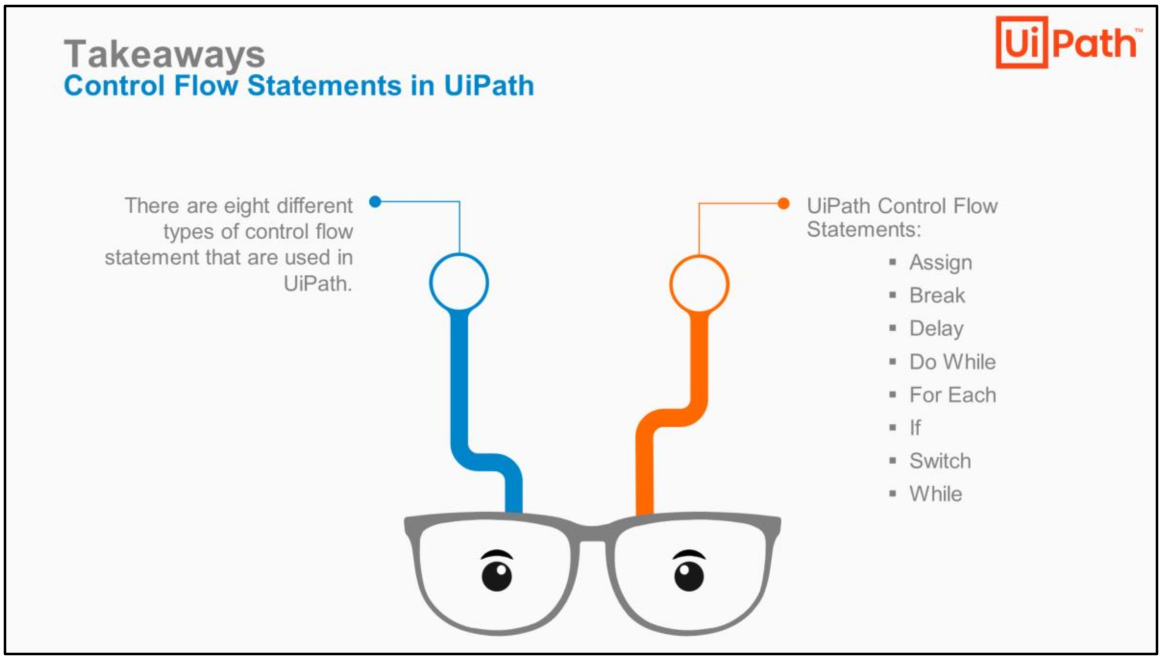
- If
- Switch
- Loop
- Break

The point of the Recap & Summary section is to go through the most important points covered in the lesson, after the students had the chance to see them in practice and obtain a consolidated view.

The teacher should use facilitation questions to help the students map the key points and offer a safe space to get questions and comments from them.

Some examples of facilitation questions:

1. What are the differences between sequences and flowcharts?
2. What would be a case in which a sequence would be preferable to a flowchart? Why?
3. How a project should be so that you would draw it directly as a flowchart?



Takeaways

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

The point of the Recap & Summary section is to go through the most important points covered in the lesson, after the students had the chance to see them in practice and obtain a consolidated view.
The teacher should use facilitation questions to help the students map the key points and offer a safe space to get questions and comments from them.
Some examples of facilitation questions:

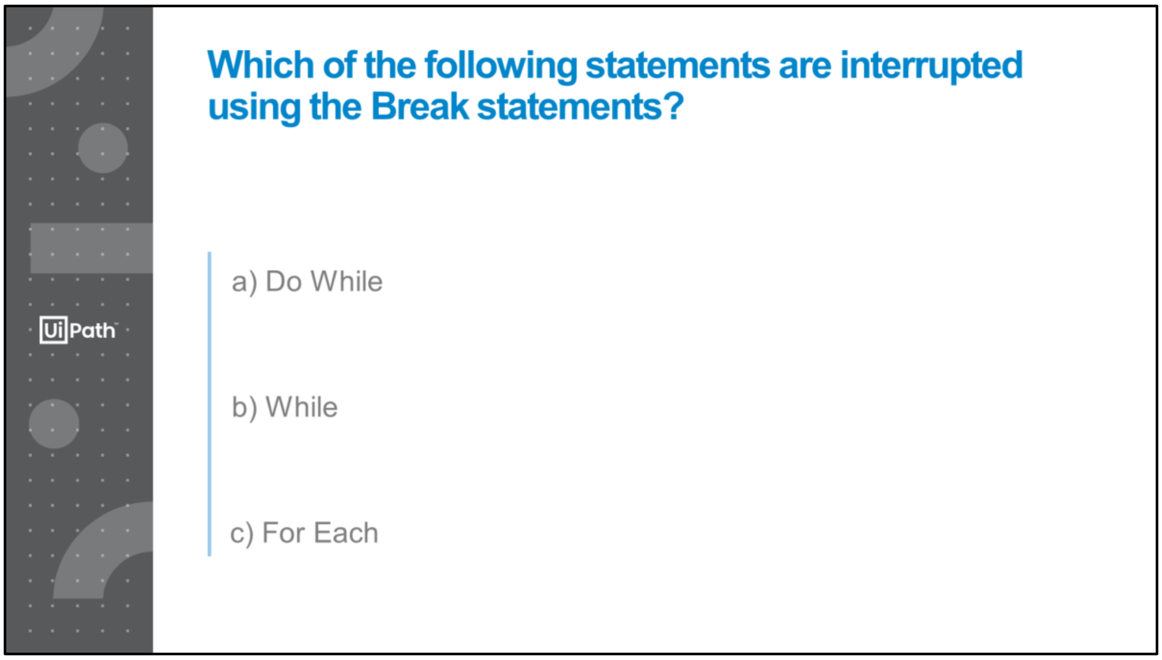
Generic statements:
1. Why is it called Control Flow?
2. Why is the **If** statement considered the quintessential control flow statement?
3. Since the **Switch** statement can be built with only 2 cases, why do we still need the If statement?
4. Since **Break** can be used to interrupt a loop, why do you think it's not just a part of the **loop** statement (and nothing else)? Why do we have Loop and Break and not Loop-Break statement?

Specific statements:
1. Why is assigned considered a control flow statement, given that it has nothing to do with the 4 generic statements?
2. Why is **Break** needed with **For Each**? Why doesn't **For Each** stop once it goes through the list?
3. Is there any other use for the Delay statement than specifying a duration to pause the execution of the main flow?

4. Are **Do While** and **While** absolutely interchangeable or are there specific cases in which only one of them should be used?
5. What is an example in which **For Each** is the only option to solve a certain challenge?
6. How is the **If** generic statement different than the UiPath If statement?
7. If a condition is Boolean, can it be transformed in an expression for a Switch statement?

Questions & Answers

Q&A
Now it’s your turn. What’s on your mind at the end of this?



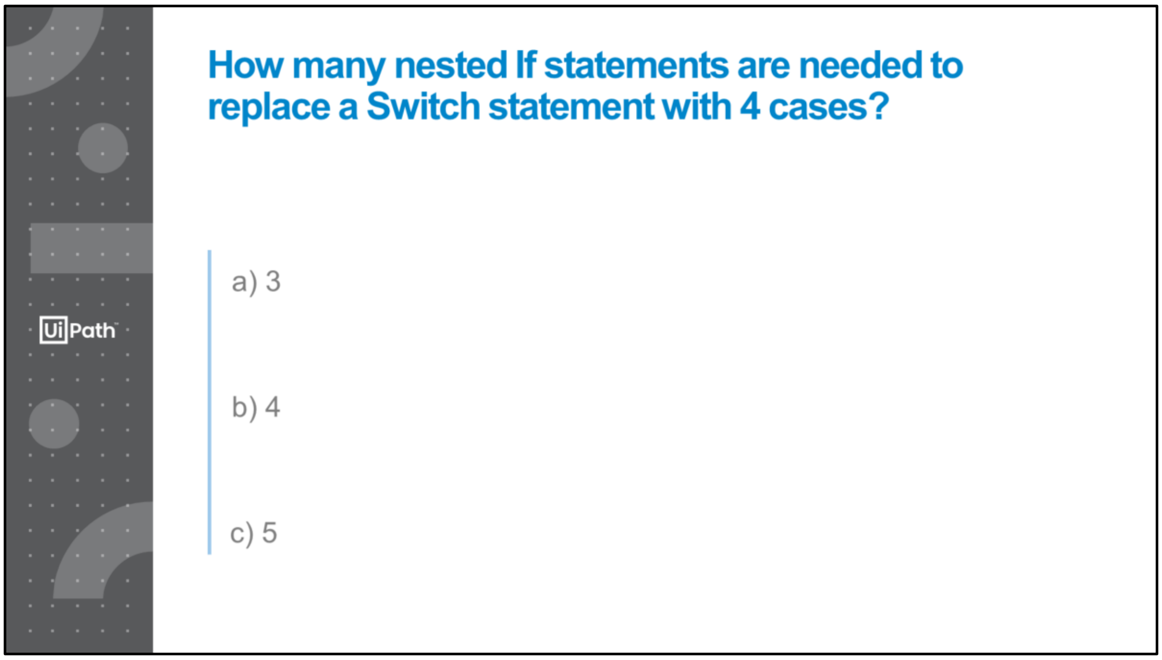
Which of the following statements are interrupted using the Break statements?

a) Do While

b) While

c) For Each

Correct answer: c) For Each



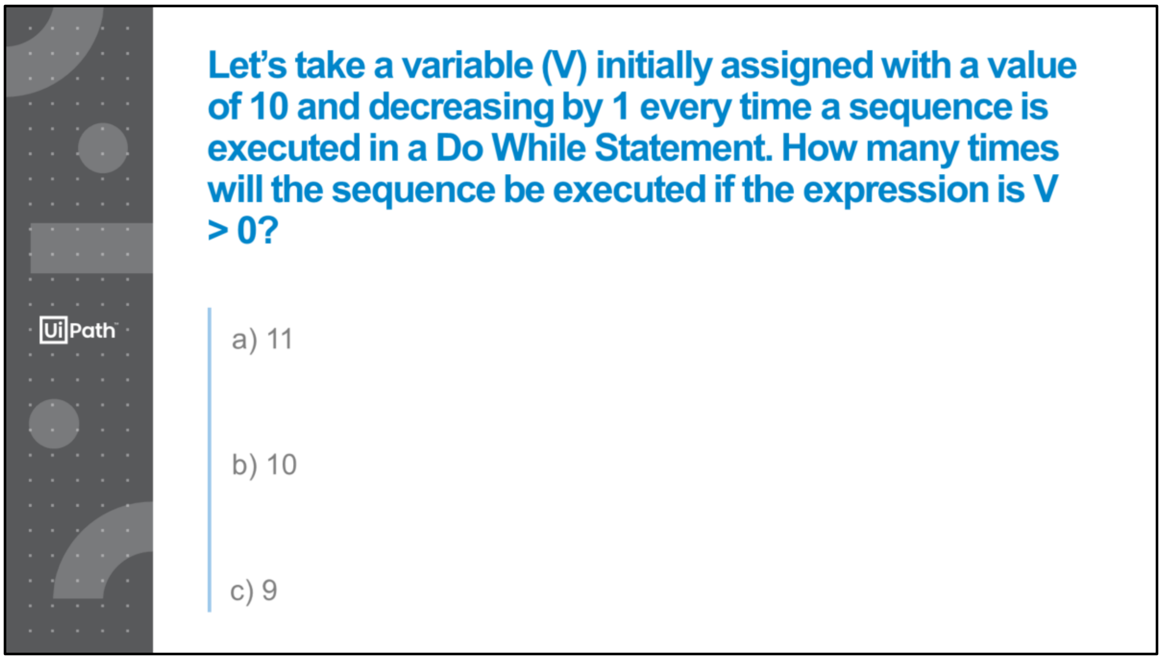
How many nested If statements are needed to replace a Switch statement with 4 cases?

a) 3

b) 4

c) 5

Correct answer: a) 3



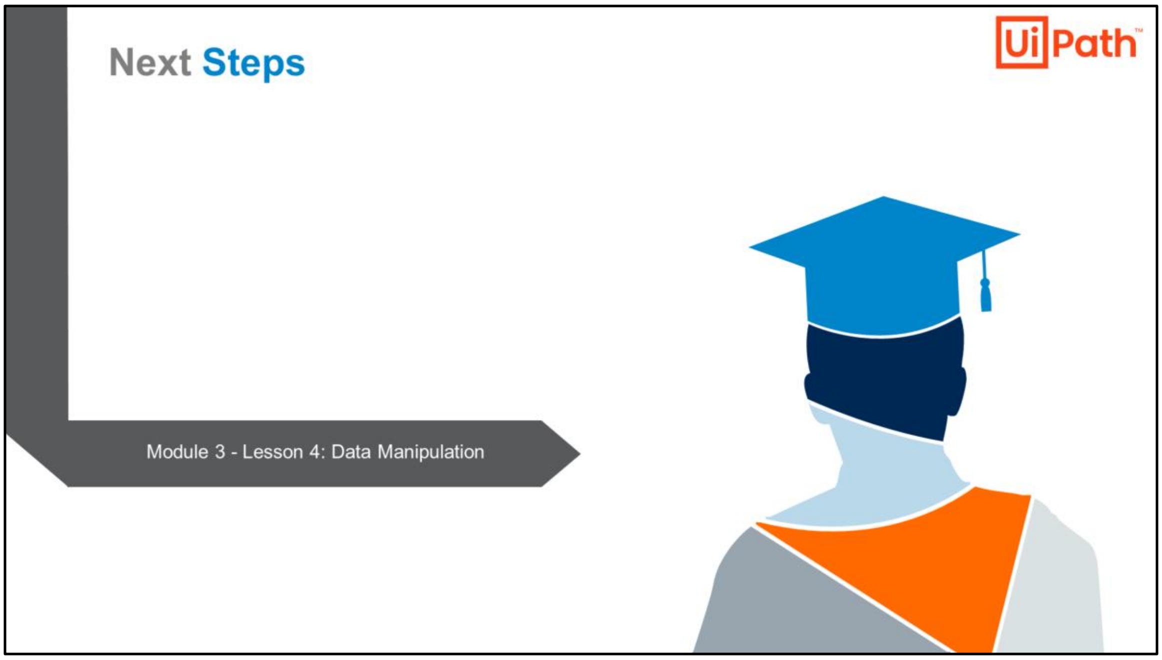
Let’s take a variable (V) initially assigned with a value of 10 and decreasing by 1 every time a sequence is executed in a Do While Statement. How many times will the sequence be executed if the expression is V > 0?

a) 11

b) 10

c) 9

Correct answer: b) 10



Next Steps

Module 3 - Lesson 4: Data Manipulation

In the next lesson, we will be covering Data Manipulation.