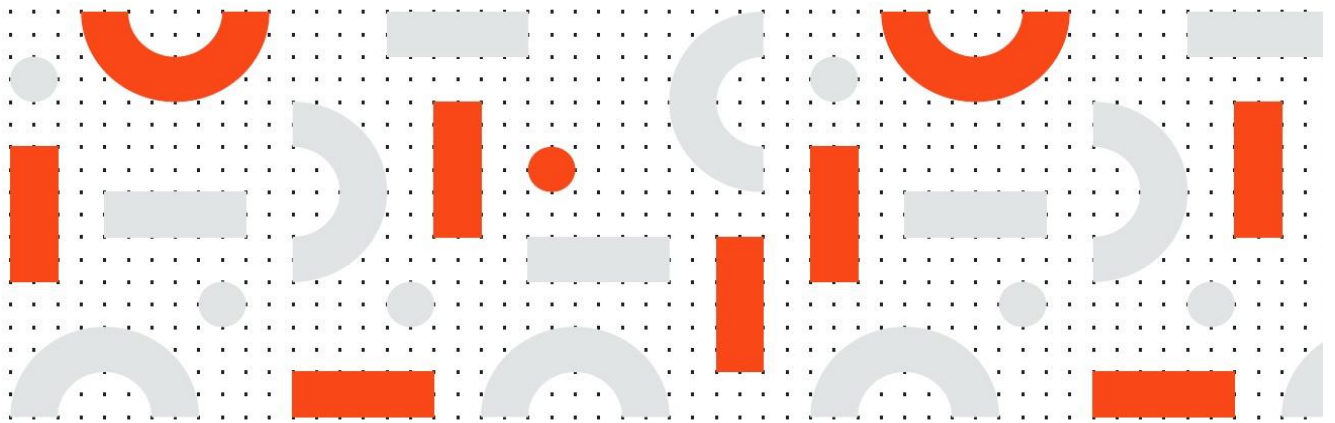


RPA Design & Development V2.0

Student Manual






Welcome to 'RPA Design and Development Course'.

Lesson 3: Variables and Arguments

The third lesson of this course is Variables and Arguments.

Agenda



01

Variables and their Types

02

Variables Panel

03

Scope of Variable

04

Arguments

05


Arguments Panel


06


Argument Directions


07


Arguments vs. Variables








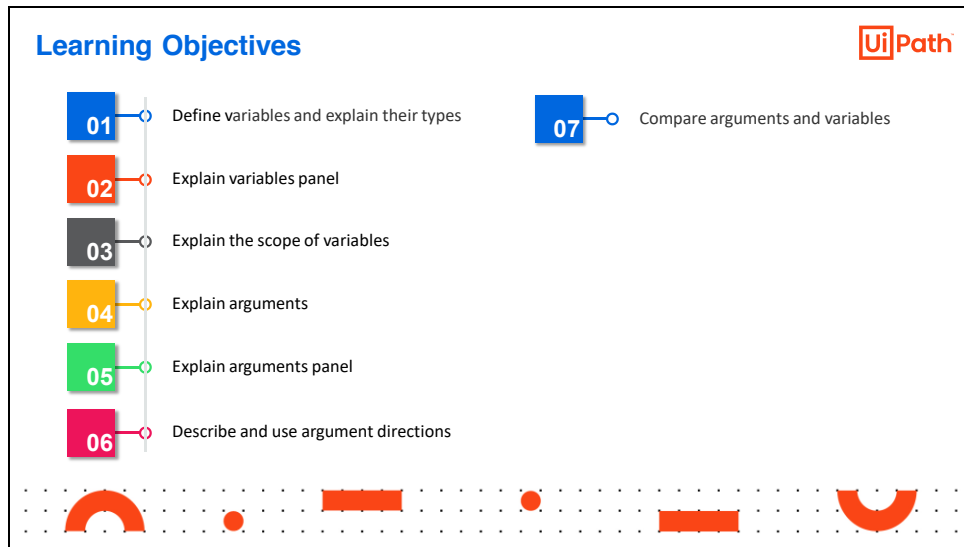






The agenda of this lesson is:

- Variables and their Types
- Variables Panel
- Scope of Variable
- Arguments
- Arguments Panel
- Argument Directions
- Arguments vs. Variables

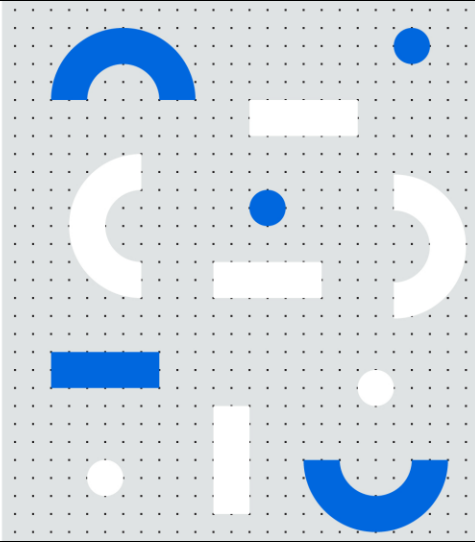


By the end of this lesson, you will be able to:

- Define variables and explain their types
- Explain variables panel
- Explain the scope of variables
- Explain arguments
- Explain arguments panel
- Describe and use argument directions
- Compare arguments and variables

Variables and their Types

- Introduction to variables
- Properties of variables
- Best practices for naming variables
- Types of variables in UiPath





This section introduces variables and their types in UiPath.

Introduction to Variables

Variables are containers that are used to store multiple types of data. A variable:

- makes it easier to label and store data which can later be used throughout the automation process
- has an initial value which may change during the program through external input, data manipulation or passing from one activity to another
- is like a box that stores data
Example: A box (variable) named Counter that tracks the number of times users clicked on an item



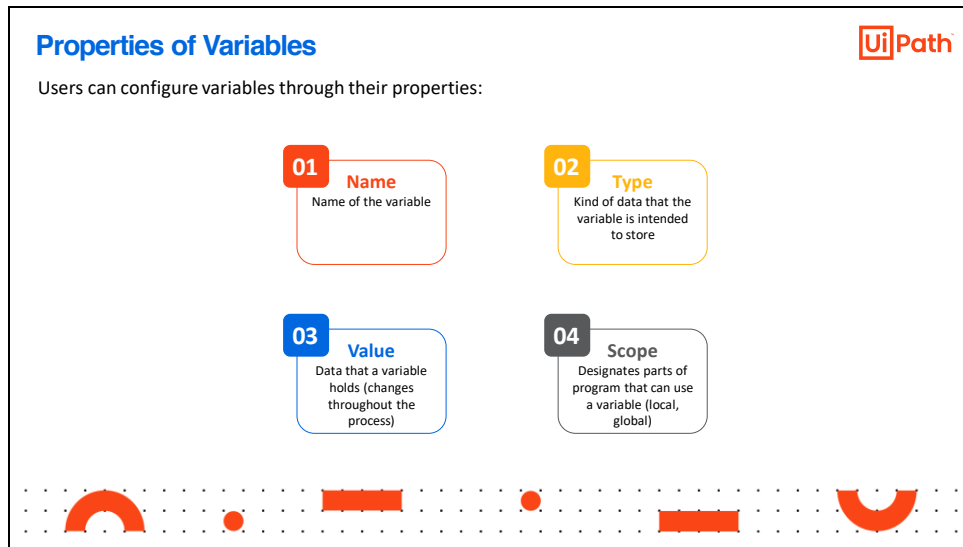
Variables are containers that are used to store multiple types of data. Using variables makes it easier to label and store data which can later be used throughout the automation process.

A variable has an initial value which may change during the program. The value of a variable can change through external input, data manipulation or passing from one activity to another. A collection variable contains multiple data entries (of same type).

It's useful to think of a variable like a box that stores data.

For example:

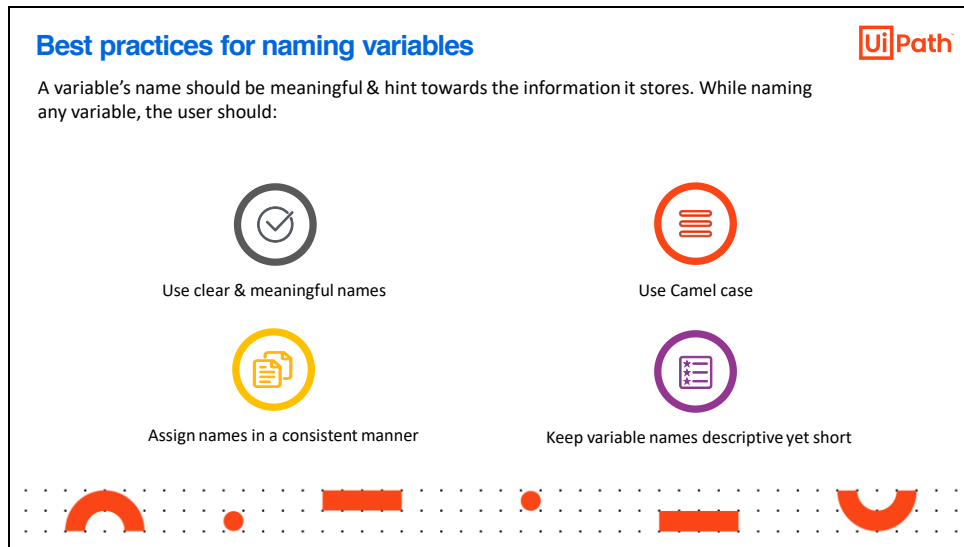
- A box (variable) named Counter that tracks the number of times users clicked on an item.



The user can configure a variable through its properties. The variables have following properties:

- **Name:** The name of a variable is its unique ID. It represents the "title" of the information that is being stored by the variable. A variable should be named to represent all possible values that it might contain. The name is essential in programming, because the user "accesses" the variable through the name. Every variable must have a unique name and it should be descriptive enough so that other users can easily read it and save time.
- **Type:** Type defines the kind of data that the variable is intended to store. The type is declared when the variable is created. There are some specific types that are more generic and can accommodate different types of data. The most common (universal) types of a variable are: text, number, array, date/time, Boolean and data tables.
- **Value:** Value is the data that a variable holds. The value of a variable changes throughout the process. Suppose a variable named 'tasks' is assigned a value '12' initially. At another point, 'tasks' may be assigned the value '27'. If no initial value is assigned at the creation of the variable, a default rule assigns a value to the variable.
- **Scope:** Scope defines from where the variable can be accessed in the workflow. The variables can be global or local. When the user defines a variable for the first time (through Variables Panel), it has a global scope, i.e., it can be accessed from every part of the program. If the user defines a variable with local scope, then outside processes cannot

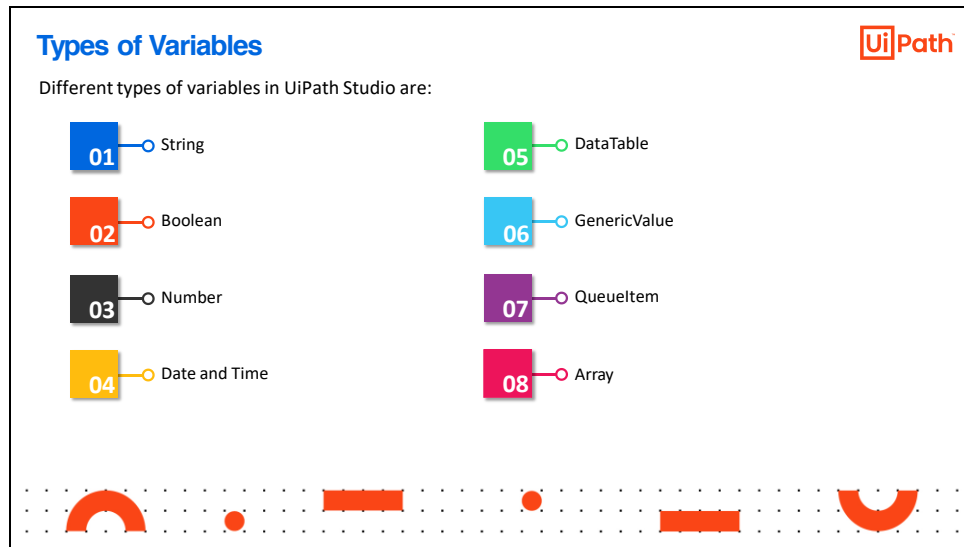
access it. There are always several variables in use, so the users should avoid making multiple variables unnecessarily global as it may cause efficiency issues.



The name of a variable should be meaningful and describe the information it stores. This makes it easier to understand the purpose of the variable and to maintain it.

For this, the users must follow the best practices for naming a variable and should:

- Use clear & meaningful names (the name of the variable should accurately describe its content)
- Assign names in a consistent manner
- Adhere to using upper Camel Case, e.g. FirstName, LastName
- Keep variable names descriptive yet short

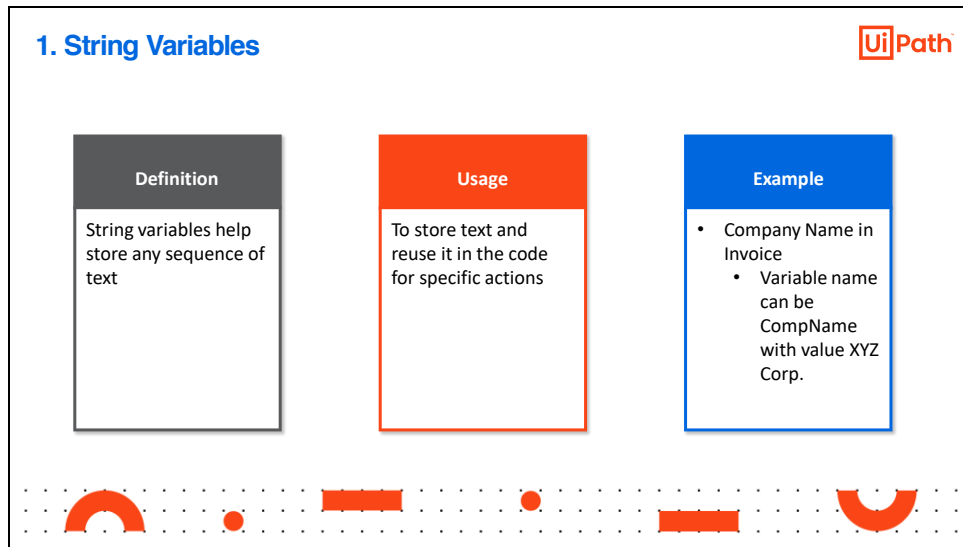


The type of a variable defines the type of data that can be stored inside the variable. The different types of variables available in UiPath Studio are:

- String variable
- Number variable
- Date and Time variable
- Data Table variable
- Boolean variable
- GenericValue variable
- QueueItem variable
- Array variable

These are discussed in detail in the subsequent slides. In addition to these, for complex situations, the users can declare data types of **.Net type** variables which are also available in UiPath.

Also, in addition to these, there are two collection variables: Lists and Dictionaries which will be discussed in Lesson 6 of this course.



String (Text) Variables are variables in which the users can store only strings, i.e., a sequence of text. Arithmetic operations cannot be applied on them. These variables can store information like usernames, employee names, etc.

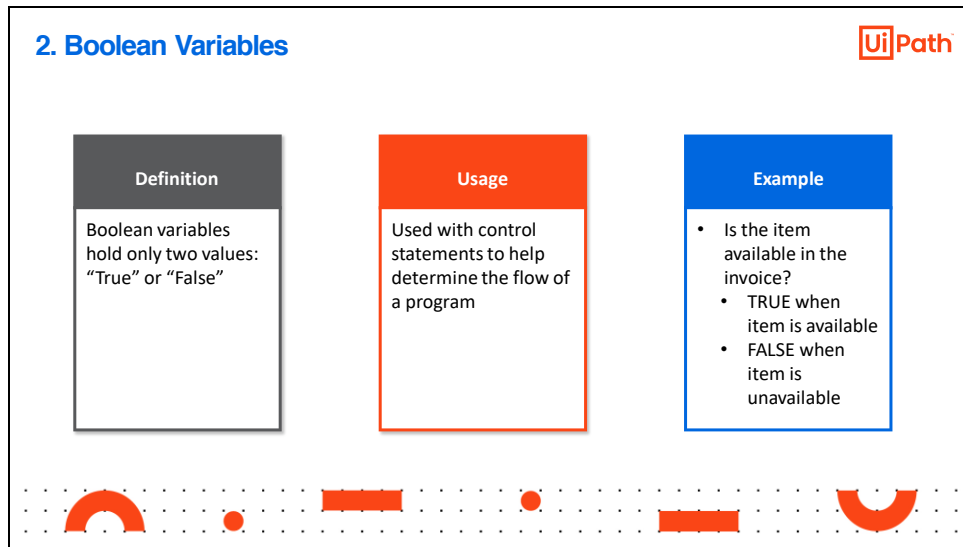
String variables are used to store text and reuse it in the code for specific actions. It can name of a person, address of a person, etc.

Example:

Consider that an invoice is being generated. There are various fields mentioned in the invoice. These fields are represented using different variables. Here, the name of the company is an example of string variable as it takes a string value. (Company Name could be XYZ Corp.)

- Variable name can be **CompName** with value XYZ Corp.

Refer <https://docs.uipath.com/studio/docs/text-variables> for example of using a String variable.



A Boolean variable (also known as true-false variable) has only two possible values: "True" or "False".


The Boolean variable is primarily associated with conditional statements, which allow different actions by changing control flow depending on whether a programmer-specified Boolean condition evaluates to true or false. Thus, Boolean variable is commonly used with control statements to determine the flow of a program as they enable the user to make decisions. A variable holds TRUE when a condition is met, FALSE when a condition is not met.

Example:


Holds TRUE when the item in the invoice is available, holds FALSE when the item is unavailable.

Refer <https://docs.uipath.com/studio/docs/true-or-false-variables> for example of using a Boolean variable.

3. Number Variables



Definition	Usage	Example
Number variables store numeric values	To execute equations or perform comparisons, pass important data, etc.	<ul style="list-style-type: none">Item Quantity<ul style="list-style-type: none">Variable name can be ItemQuant with value 50



Number variables (also known as integer) are the variables in which the user can store numeric values. They are used to execute equations or perform comparisons, pass important data, etc. Number variables specify values such as age of a person: (eg: 34)


Example:

Item Quantity.


- Variable name can be **ItemQuant** with value 50.

Refer <https://docs.uipath.com/studio/docs/number-variables> for example of using a Number variable.

4. Date and Time Variables



Definition	Usage	Example
Date and time variables store information about any date and time	Used to calculate the number of days between two dates, store current date details, etc.	<ul style="list-style-type: none"> Invoice Date Variable name can be <code>InvDate</code> with value <code>01/01/2020</code>



The date and time variable is a type of variable that enables you to store information about any date and time such as a variable **Today** holds today's date.

The main usage of Date & Time variable is to store a specific date and time inside or to perform other operations with them. (Ex. Calculate how many days are left until the end of the month). They may be used to append dates to time-sensitive invoices or documents.


Example:

Invoice Date

- Variable name can be **InvDate** with value `01/01/2020`.

Refer <https://docs.uipath.com/studio/docs/date-and-time-variables> for example of using Date & time variable.

5. DataTable Variables



Definition	Usage	Example
<p>DataTable variables store tabular data in rows & columns and may hold large pieces of data & act as a database</p>	<p>Used to migrate data from a database to another, extract information from a website and store it locally in a spreadsheet</p>	<ul style="list-style-type: none"> List of all items in the invoice Variable name can be <code>InvItem</code> with values <code>Item1</code>, <code>Item2</code>, <code>Item3</code>,....

DataTable variables can store tabular data in rows & columns and may hold large amount of information and act as a database. In UiPath Studio, DataTable variables can be found in the Browse and Select a .Net Type window, under the System.Data namespace.

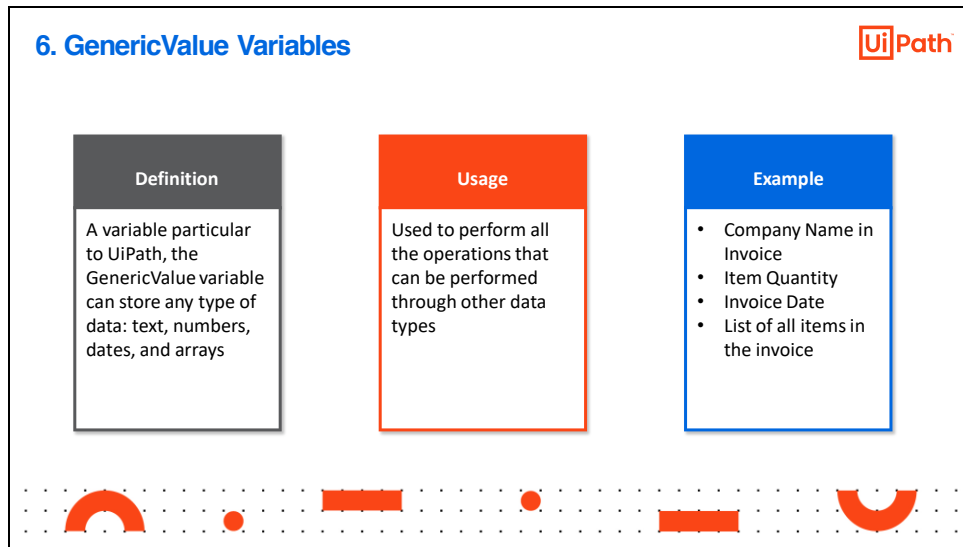
The practical usage of data tables is to store big pieces of information and do certain operations on it, such as, filtering, searching, copying, etc. They are often used to migrate data from a database to another, extract information from a website and store it locally in a spreadsheet.

Example:

List of all items in the invoice with quantity of each

- Variable name can be **InvItem** with values `Item1`, `Item2`, `Item3`,....

Refer <https://docs.uipath.com/studio/docs/data-table-variables> for example of using DataTable variable.



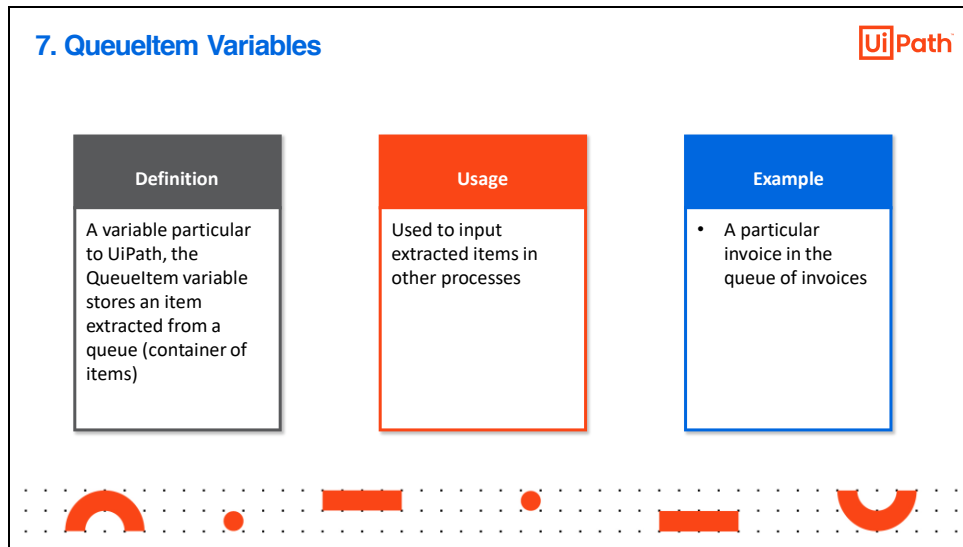
The GenericValue variable can store any type of data: text, numbers, dates, and arrays. It is a proprietary variable of UiPath.

It automatically converts to other types to perform certain actions through an automatic conversion mechanism. The user can receive the desired outcome by carefully defining their expressions. The first element in the expression is used as a guideline for the operation that Studio performs.

Example: Can store any type of data such as

- Company Name in Invoice (Variable name can be **CompName** with value XYZ Corp)
- Item Quantity (Variable name can be **ItemQuant** with value 50)
- Invoice Date (Variable name can be **InvDate** with value 01/01/2020)
- List of all items in the invoice (Variable name can be **InvItem** with values Item1, Item2, Item3,...)

Refer <https://docs.uipath.com/studio/docs/genericvalue-variables> for example of using GenericValue variable.



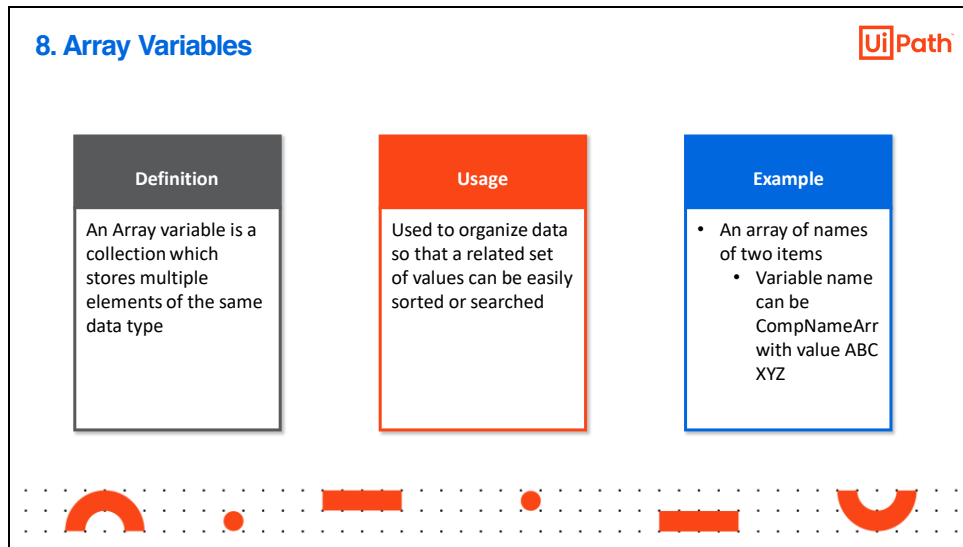
The QueueItem variable stores an item which has been extracted from a container of items, a queue. Through this extraction, the user can further use the queue items in other processes. It is a proprietary variable of UiPath.

For example: An employee in a bank handling deposits' report of 50 clients wants to process the data of each client who had previously deposited an 'On-Us' Check smaller than \$5000, using a third-party financial application. Here, the user can create a workflow in which the QueueItem variable stores, by turn, the information of each client. Then each entry read by the QueueItem variable is stored in a queue item and added to the designated queue in Orchestrator. This results in a queue which contains the clients' information, each in the corresponding queue item.

Example:

Consider that several invoices are being generated and placed in a queue to be processed. Any particular invoice in this queue can be extracted using QueueItem variable.

Refer <https://docs.uipath.com/studio/docs/queueitem-variables> for example of using QueueItem variable.



An array is a data structure that contains a group of elements of the same data type, such as an integer or string. So, an Array variable enables users to store multiple values of the same data type.

Array is a type of collection variable used to store multiple entries. The size of an array is defined when it is created, and it is not dynamic.


A search engine may use an array to store Web pages found in a search performed by the user. When displaying the results, the program will output one element of the array at a time. This may be done for a specified number of values or until all the values stored in the array have been output.

While the program could create a new variable for each result found, storing the results in an array is more efficient way to manage memory.


Example:

- Variable name can be **CompNameArr** which is an array of two strings: Company Name 1 and Company Name 2
(Based on previous examples of String Variables, the value of this array could be ABC XYZ).

Refer <https://docs.uipath.com/studio/docs/array-variables> for example of using an Array variable.

Array vs. String 

Array	String
It is a sequential collection of elements of similar data types.	It is a sequence of single characters represented as a single data type.
Its elements are stored contiguously in increasing memory locations.	It can be stored in any manner in memory locations.
It is a special variable that can hold more than one value at a time.	It can hold only character data.
It is mutable (the fields can be modified).	It is immutable (the value cannot be changed in memory once created).
Its length is predefined.	Its size is not predefined.



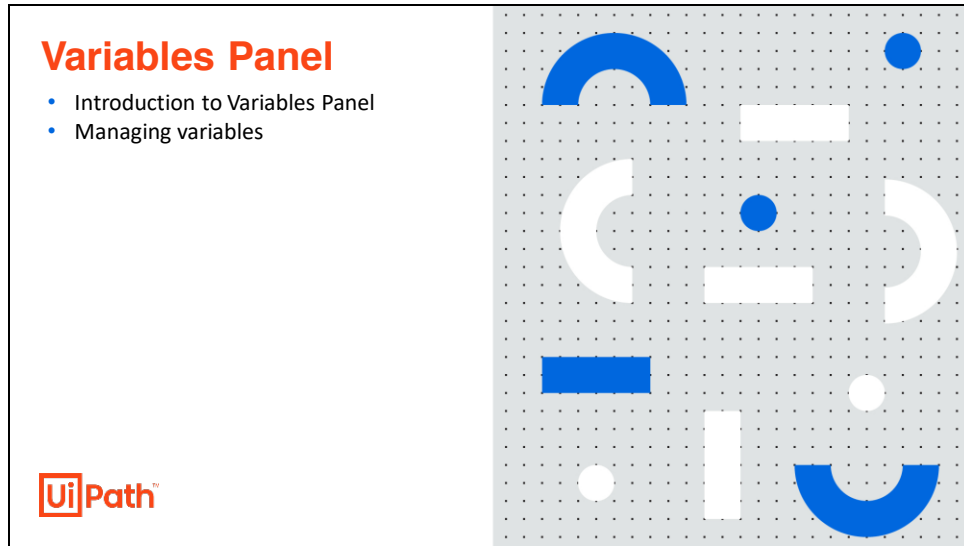
Array:

- It is a fixed-size sequenced collection of elements of the same base types that share a single name and can be used to represent a list of names or numbers.
- Its elements are stored contiguously in increasing memory locations.
- It is a special variable that can hold more than one value at a time.
- It is mutable (the fields can be modified).
- Its length is predefined.

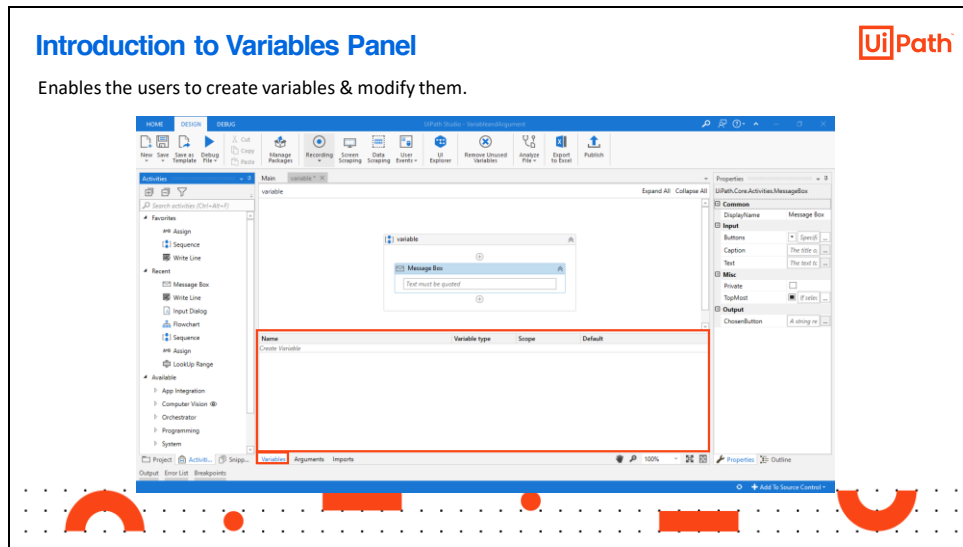
A string is similar to an array with a few exceptions.

String:

- It is a sequence of single characters represented as a single data type.
- It can be stored in any manner in memory locations.
- It can hold only character data.
- It is immutable (the value cannot be changed in memory once created).
- Its size is not predefined.



This section gives an overview of the Variables panel and how to manage variables.



Studio has a **Variables** Panel that enables the users to create variables and modify them.

The panel is located at the bottom right side of the Activities and it is hidden by default. When clicked, the panel opens and displays the existing variables in the project and allows the user to modify them. Whenever a user changes the name of a variable in this panel, it automatically gets updated at all occurrences in the current file.

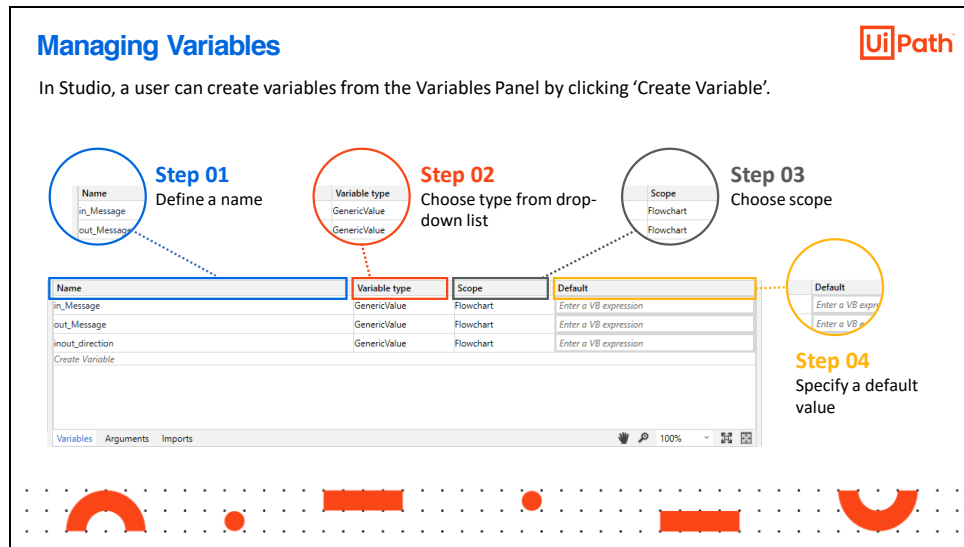
The Variables Panel consists of fields through which the users can define the different properties of a variable. For both existing and new variables, it has the following fields:

- **Name:** The user defines a variable by giving it a name. It is the title of the data which If the user renames a variable in the Variables Panel, it will be renamed in all instances where it's used. This is useful in complex workflows. If the user does not add a name to a variable, it is automatically generated.
- **Variable Type:** The user specifies the data type that the variable will store. The data type for the variables is chosen from the drop-down list. It can be String, Array, GenericValue, number, etc. Also, the user can change the variable type library by clicking Browse types and then choosing a variable library.
- **Scope:** The user specifies the region in which a variable is visible. When the user defines any variable scope in the container, then it is available only in those containers where you define

it. When the user defines a variable scope in the main class, then it is available in all containers. By default, a variable is available in the entire project.

- **Default:** The user specifies the initial value of the variable in 'Default' field. If this field is empty, the variable is initialized with the default value of its type. For example, for an Int32, the default value is 0.

The Variables Panel also allows the users to copy variables from one workflow to another and add/edit comments to a variable.



In order to manage the variables in UiPath, the users needs to perform several actions on them. These are:

- Creating variables
- Initializing variables
- Removing variables

• **Creating Variables:**

There are 3 ways to create variables in UiPath:

- **From the Variables panel** – Open the Variables panel, select 'Create Variable', and fill in the fields as needed. (The variable is created, and its default type is String.)
- **From the Designer panel** – Drag an activity from Activities panel to Designer panel (Drag an activity with a variable field visible (i.e. 'Assign')) and press Ctrl+K. Name it and then check its properties in the Variables panel. (The variable is created, and its type is according to the activity.)
- **From the Properties panel** – In the Properties panel of the activity, place the cursor in the field in which the variable is needed (i.e. Output) and press Ctrl+K. Name it and then check its properties in the Variables panel. (The variable is created, and its type is according to the selected property.)

(The user can create a variable only when the Designer panel contains at least one activity)

2. Initializing variables

While creating a variable, its properties need to be initialized. This is known as initialization of variable. The user can create a variable by following the steps mentioned earlier and initialize it as follows:

1. Define a name for the variable (one that defines the variable and will be easy to remember later in the automation workflow)
2. Choose the type of the variable from the drop-down list. This list displays the types that a user has mostly used. In case a different type is needed, the user can click on "Browse for types..." (The last option in the drop-down list) which opens a separate screen of project window, Browse and Select a .Net Type, where the user can use keywords to search for the desired type. Click on it to add the type.
3. Choose the scope (available only in the sub-process or for the entire workflow)
4. Now choose a default value.

When a string variable is assigned a value that contains quotations, users need to use each quotation character twice; otherwise, it will be considered as a string delimiter.

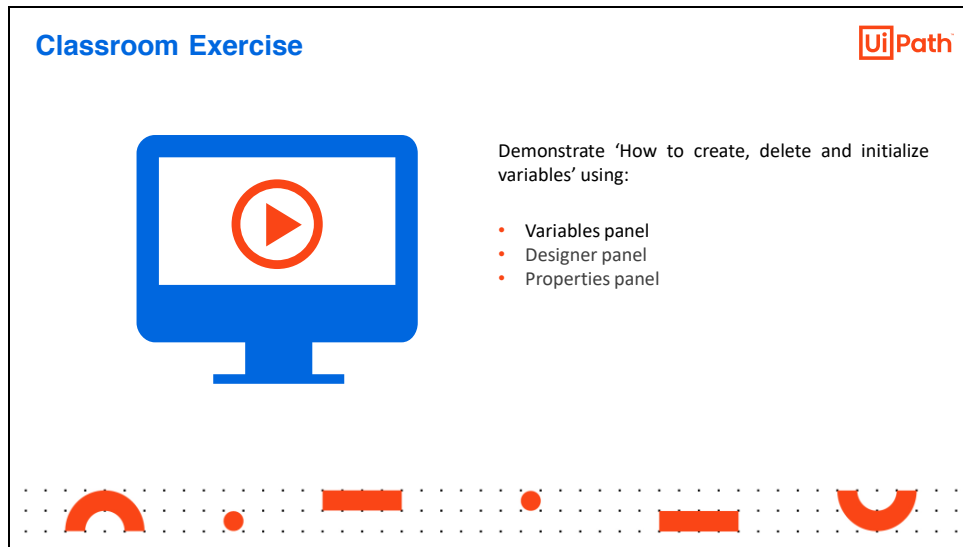
Collection variables: The process of creating collection variables is similar to the process of creating simple variables (numbers, characters, strings, etc.). However, for initializing collection variables, the user needs to specify the type of collection variable (it may be Strings, Integers, etc. and then, it will create an Array of strings, a List of Integers and so on respectively).

3. Removing variables

A user can follow either of the following steps to remove a variable:

- Open the Variables panel, right-click a variable and select the Delete option.
- Open the Variables panel, select a variable and press the Delete key.

To remove all the variables available in the Variables panel, click Remove Unused Variables on the Design ribbon tab. This removes the variables not used in the currently opened project and also the ones that are unused but mentioned in annotations.



Open UiPath Studio and demonstrate 'how to create, delete and initialize variables' with the use of Variables, Designer and Properties panel by following the below mentioned steps:

Manage variables through Variables panel:

- In the Designer panel, click Variables tab. Variables panel is empty.
- Drag and drop Sequence activity in the designer panel.
- In the Variables panel, click Create variable and enter a variable name, say Test1.
- Right click on this variable and select Delete to delete this variable.

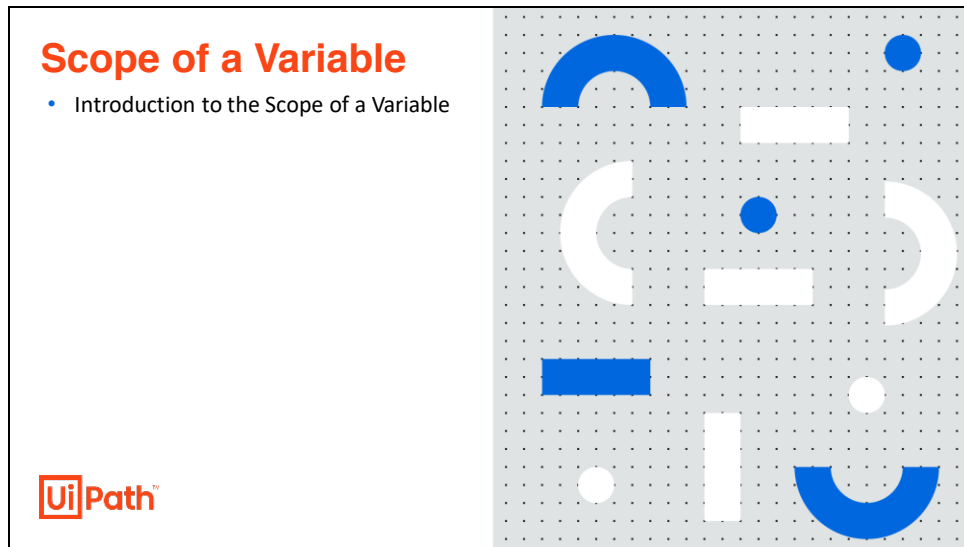
Create variables in Activities within Designer panel:

- Drag and drop two Message Box activities within the Sequence activity.
- Right click in the text area of the first Message Box activity and select Create Variable from the context menu.
- Enter variable name, say Int1, after Set Var.
- In the Variables panel, change its Variable type to Int32 for integer, and enter 1 as its default value.

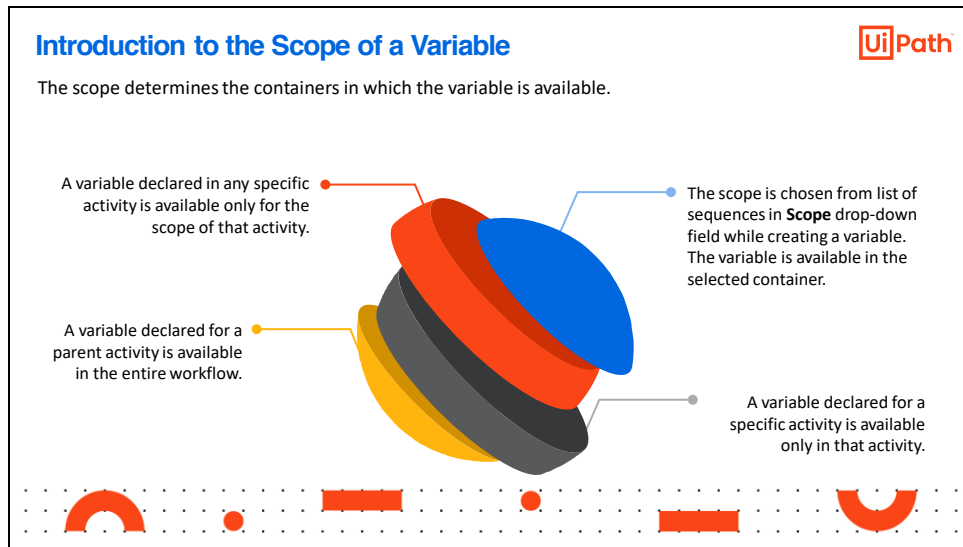
Manage variables through Property panel:

- Click second Message Box activity.
- Right click its Text property from Properties panel

- Select Create Variable from the context menu.
- Enter variable name, say String1, after Set Var.
- In the Variables panel, change its Variable type to String, and enter “Variables Panel” as its default value.
- Run the program to get the output of the integer and the string variable.



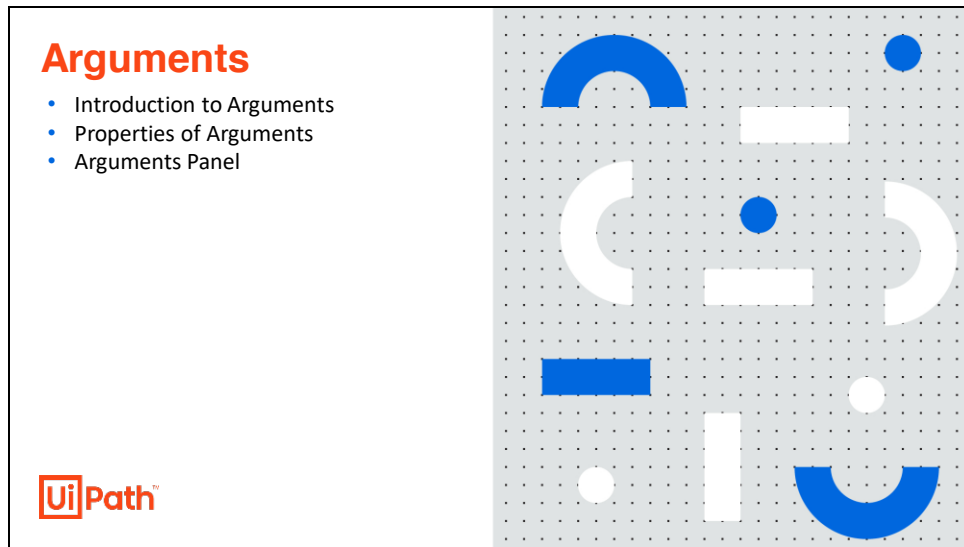
This section gives an overview of the scope of a variable.



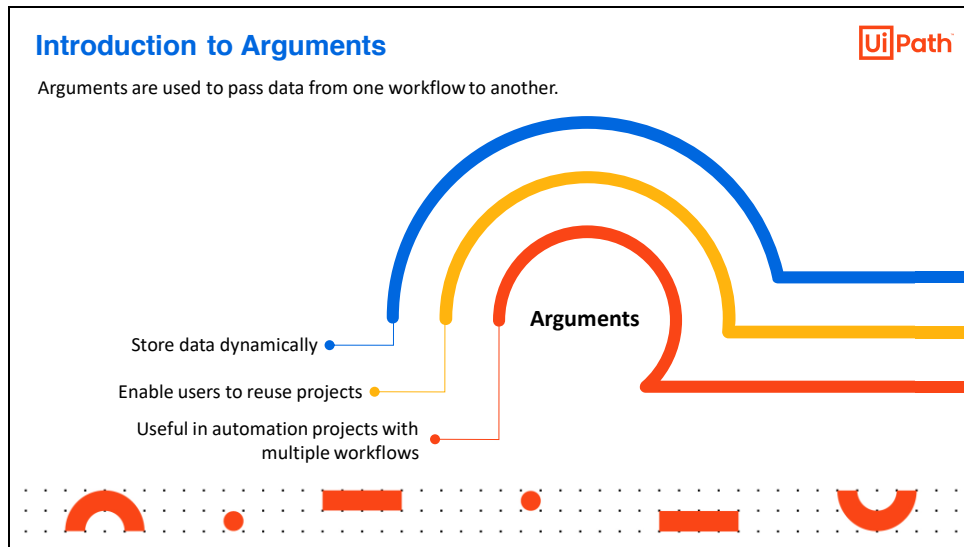
Scope field determines the containers in which the variable is available. When the variable is created, user chooses the scope from the **Scope** drop-down field where all major sequences & sections of the program are listed. The user chooses one of these sequences and the variable will then be available in that part of the workflow. The scope of the variable is limited to the selected container.

If a variable is declared for a parent activity, then the variable is available in entire workflow. However, if the variable is declared in any specific activity then it is available only for the scope of that activity. The scope of a variable cannot exceed the workflow in which it was defined.

The **scope** of each variable should be defined correctly. The variables should be defined only in the scope in which they are used. Making multiple variables unnecessarily available for entire workflow can cause efficiency issues as well as possibility for confusion.

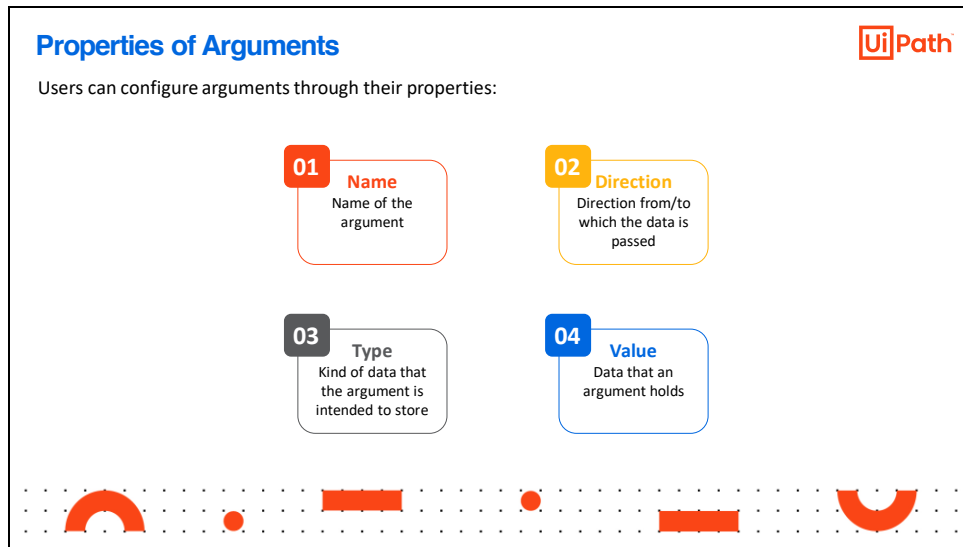


This section gives an overview of arguments in Studio.



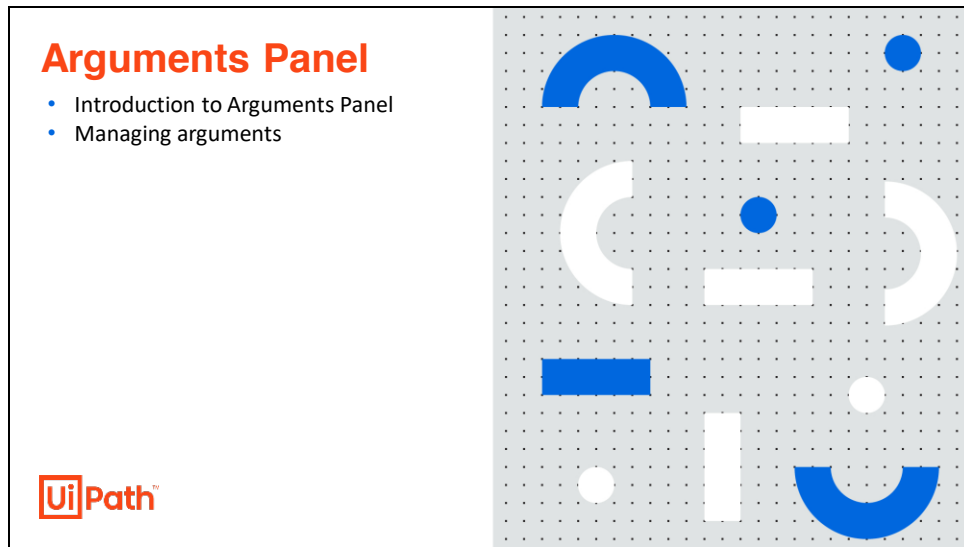
Arguments are used to pass data from one workflow to another by storing the data dynamically.

They are similar to variables as they store data and pass it on. However, the variables pass data between activities, while arguments pass data between workflows. So, arguments enable the users to reuse automation projects. Hence, they can be accessed outside the workflows in which they are defined and are useful in automation projects consisting of multiple workflows.

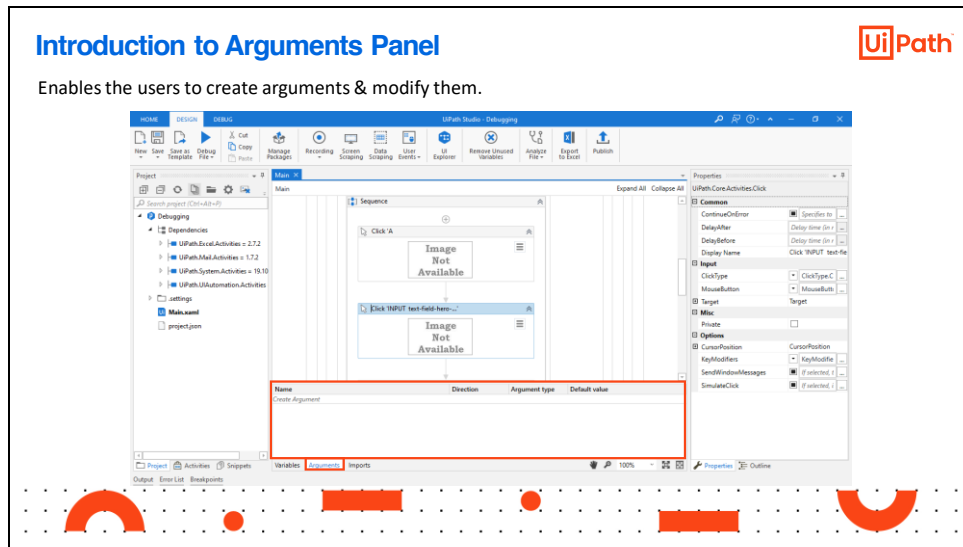


The user can configure an argument through its properties which are similar to that of variables (except for one, i.e., argument direction). The arguments have following properties:

- **Name:** This specifies the name of the argument. It represents the "title" of the information that is being stored by the argument. If the user does not add a name to an argument, it is automatically generated.
- **Direction:** As arguments pass data between workflows, the users need to specify the direction from/to which the data is passed. It can be In, Out and In/Out (explained in next section).
- **Argument Type:** Type defines the kind of data that the argument is intended to store. In UiPath, the type is declared when the argument is created. The types of arguments are: string, number, array, date/time, Boolean and data tables.
- **Default Value:** Value is the data that an argument holds. (Must be provided in English)



This section gives an overview of Arguments panel and how to manage arguments.

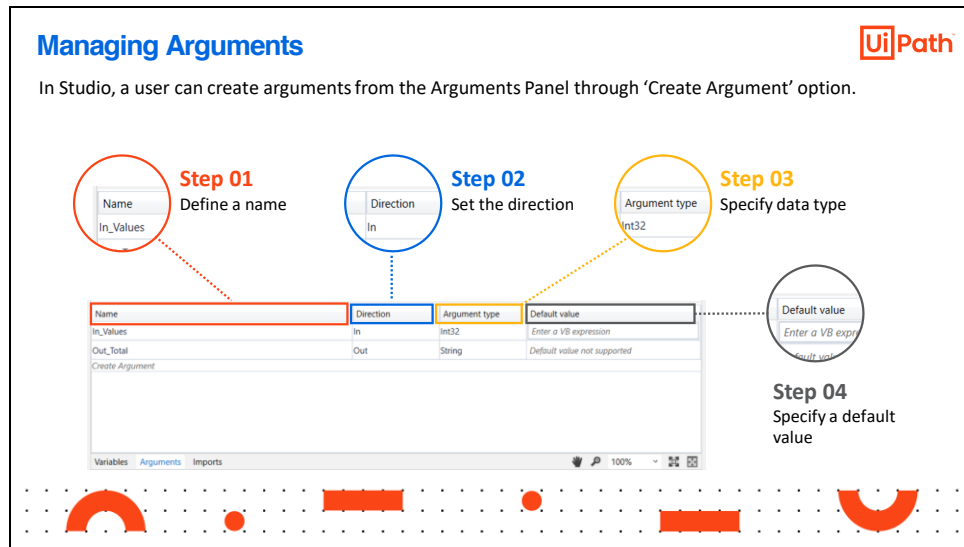


Studio has an **Arguments** Panel that enables the users to create arguments and modify them. Whenever a user changes the name of an argument in this panel, it automatically gets updated at all occurrences in the current file.

The panel is located at the bottom right side of the Activities and it is hidden by default. When clicked, the panel opens and displays the existing arguments in the project and allows the user to modify them.

The Arguments Panel consists of fields through which the users can define the different properties of an argument. It has the following fields:

- Name
- Direction
- Argument Type
- Default Value



In order to manage the arguments in UiPath, the users can need to perform several actions on them. These are:

- Creating Arguments
- Removing Arguments

1. Creating Arguments

There are 3 ways to create arguments in UiPath:

- **From the Arguments panel** – In Designer panel, click Arguments to display Arguments Panel. Click Create Argument & name it. (The argument is created, and its default type is String with 'In' direction.)
- **From the Designer panel** – Drag an activity from Activities panel to Designer panel and press Ctrl+M. Set Arg field is displayed. Name it and then check its properties in the Arguments panel. (The argument is created, and its type is according to the activity.)
- **From the Properties panel** – In the Properties panel of the activity, choose the field in which the argument is needed (i.e. Output) and press Ctrl+M. Name it. (The argument is created, and its type is according to the selected property. Check its properties in the Arguments panel.)

While creating an argument, its properties need to be initialized. The user can create an argument by following these steps:

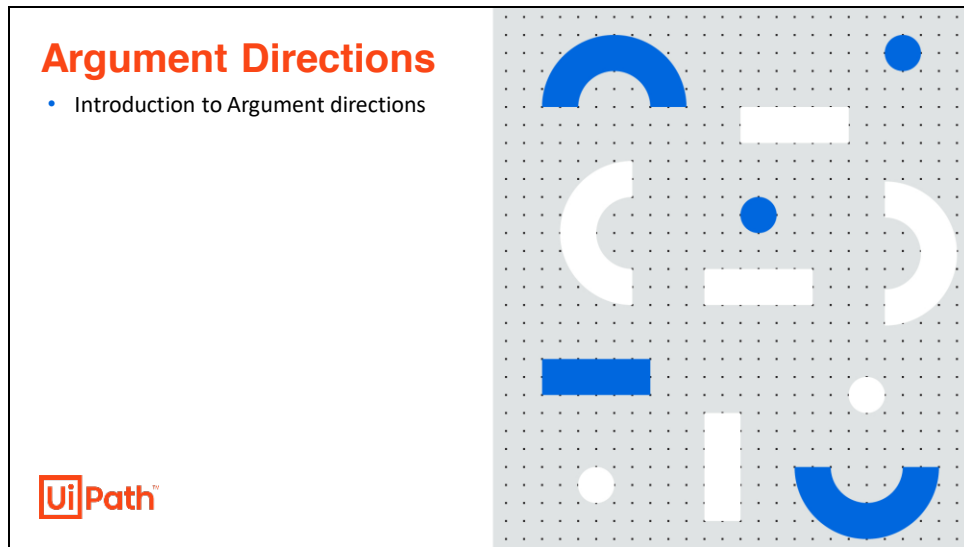
- Define a name for the argument.

- Set the direction of the argument (In, Out, In/Out).
- Choose the type of the argument.
- Now choose a default value.

2. Removing Arguments

A user can follow either of the following steps to remove an argument:

- Open the Arguments panel, right-click an argument and select the Delete option.
- Open the Arguments panel, select an argument and press the Delete key.



This section gives an overview of argument directions.

Introduction to Argument Directions

Specify the direction from/to which the data is passed.

01

In

- Argument can only be used within the given workflow

02


Out

- Argument can be used to pass data outside the given workflow

03

In/Out

- Argument can be used both within & outside the given workflow





Argument direction tells the applications where the information stored in them is supposed to go, i.e., it specifies the direction from/to which the data is passed.

The directions are:

- In – the argument can only be used within the given workflow.
- Out – the argument can be used to pass data outside of a given workflow.
- In/Out – the argument can be used both within and outside of a given workflow.


Argument names should be prefixed for easy identification of the argument direction, such as `in_FileName`, `out_TextResult`, `io_RetryNumber`.

Classroom Exercise




Demonstrate how to pass arguments between two workflows to display the sum of two numbers.

- One workflow will accept the numbers from the user and pass it to the other workflow.
- The other workflow will then add the values and return the value.



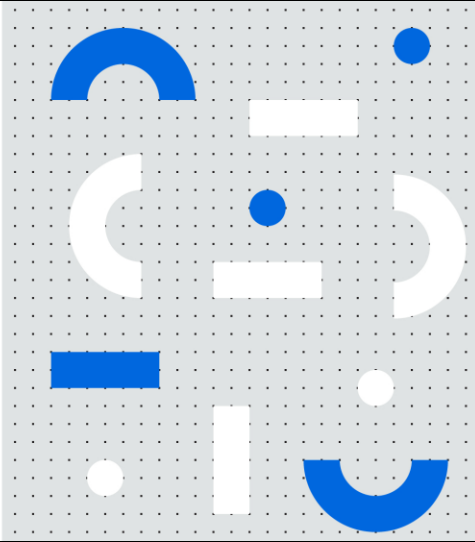
Demonstrate how to pass arguments between two workflows to display the sum of two numbers.

- Go to Arguments panel.
- Click Create Argument and enter an argument name called Input1.
- Select a Direction for the argument. You can choose between four available options:
 - o In takes value of an argument inside the workflow
 - o Out takes value outside the workflow.
 - o In/Out takes value inside or outside the workflow
- For this first argument, select Direction as In and Argument type as Int32. Int32 is for integers.
- Create second argument called Input2, select Direction as In, and Argument Type as Int32.
- Create third argument called Output, select Direction as Out, and Argument Type as Int32.
- Drag and drop Assign activity in the Designer panel.
- Enter Output in the first text area and Input1+Input2 in the second text area.
- Right click on the Sequence container and select Extract as Workflow from the context menu.
- Enter SumTwoNumbers as workflow name and click Create.
- Go to main file, and create three variables:
 - o An input variable A with Variable type as Int32 and Default value as 5
 - o An input variable B with Variable type as Int32 and Default value as 10
 - o An output variable Result with Variable type as Int32.


- Click Import Arguments within Invoke workflow, and pass the variables for each argument value:
 - o A for Input1
 - o B for Input2
 - o Result for Output
- Drag and drop a Message box activity below Invoke workflow and enter Result in text area.
- Run the file to get the output as 15.

Arguments vs. Variables


- Comparison



This section gives a comparison between arguments and variables.

Arguments vs. Variables 

Arguments	Variables
Argument stores data and passes it between workflows	Variable stores data and passes it between activities
It can be used across multiple workflows (direction to be defined)	It is limited to the workflow in which it was defined
It is created & modified through Arguments Panel	It is created & modified through Variables Panel
It is defined by properties: Name, Direction, Type, Default Value	It is defined by properties: Name, Type, Scope, Default Value
If a user does not assign a value to an argument, it does not have any default value	If a user does not assign a value to a variable, it is automatically assigned a default value





Arguments and Variables are quite similar to each other. They both are used to store information of a particular data type and pass it.

But some of the difference between them are:

- Argument stores data and passes it between workflows. Variable also stores data but passes it between activities.
- An argument can be used across multiple workflows (direction to be defined) but a variable is limited to the workflow in which it was defined.
- An argument is created & modified through Arguments Panel whereas a variable is created & modified through Variables Panel.
- An argument is defined by properties: Name, Direction, Type, Default Value whereas a variable is defined by Name, Type, Scope, Default Value.
- If a user does not assign a value to an argument, it does not have any default value whereas if a user does not assign a value to a variable, it is automatically assigned a default value.

Practice Exercise





Build a workflow that swaps two numbers using a third variable.

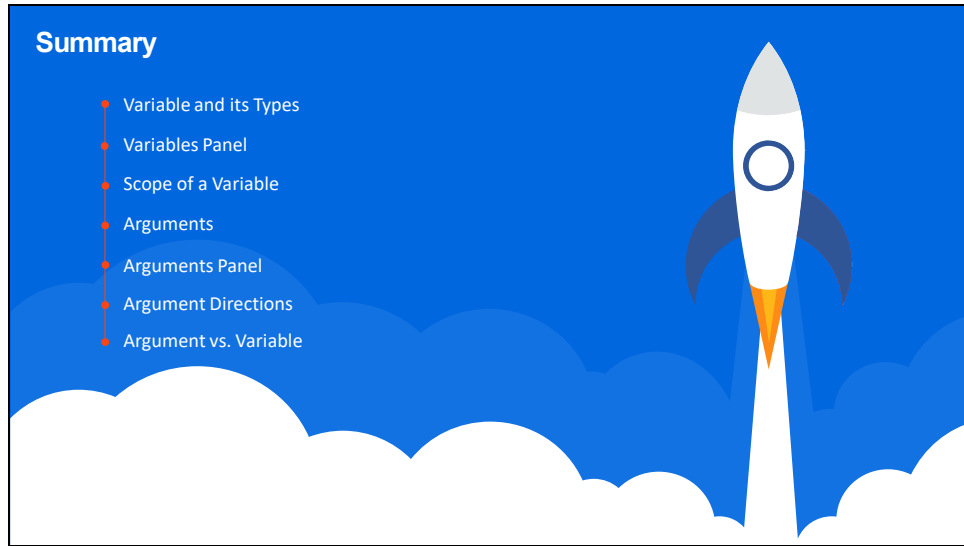
- Ask the user to input two numeric values and store in two variables.
- Swap the values of both the variables using a third variable.
- Display initial and swapped values of both the variables in the Output panel.

Build a workflow that swaps two numbers using a third variable.

- Ask user to input two numeric values and store in two variables.
- Swap values of both the variables with each other using a third variable.
- Display initial and swapped values of both the variables in the Output panel.

Algorithm

- START
- Use Input Method activity to receive two numeric values from the user
- Store the received values in two integer variables called **First_Input_Value**, and **Second_Input_Value**
- Declare a third integer variable called **Swapping_Support_Variable**
- Use Assign activity to assign value of **First_Input_Value** to **Swapping_Support_Variable**
- Use second Assign activity to assign value of **First_Input_Value** to **Second_Input_Value**
- Use third Assign activity to assign value of **Second_Input_Value** to **Swapping_Support_Variable**
- Use Write Line activity to display initial and final values of **First_Input_Value** and **Second_Input_Value** in the Output panel
- STOP



To summarize, this lesson explained:

- Variable and its Types
- Variables Panel
- Scope of a Variable
- Arguments
- Arguments Panel
- Argument Directions
- Argument vs. Variable