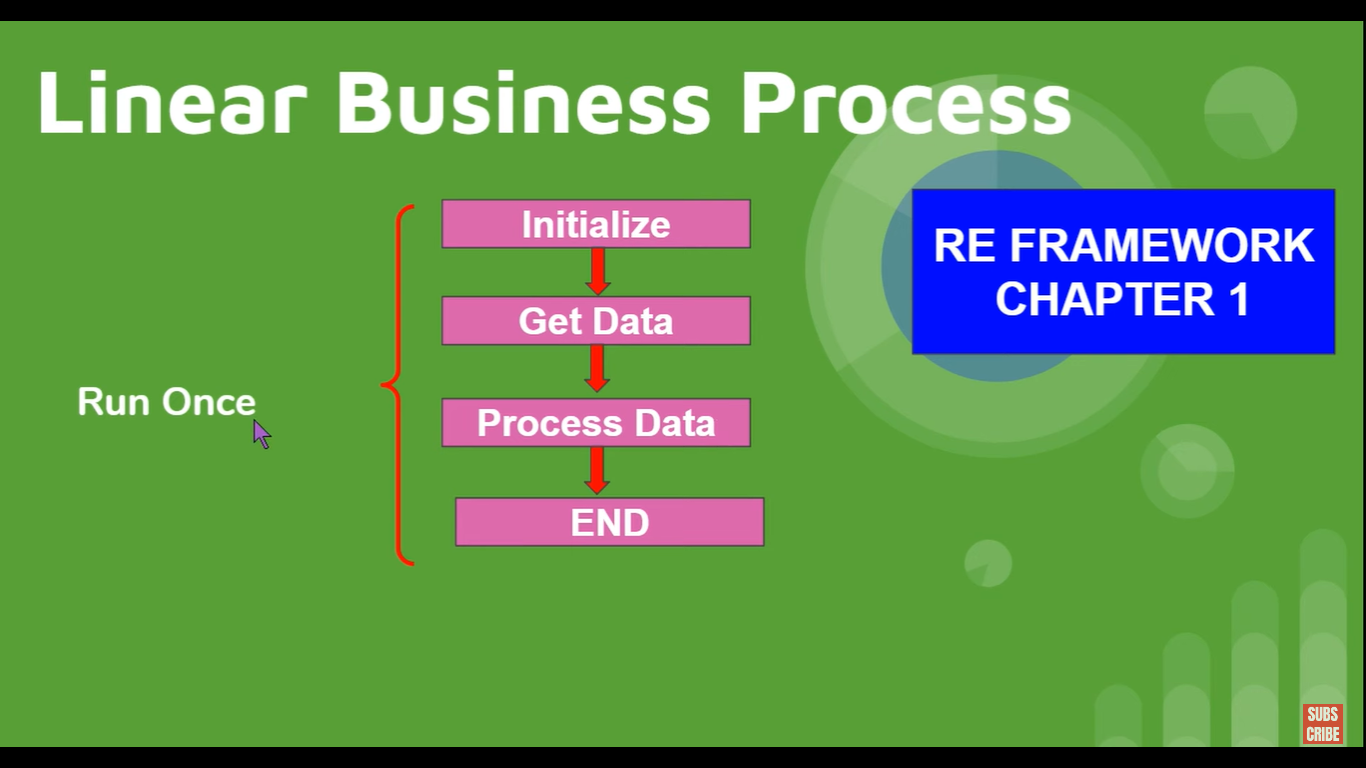
RE Framework(Robotic Enterprice Framework):

UiPath ReFramework provides a framework for building and automating business processes across different categories. Here are the three categories of business processes in ReFramework:

Sequential Processes: (linear process)

Sequential processes are linear and involve a series of steps that need to be executed in a specific order, such as employee onboarding, customer account setup, and purchase order processing. ReFramework provides a sequential workflow for automating these processes, with built-in support for error handling and retry mechanisms.

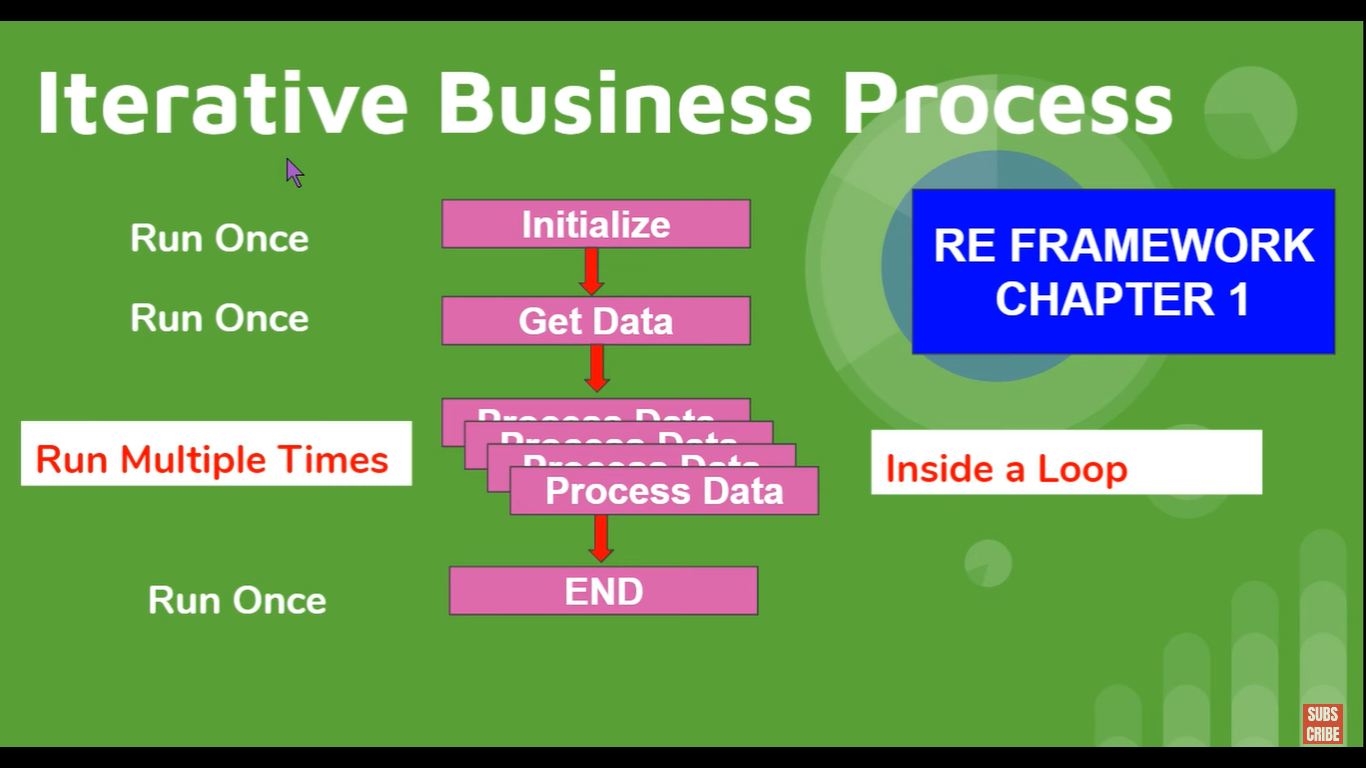
* Process run once



* Not suitable for repeatative processes

State Machine Processes: (Iterative process)

State machine processes are complex processes that involve multiple states and conditions, such as customer service inquiries, claims processing, and loan approval. ReFramework provides a state machine workflow for automating these processes, with support for conditional branching and decision-making based on the state of the process.



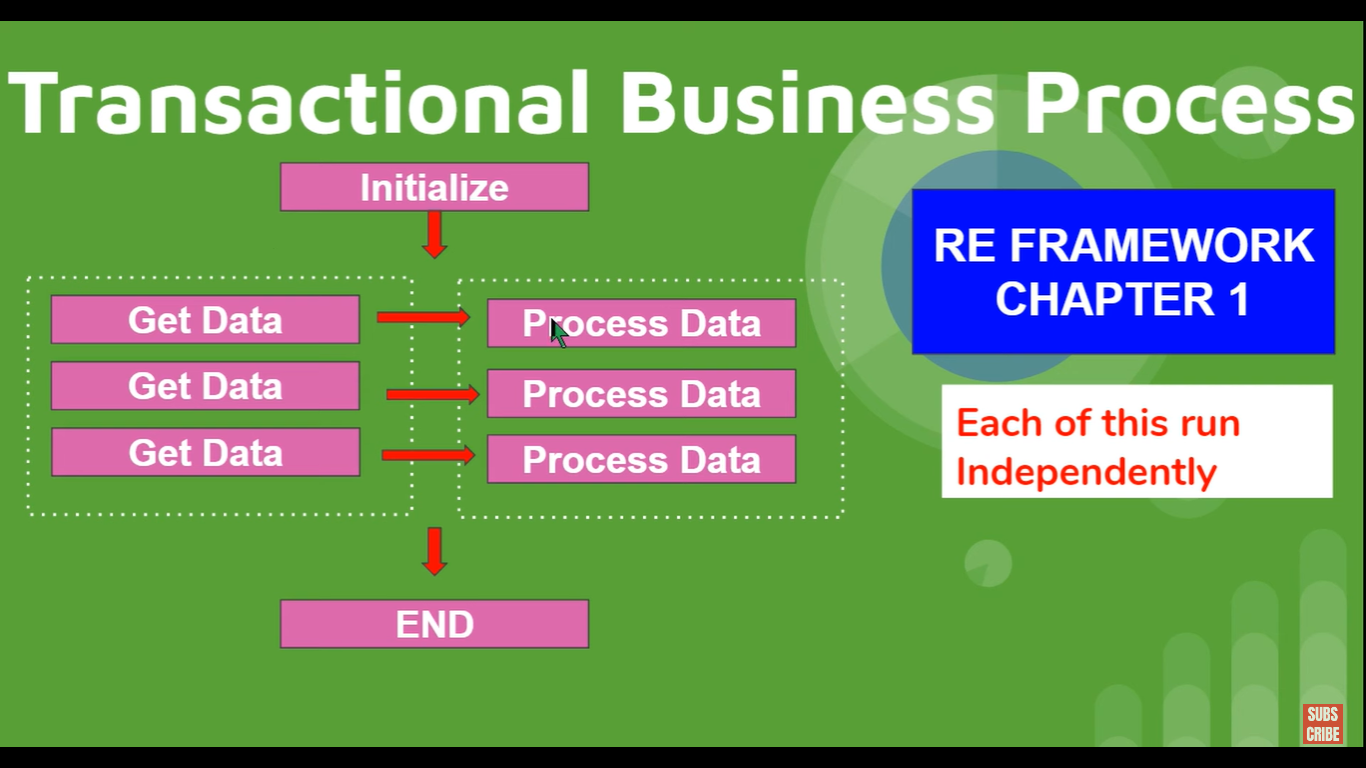
* Get data once and process it mutiple times

Drawback:

* If one email is failed entire process will be failed

Transactional Processes:

Transactional processes are repetitive and standardized processes that involve a large volume of transactions, such as invoice processing, order processing, and data entry. ReFramework provides a standardized workflow for automating transactional processes, including error handling, logging, and recovery mechanisms.



* Data is indepenent
* Like get data and process-it( one loop combined)

By providing a standardized framework for automating business processes across these categories, ReFramework helps organizations to streamline their operations, improve efficiency, and reduce costs.

RE Framework – stages and transitions:

Stages:

4 Stages

* Init
* Get transaction data
* Process transaction
* End process

Transitions:

7 transitions – move from one state to another

Stages in detail:

Init:

This transition is used to move the bot from the starting state to the initialization state. The initialization state is where the bot sets up any necessary variables, opens applications or webpages, and performs any other tasks required to prepare for the main process.

Get Transaction Data:

This transition is used to move the bot from the initialization state to the transaction data retrieval state. Here, the bot retrieves the data it needs to perform the main process, such as a list of items to process, from a data source such as a spreadsheet or database.

Process:

This transition is used to move the bot from the transaction data retrieval state to the main processing state. Here, the bot performs the main process, such as filling out a form or processing an invoice, using the data it retrieved in the previous state.

End Process:

This transition is used to move the bot from the main processing state to the end state. Here, the bot performs any necessary cleanup tasks, such as closing applications or saving data, and then ends the process.

Exception Handling:

This transition is used when an error occurs during the bot's execution. It allows the bot to handle the error and recover, either by retrying the task or taking some other action to resolve the error.

Init:

* Config Variable – Dictionary type(no default value at start )
* Init all settings InvokeWork Flow:
* Variable – Out\_config(Dictionary) – no value
* Reading the config file and storing the values in a Dictionary
* Output of Init:

🡪We are getting Config dictionary variable which has all the Config excel file data

🡪 Assined archestrator Queue name

🡪 Kill applications

🡪.Open applications

Get Transaction Data:

* Get transition data
* Orchestrator Queue

Steps:

🡪 Create a queue in Orchestrator

Ex: upload a file in to the server (PDF Invoices) – once uploaded you want to download and process it

* To upload data use InIt – first state.
* To download data – Get transtion state.—2nd state
* The value of get transaction item is stored in new arument called TransactionItem

Process State:

* Key Variable or arguments