REXX Conversational and Variable Management Instructions

Objectives

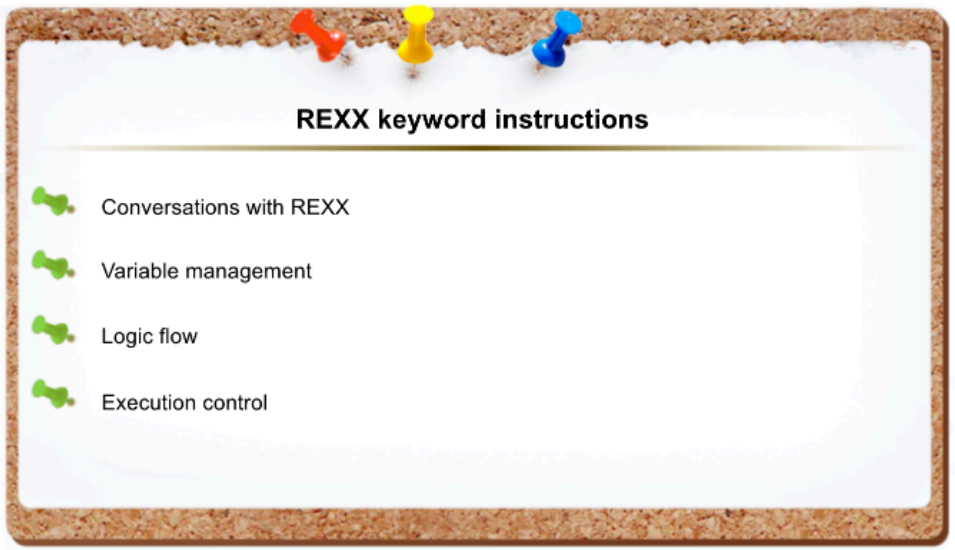
REXX Conversational and Variable Management Instructions

In this module, you will explore the coding and use of conversational and variable management keywords in REXX.

You will look at the external data queue and its instructions, and see how variable templates can be used to assign variable values.

After completing this module, you will be able to:

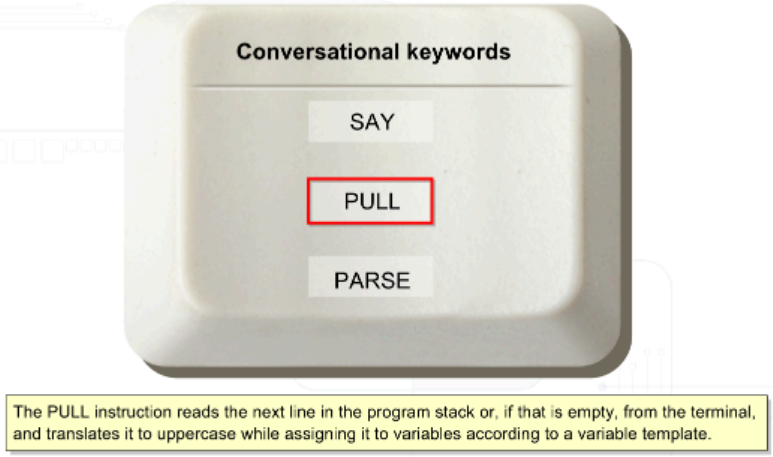
- Identify Conversational and Variable Management Keyword Instructions
- Identify How to Use and Manage the External Data Queue
- Define Simple and Complex Variable Templates to Assign Variable Values



REXX provides a range of keyword instructions to control execution flow within an exec and provide various services to programmer and user. Some instructions also provide other instructions that are nested within them.

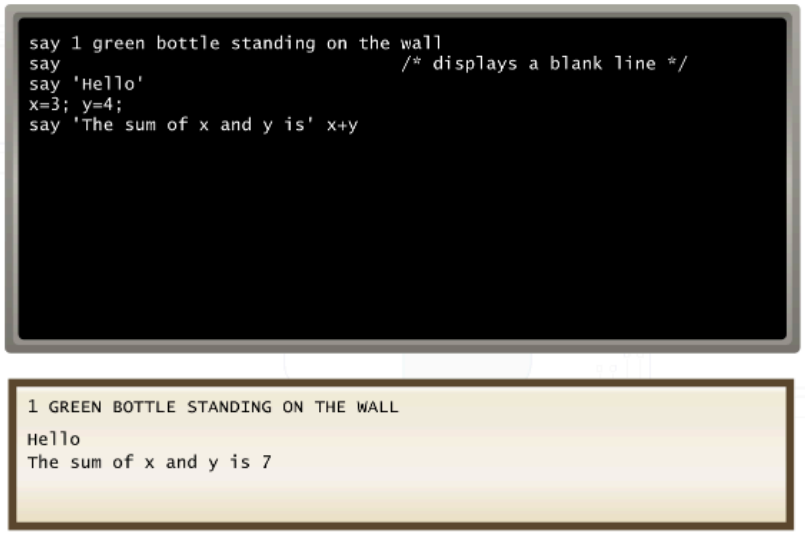
A REXX keyword instruction must be the first token in a clause and is not case dependent. For example, an instruction entered as "do" is the same as "DO".

REXX instructions can be used to manage variables, provide logic functions, control execution and processing of REXX and other programs, and manage conversations in REXX.



Conversational instructions help you manage conversations in REXX. Listed above are the three main REXX conversational instructions.

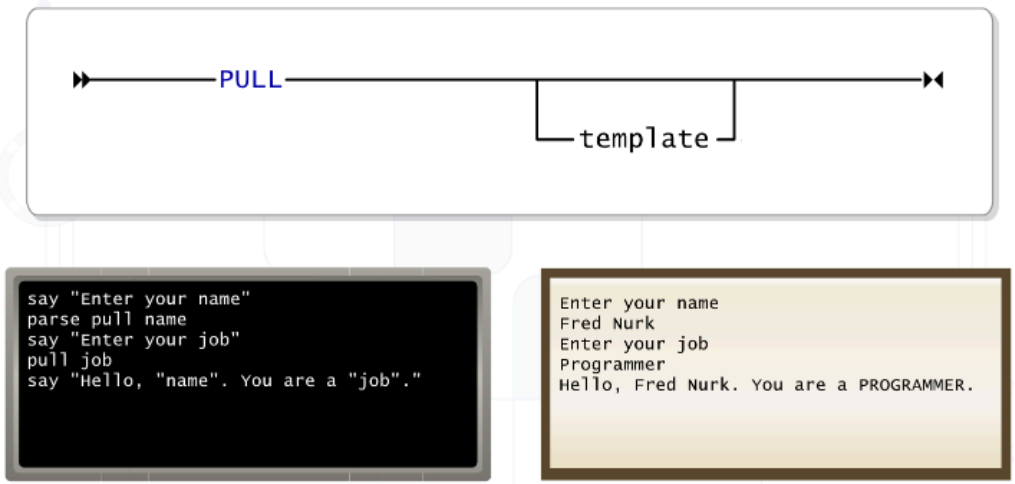
Mouse-over each instruction for a description.



The SAY instruction is used to display output on the terminal.

If the SAY instruction is coded without any expression, REXX will output a blank line; otherwise, any valid REXX expression such as a variable, constant, arithmetic expression, character string, or any combination of these can be coded.

Click **Play** to see some examples of the SAY keyword instruction.



The PULL instruction is used to receive input data from the external data queue or, if that is empty, from the terminal. It is mostly used in combination with the SAY instruction. SAY is used to ask a question and PULL is used to get the answer.

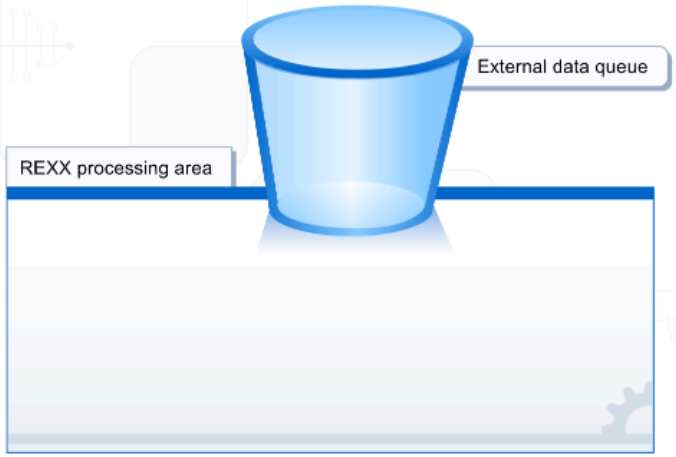
The PULL instruction is an abbreviated form of the PARSE UPPER PULL instruction, and it converts all character input to uppercase. The PARSE PULL instruction must be used to receive input data without uppercase translation. The template is a model of how the data received should be broken down into variables. If only one variable name is used, it is set to the entire string entered.

Click **Play** to see how the PULL and PARSE instructions are used.

Variable management instructions	
ARG	Retrieves input argument strings and translates them to uppercase
DROP	Resets or drops one or more variables
PARSE	Assigns data from various sources to variables
PROCEDURE	Provides for local variables within an internal routine
PULL	Reads the next line in the program stack and translates it to uppercase
PUSH	Places the results of a given expression at the beginning of the program stack; last in, first out
QUEUE	Places the results of a given expression at the end of the program stack; first in, first out
UPPER	Translates one or more variables to uppercase

Variable management instructions are used to control, set, and manage variables and data during the execution of a REXX program.

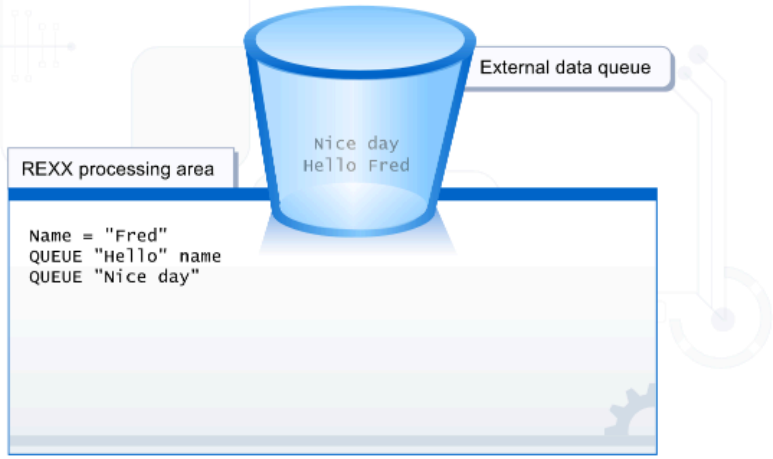
Listed above are the instructions that are used to manage variables.



The external data queue, which is also referred to as "the data stack" or just "the stack", is used to store raw data in the form of records without using variables.

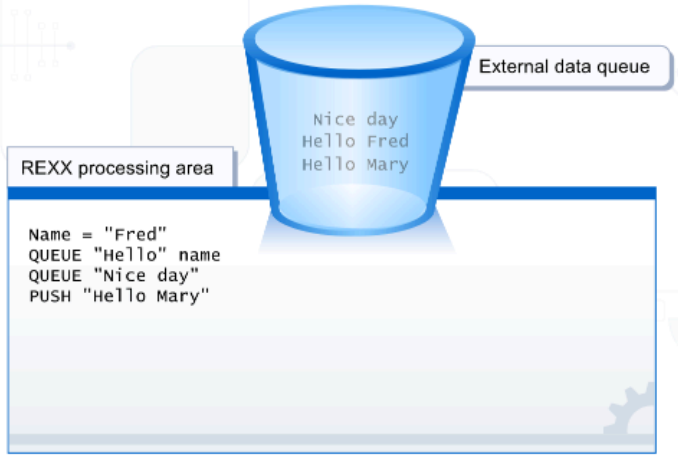
Imagine a bucket with a hole in the bottom of it, which is placed on top of the area in which a REXX procedure is processing. All the data records to be stored are placed in the bucket and then pulled, one at a time, through the bottom as required.

The external data queue is managed by the QUEUE, PUSH, PULL, or PARSE PULL instructions.



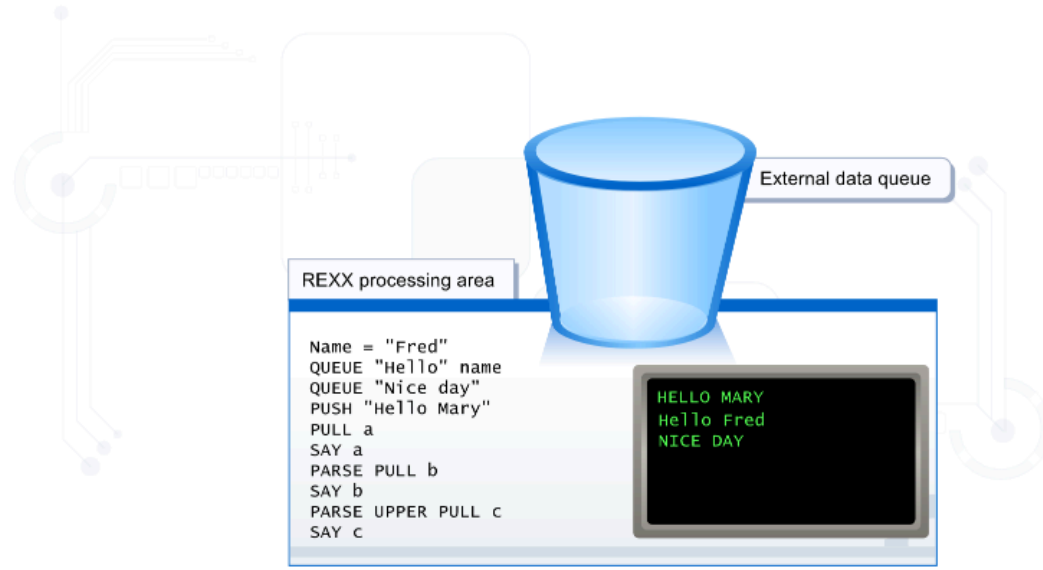
The QUEUE instruction evaluates an entered expression and places the resulting data as a record into the data stack in first in, first out order (FIFO), or puts it into the top of the bucket as shown in the previous diagram.

Click **Play** to see how the QUEUE instruction works.



PUSH evaluates an entered expression and places the resulting data as a single record into the data stack according to last in, first out (LIFO) processing, or is "pushed" up into the bucket, as shown in the previous diagram.

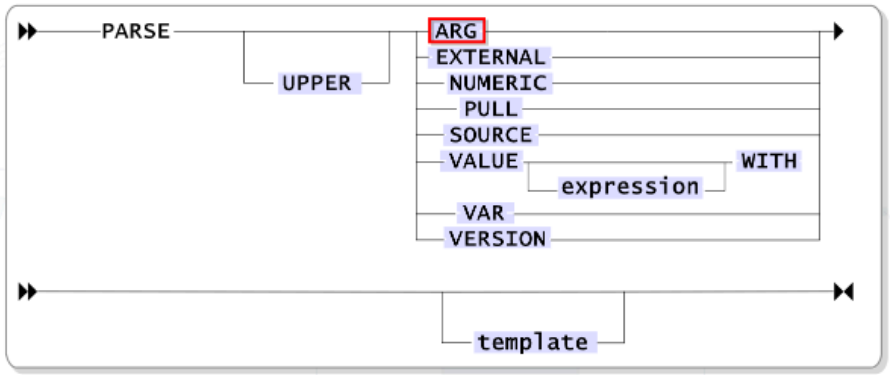
Click **Play** to see how the PUSH instruction works.



The PULL instruction is used to extract data from the external data queue one line at a time and break it down according to a variable template. Using the same bucket analogy, the data would be pulled from the stack though the bottom of the bucket.

PARSE PULL and PARSE UPPER PULL can also be used. If the PULL instruction is used and there are no records in the stack, the REXX interpreter waits for input from the terminal. Records left in the stack after the REXX program has completed will be processed by the application running the program, but results could be unpredictable.

Click **Play** to see how the PULL instruction works.



PARSE ARG accepts the source as the parameters specified when the program is executed.
For example, if the REXX named MYREXX1 was executed by the command:
MYREXX1 December 2008

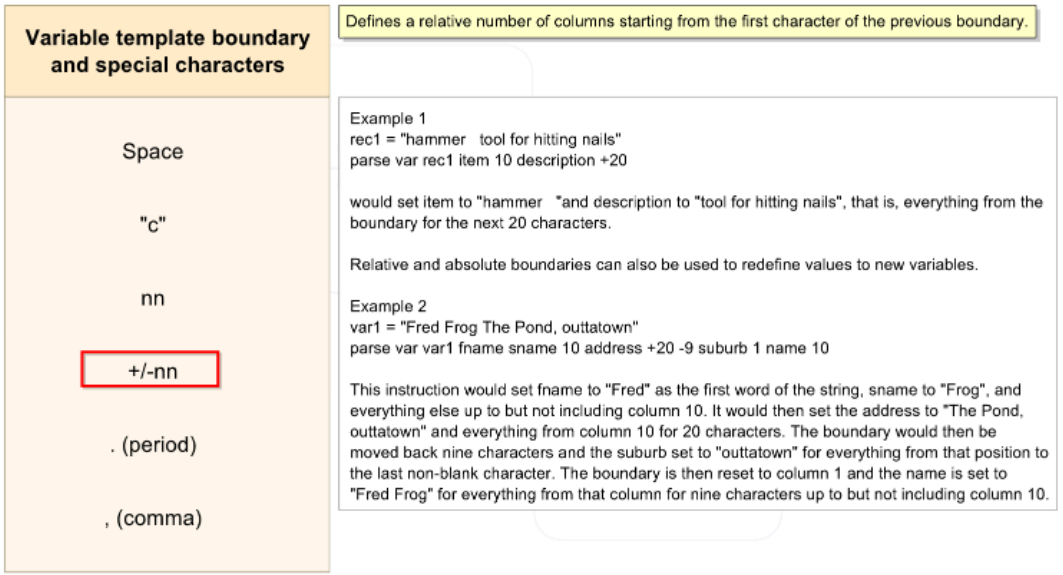
then the following instruction:
Parse arg month year

would result in the variable month being set to "December" and the variable year being set to "2008".

To "parse" in programming is to analyze or separate a value or string of characters into more easily processed components.

The PARSE instruction is used to assign values from various sources to variables. This is done according to the rules specified in the PARSE instruction variable template.

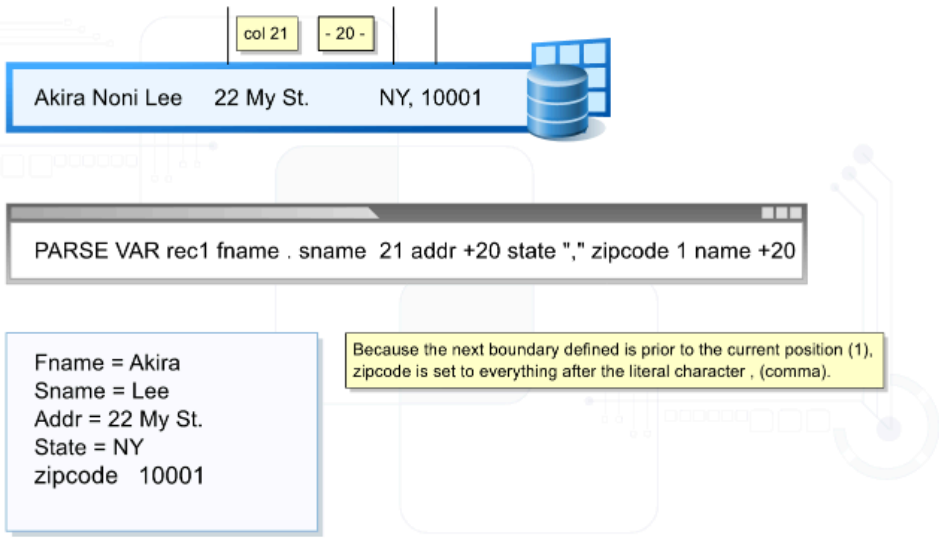
Mouse-over the syntax to see definitions of the parameters and source areas.



PARSE is a powerful keyword instruction that breaks down character strings and sets variables. When coding the PARSE instruction, templates can be used to describe how the data is to be divided. A template is a model of how a character string should be divided.

Spaces, literals, and relative and absolute column values can be used to define boundaries, and variables are set to the values within those boundaries.

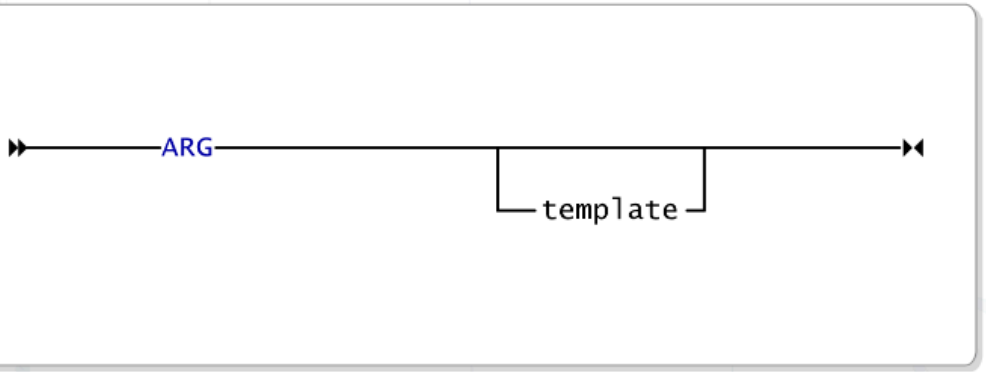
Mouse-over the boundary values for descriptions and examples.



The use of boundary characters and multiple variables with the PARSE instruction enables you to use one statement to perform what would normally take many assignment statements.

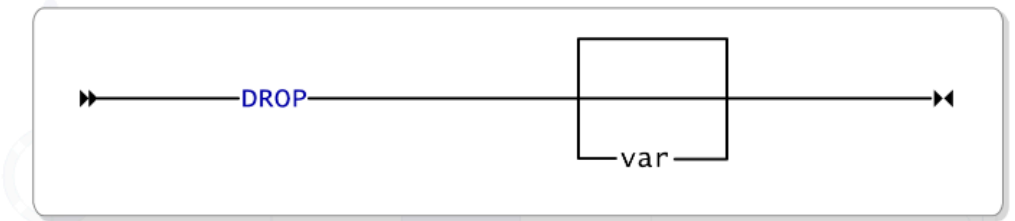
In this example, a variable called rec1 contains a typical data record.

Click **Play** to see this example in use and the values that it would set.



The ARG instruction is an abbreviated version of PARSE UPPER ARG, which can define multiple strings and templates if the REXX has been called from another REXX.

Use PARSE ARG to receive parameters or arguments in the same case in which they were passed.



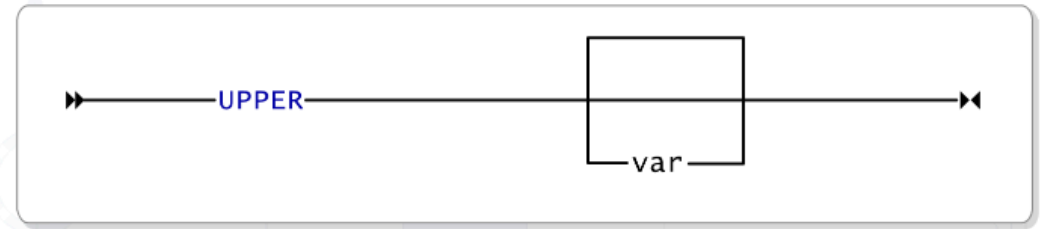

```
Name = "Fred Nurk"
Job = "Programmer"
say "Hello, "name". You are a "job"."
DROP name job
say "Hello, "name". You are a "job"."
```

```
Hello, Fred Nurk. You are a Programmer.
Hello, NAME. You are a JOB.
```

Any word that is not recognized as a keyword instruction, function, or literal by the REXX interpreter is considered to be a variable. Unlike most other programming languages, the initial value of any variable is the variable name in uppercase.

Variables can be assigned a value though several facilities and they retain that value until the REXX completes processing. Use the DROP instruction if you want a variable to be reset or unassigned.

Click **Play** to see an example.




```
Name = "Fred Nurk"
Job = "Programmer"
say "Hello, "name". You are a "job"."
UPPER name job
say "Hello, "name". You are a "job"."
```

```
Hello, Fred Nurk. You are a Programmer.
Hello, FRED NURK. You are a PROGRAMMER.
```

The UPPER instruction converts to uppercase the value of any alpha characters assigned to a variable name. There is no equivalent for lowercase.

Click Play to see an example.