OFFICIAL DOCUMENTATION FOR mushroom

Spread the spores!

DATA TYPES

INTEGERS:

The INTEGER data type can only store integers.

e.g. 1, 5, -3

FLOATS:

The FLOAT data type can store decimal numbers.

e.g. 1.2, 5.6, -7.2

STRINGS:

Strings can store arrays of characters. Strings have to be used between quotation marks.

e.g. “Hello World”

LISTS:

Lists can store more than one value. These values can be from different data types. The elements of a list can be accessed by using indexes. The first element’s index is 0 and the rest are consecutive. A list element can be accessed by using listName/elementIndex

Elements of a list can be declared by using

VAR listName= [listElement,listElement]

NOTE: All data types are declared and made operations on by using the VAR keyword.

OPERATORS

There are 5 operators that can be used on numbers (integers and floats). These are the ADDITION operator (+), the SUBTRACTION operator (-), the MULTIPLICATION operator (*), the DIVISION operator (/) and the POWER operator (^).

KEYWORDS

VAR:

Used to declare and make operations on a variable. The type of the variable is declared automatically.

SYNTAX:

VAR (variable name) = (variable value)

VAR (variable name) (operation) (value)

AND/OR:

Logical operators used to return TRUE or FALSE based on the statements given.

AND= Returns TRUE if both statements are TRUE

OR= Returns TRUE if one of the statements are TRUE

SYNTAX:

(statement) Logical Operator (statement)

NOT:

NOT= Flips the value of a statement

SYNTAX:

NOT (statement)

IF STATEMENTS:

If statements are used to do an action if the condition is TRUE. ELIF can be used to create other conditions to be checked if the initial one is false. ELSE can be used to create actions to be done if none of the conditions are true. THEN is used after the condition statement. END is used to specify the end of an IF STATEMENT. ELIF and ELSE are not necessary for a IF STATEMENT to function.

SYNTAX:

IF (condition) THEN (action)

ELIF (condition) THEN (action)

ELSE (action)

END

FOR LOOPS:

FOR LOOPS can be used to repeat an action over and over. A FOR LOOP can be constructed by using the FOR, TO, THEN ,STEP and END keywords. TO can be used to determine how many times the action will be executed by setting up a range of indices. Each action will take an index on the FOR LOOP. Usually "i" is used to show indices however anything can be used. THEN is used after the range of indices. STEP can be used to determine the value of increment. If STEP is not used this value will be 1. END is used in the end of the loop.

SYNTAX:

FOR i=(first value) TO (last value) THEN

(action)

END

FOR i=(first value) TO (last value) STEP (increment value)THEN

(action)

END

NOTE: i will take the values from (first value) to (last value) for every iteration of the loop. (last value) is not included.

WHILE LOOPS:

A WHILE LOOP is another kind of loop used similarly to the FOR LOOP. However WHILE LOOPS take conditions instead of ranges. A WHILE LOOP uses the WHILE, THEN and END keywords. A WHILE LOOP will check the condition at the start of the loop and for every iteration. The WHILE keyword is used at the beginning of the loop. THEN is used after the condition. END is used at the end of the loop.

SYNTAX:

WHILE (condition) THEN

(action)

END

CONTINUE/BREAK:

CONTINUE and BREAK can be used in FOR and WHILE loops. CONTINUE can be used to stop the current iteration and start the next one. BREAK can be used to stop the current iteration and end the loop.

FUNCTIONS:

A FUNCTION is used to create a code package. This FUNCTION can be called later to apply the code piece in it. FUNCTIONS will return the result of the piece of code. FUNCTIONS can have parameter or parameters when being called. These parameters can be stored in variables in the scope of the FUNCTION. These variables will not have an effect outside the FUNCTION. FUN, RETURN and END keywords are used in a FUNCTION. FUN s used at the beginning of a FUNCTION. RETURN is used to return a value. END is used in the END of a FUNCTION. Additionally "->" can be used to create one-lined functions. "->" is used as an alternative to RETURN. If "->" is used there is no need for RETURN or END.

SYNTAX:

FUN (function name) ((variable name to store parameter))

(code piece)

RETURN (value to return)

END

NOTE: More than one parameter can be used in a function. These parameters can be taken by using commas:

FUN (function name) ((variable name to store parameter),(second variable name to store parameter))

ONE-LINED FUNCTION SYNTAX:

FUN (function name) ((variable name to store parameter)) -> (value to return)

NOTE: Functions can be called using:

(function name)(parameters)

NULL/FALSE/TRUE:

NULL can be used to represent having no values or 0. FALSE and TRUE can be used to represent a conditions state. FALSE represents 0 and TRUE represent 1.

MATH_PI:

MATH_PI returns the first 15 digits of PI.

NOTE: MATH_PI can be called from the shell like a function. However it doesn't require parameters.

FUNCTIONS

PRINT/PRINT_RET:

PRINT can be used to print data to the screen. This data can be integers, floats, strings or lists. PRINT_RET returns the value that would be printed but doesn't print it on the screen. The values to be printed on the screen are given as parameters.

INPUT/INPUT_INT:

INPUT can be used to get a string input from the user. INPUT_INT is used to get an integer input. If the input isn't an integer, it will give an error and ask for another input. Both functions don't take any parameters.

CLEAR/CLS:

Both CLEAR and CLS clear the screen. Both functions don't take any parameters.

IS_NUM/IS_STR/IS_LIST/IS_FUN:

IS_NUM will return 1 if the parameter is an integer and 0 if it isn't. IS_STR will return 1 if the parameter is a string and 0 if it isn't.

IS_LIST will return 1 if the parameter is a list and 0 if it isn't. IS_FUN will return 1 if the parameter is a function and 0 if it isn't.

APPEND:

APPEND is used to append a value to the end of a list. It can only be used to append one value. It takes a list as the first and the value to be appended as the second parameter.

POP:

POP is used to remove an element of a list by it's index. It takes a list as the first and an integer(the index) as the second parameter.

EXTEND:

EXTEND is used to combine two lists. The second list is added to the first list's end. It takes two lists as parameters.

SHELL-ONLY COMMANDS

SETUP:

SETUP is used to change the default keywords and built-in functions by taking the file paths to keywords.txt and functions.txt

ADD TO PATH:

ADD TO PATH is used to add mushroom to the system Path variable. By doing this you can access mushroom from any directory by just running "mushroom" in a terminal.

TRANSLATE:

TRANSLATE is used to translate code from your custom language to any other custom language by taking two pairs of keywords.txt and functions.txt files. This command can also be used to translate to the default language.

VSCODE:

VSCODE is used to edit the Visual Studio Code extension for mushroom to suit your custom language. It takes the path to keywords.txt, functions.txt and mush.tmLanguage.json located in the VSCode files (the command will guide you to the file).

RUN:

RUN takes the path to a script you want to run and executes it.

ADDITIONAL NOTES

- mushroom can detect new lines without semicolons (;), however you can use semicolons to fit multiple lines of code to just one line.
- mushroom uses .mush as a file type however it can still run .txt files