Strings

Strings of characters are denoted by quotation marks on both ends:

```
"This is a string."
「This is also a string.」
```

Lakshayati organizes quotes by family. For English speakers, the single quote ' ‘ and ’ are all treated the same in Lakshayati, but they are in a distinct family from the double quote " “ and ”. Since they are treated the same, ” can begin a string and “ can end it, but this is not true for all quotation marks. For example, regarding CJK scripts, 「 must mark the start of a string and 」 must conclude it. A list of quote families based on whether order matters can be seen here:

Order does not matter	Order matters (former must begin a string, latter must end it)
" “ ” „	「 」
' ‘ ’ ‚	『 』
« »	《 》
‹ ›	〈 〉

** Note that ‚ is a quotation mark despite looking almost identical to a comma , Commas cannot denote strings in Lakshayati.*

But what should a programmer do if they want quotation marks within the string? There are several methods for doing this, and they bear much in common with the methods used in other programming languages. Read whichever section applies to you based on the category of quotation mark that you will rely on.

Quotes in strings if order does not matter…

There are two possible methods for including quotation marks within strings:

a) Any odd number of quotations marks may start a string, and then the same number of quotes in the same family may conclude a string.

```
    ex:  '''''The quotes ' and ''' appear in the string.'''''
result:  The quotes ' and ''' appear in the string.
```

b) Type a slash (/ or ／) before a quotation mark to indicate it is within the string, not concluding the string. The Lakshayati interpreter will know to remove the slash.

```
    ex:  "/"Greetings, friend!/" Celia exclaimed."
result:  "Greetings, friend!" Celia exclaimed.

    ex:  """／"Praise be," said the preacher."""
result:  "Praise be," said the preacher.
```

Notice how the first " in the last string requires a slash while the second does not. Without the slash, Lakshayati would detect an even number of quotes """", which would mark an empty string. Additionally, if one wants to clarify a slash is not to be removed, they should type two slashes.

```
    ex:  "One slash will appear at the end of this string //"
result:  One slash will appear at the end of this string /

    ex: "///" denotes a slash before a quotation mark."
result: /" denotes a slash before a quotation mark.
```

Quotes in strings if order does matter…

As with the previous section, there are two similar methods for including quotation marks within strings:

a) Any set of initial quotation marks may start a string, and then the same number of concluding quotation marks must end it.

ex: `「「The quotes 」and「「 can appear in the string.」」`
result: `The quotes 」and「「 can appear in the string.`

b) Type a slash (/ or ／) before a quotation mark to indicate it is within the string, rather than beginning or concluding the string. The Lakshayati interpreter will know to remove the slash.

ex: `「／「Greetings, friend!/」 Celia exclaimed.」`
result: `「Greetings, friend!」 Celia exclaimed.`

ex: `「「「／「Praise be,」 said the preacher.」」」`
result: `「Praise be,」 said the preacher.`

Note that the 「 in the last string requires a slash, since Lakshayati would otherwise detect 「「「「 as the onset of a string. Furthermore, if someone wants to clarify a slash is not to be removed, they should type two slashes.

ex: `「One slash will appear at the end of this string ／／」`
result: `One slash will appear at the end of this string ／`

ex: `「///「 shows a slash coming before a quotation mark.」`
result: `/「 shows a slash coming before a quotation mark.`

Lines of Code

In Lakshayati, a line of code terminates when there is a newline character *outside of a string*:

```
Suppose this is valid code "The
line will not be terminated
until all strings conclude" "and
there is a newline character at the end."
This is a new line of Lakshayati
```

An exception to this rule is, if all strings are concluded, Lakshayati will terminate the *last* line of a program even if a newline character is not technically found at the end. If the last string in a program is not concluded, you will receive an error message.

Additionally, spaces at the beginning and end of a line are ignored by Lakshayati, so feel free to indent lines as you please.

TL;DR: Lines of code in Lakshayati are structured much like Ruby’s.

Variables

Variables in Lakshayati can consist of any set of Unicode characters, except whitespace characters and quotation marks in the 8 families. Variables are assigned to a string value by typing the name of the variable then the string on a line:

ex: `first_var "Hello"`

Variables can have string values as well:

ex: `second_var my_var`

`second_var` currently has the same value as `first_var`, which is "Hello"

Multiple string values can be placed on a line if they are separated with spaces, and they will be joined together:

ex: `final_var my_var ", " "world!"`

`final_var` is given the value of `final_var` + `", "` + `"world!"` which is, `"Hello, world!"`

Finally, string values can be displayed to the screen by them to assigning the variable / or ／ (in Lakshayati, these characters are treated as the same).

ex: `/ final_var`

This displays `"Hello, world!"` to the screen. Since / is a variable, the same rules for all variables still apply.

Executing Strings

If the first (non-whitespace) character on a line is a quote, then all strings on the line will be joined together, and then the value of that string will be executed as Lakshayati code.

ex:
```
'my_var ' "'String within a string.'
 / my_var"
```

First, the two strings are joined together, yielding the value:

```
"my_var 'String within a string.'
 / my_var"
```

Since the line begins with a quote, the string above is treated as Lakshayati code, and the output becomes:

```
String within a string.
```

If variable is on a line by itself, its string value is also executed:

ex:
```
my_var "/ 'String within a string'"
my_var
```

`my_var` is on a line by itself, and the resulting output is the same as before.

Finally, if programmer wants to include a variable’s string value at the beginning of a line with other string values and wants to evaluate their combined string value, they can simply begin the line with an empty string:

ex:
```
first_var "/ '"
second_var "String within a string'"
"" first_var second_var
```

The empty string has no value, so `first_var` and `second_var` are joined together and executed. The output, again, becomes "String within a string"

Recursion

To simulate while-loops as found in other programming languages, Lakshayati uses recursion. This means an executed string can contain and execute itself multiple times. Take this program for example:

ex:
```
recursive_function "
        do_something
        recursive_function
 "
recursive_function
```

The action, in this case, will be performed forever. In practice, it is best to call the recursive function on the last line available (after `do_something`), since any following lines of code can never be read and only take up storage space.

Commentary

This is the last and simplest section of Lakshayati's documentation. If you would like to create comments within the program, the simplest method is to make your comment one or more long, empty variable(s):

```
This_is_one_way_to_write_comments._So_long_as_nothing_is
assigned_a_value,_there_will_be_no_effect_on_the_program
```

And that is everything you need to program in Lakshayati!