# Scripting Tutorial - Lesson 16: Tips and Tricks I

Download supporting files for this tutorial

Download this page in PDF format

Texas Instruments TI-Nspire Scripting Support Page

## Afterthoughts

So, where to from here?

Obviously, the sky is the limit. Depends on your own imagination and, perhaps, occasional need for sleep?

There is much more to be learned, and hopefully the grounding you have received here may equip you to search for answers in more formal places, mixing it with "real" programmers, perhaps? You certainly should visit (and maybe even contribute to) the Lua Nspired Wiki, maintained by all sorts of interesting and clever people.

You really *must* browse the excellent tutorials at Nspired Lua, and you will find some lovely examples at good old TI-Calc.org Lua archives.

But still a few things that I would like to cover here that didn't come up or were not adequately covered in lessons 1 - 15, so hang in there just a little longer.

*The first three notes come from my eminent guru of Lua, John Powers from TI, who has been largely responsible for developing Lua for our platform.*

### Find

If you want to know if a string contains a substring, and where the substring is, then use *find*. It returns two numbers, start and end, indices of the beginning and ending of the first match. It returns nil if the substring is not contained in the string. The find method has a versatile pattern matching capability that you may want to check out. For instance

**str:find("%b()")** returns the beginning and ending indices of the first set of balanced parentheses it finds.

**formula = "2*(3+(4+5))"**

**formula:find("%b()") returns 3, 11.**

## Replace

If you want to replace the substring with another string, use something like:

**newstring = oldstring:gsub("=", "->")**

This returns a new string with all occurrences of "=" replaced with "->". If you want only the first occurrence to be replaced, call oldstring:gsub("=", "->", 1). If no occurrence of the substring is found, the original input string is returned unchanged. The gsub method uses the same pattern matching facility as find.

For a nice application of this, what about a "pretty print" function that replaces ugly "^2" and "^3" with the much nicer "$^2$" and "$^3$" (and a few others) in entered expressions?

**NOTE that certain common symbols, such as "^" and "+" are part of Lua pattern definitions. Using the "%" ignores this and simply uses what follows.**

Study the function that follows and see what each line achieves - this would convert a quadratic such as "-1x^2+-0x+-4" into "-x$^2$-4". (For a more complete Pretty function, download and study either of the attached documents, pretty.tns or pretty.lua)

```
function pretty(input)

    input = tostring(input)

    input =
    input:gsub("%+%-", "-")

    input =
    input:gsub("0x%^2%+",
    "")

    input =
    input:gsub("0x%^2%-",
    "")

    input =
    input:gsub("0x%+", "")

    input =
    input:gsub("0x%-", "")

    input =
    input:gsub("%+0", "")

    input = input:gsub("%-
```

*Click anywhere on this image for a video demonstration*

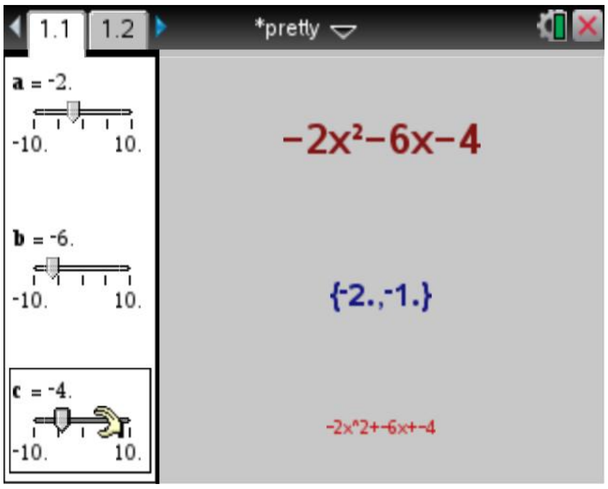
```
        0", "")

        input =
        input:gsub("1x", "x")

        input =
        input:gsub("%^2","²")

        return input

    end
```



## Split

If you would like to split a string into the parts before and after the equal sign, use something like:

**parts = equation:split("=")**

**left = parts[1]**

**right = parts[2]**

You could do it all in one statement with:

**left, right = unpack(equation:split("="))**

---

*Back to Top*

Continue your exploration of further powerful and useful Lua functionality in Lesson 17, where we cover the **math.eval** function, which allows Lua to reach across and make use of TI-Nspire functionality!

---