The Append Operator in Ruby.

In mathematics, an operator is a symbol, such as:

+

or:

−

that performs an arithmetic work such as:

addition

or:

subtraction

respectively.

‘Operator’ is Latin for ‘worker.’

Programming is no different when it comes to its operators. If you wish for a program to work, you must use operators!

In Ruby:

≪

is the append operator.

In anatomy, *appendages*, such as arms and legs, are those organs that *hang on* to the trunk, or core.

The term ‘to append,’ etymologically, means ‘to hang [something] on to [something.]’

‘**pendō**)*present infinitive*: ‘**pendere**,’ *perfect active*: ‘**pependī**,’ *supine*: ‘**pensum**’); *third conjugation*.

is the Latin verb, ‘to hang.’

‘ad’

is the Latin preposition that means:

‘to, toward.’

Affix the preposition, ‘ad,’ to the verb, ‘pendō,’ and we get ‘appendō,’ which is the Latin verb, ‘to hang [something] towards [something else.]’

In Ruby programming, we can assign a string-literal value to a variable like so:

a = “Hello”

In Ruby, strings are mutable[1]

Should we wish the variable:

a

to contain the string-literal data:

“Hello, world!”

we could simply reassign the variable:

a

[1] From the Latin 1st-conjugation verb, ‘mūtō, mūtāre, mūtāvī, mūtātus,’ which means ‘to change,’ and the Latin 3rd-declension adjective, ‘habilis, habile,’ which means ‘having,’ whence we derive the Latin adjective-making suffix, ‘-abilis, -abile.’ Combine the Latin verb, ‘mūtō,’ with the suffix, ‘-abilis, -abile’ and we get the 3rd-declension Latin adjective, ‘mūtābilis, mūtābile,’ which denotes something that *has* [the *ability*] to *change*. In Ruby Programming, as distinct from other programing languages, strings *have* the *ability* to *change*.

to:

“Hello, World!”

thus:

a = “Hello, world!”

but this is not necessary! A more efficient way would be to append the string-literal data:

“, world!”

to the string-literal value:

“Hello”

by using the append operator:

<<

The following is how we do it[2]:

a = “Hello” ↵

=> “Hello”

a ↵

=> “Hello”

a << “, world!” ↵

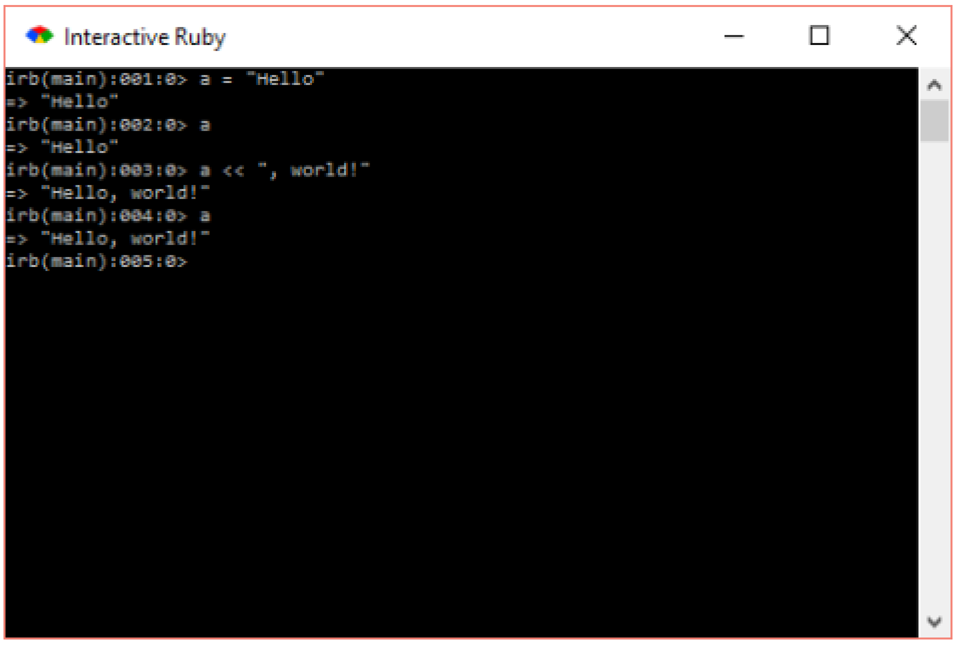
=> “Hello, world!”

a ↵

=> “Hello, world!”

[2] The arrows, ↵ , represent the pressing of the return key.

Below is an image of this appending’s being done in an Interactive Ruby Window:



```
irb(main):001:0> a = "Hello"
=> "Hello"
irb(main):002:0> a
=> "Hello"
irb(main):003:0> a << ", world!"
=> "Hello, world!"
irb(main):004:0> a
=> "Hello, world!"
irb(main):005:0>
```

Figure 1: This is a screenshot that I took with *Snipping Tool*, a *Windows-10* application. Because "world" is, in this instance, in the *vocative case*, i.e. the case of direct address – you are saying "hello" to it, remember! – in English punctuation, a comma must, therefore, go before it.