Full Stack Notes

Constants, Variables, String and Numbers

In Ruby we use `snake_case` to define our variables. The most basic of data types (integers, floats, and strings) are object on which you can call methods.

Table of Contents

Constants

You can also create constants in Ruby. A constant holds a value that you do not expect to change. Constant names must start with a capital letter.

Unlike other languages, Ruby allows you to change the values in constants by assigning a new value to them:

```
NUMBER_OF_ARMS = 2
NUMBER_OF_ARMS = 1
```

However, you'll get a warning each time you do this:

```
warning: already initialized constant NUMBER_OF_ARMS
```

Variables

Ruby can store your data in variables. Variables are named placeholders that can store numbers, strings, and other data. Any lowercase word is a variable in ruby. Variables may consist of letters, digits and underscores.

You reference the data stored in a variable by using the variable's name. For example, to store a value of `-5` in a variable named `temperature_on_mars`, you assign that variable the value.

```
temperature_on_mars = -5
puts temperature_on_mars * 2
```

On the second line of the above program we are outputting twice the value of the number stored in our variable.

Strings

Strings are any sort of characters (letters, digits, punctuation) surrounded by quotes. Both single and double quotes are used to create strings.

```
opening_line  = 'It was the best of times.'
movie_trailer = "In a land that time forgot, "
```

Concatenation

We can add to a Strings by using the plus operator.

```
opening_line   = opening_line + " It was the blurst of times."
movie_trailer += 'there grew a ninja '
movie_trailer += "who would never be forgotten."
```

RESOURCES

- String - RubyDocs

Expression Interpolation

Ruby performs expression interpolation on double-quoted strings using the `#{}` interpolation wrapper.

So instead of this:

```
time = "4 o'clock"
state_of_town = "all is well"
town_crier = "It's " + time + " and " + state_of_town + "."
```

We can do this:

```
town_crier = "It's #{time} and #{state_of_town}."
```

Note: Expression interpolation will not work with single quoted strings.

Numbers - Integers

The most basic type of number is an integer, a series of digits which can start with a plus or minus sign.

Commas are not allowed in numbers, but underscores are. If you choose, you can make large numbers more readable by marking your thousands with underscores.

```
very_large_number  = 34_000_000_001
even_larger_number = very_large_number * 12_345
puts "Whoa #{even_larger_number} is a large number!"
```

Output:

```
Whoa 419730000012345 is a large number!
```

RESOURCES

- Integer - RubyDocs

Numbers - Floats

In Ruby decimal numbers are called floats. They consist of a number with a decimal place or a number that uses scientific notation.

```
ALMOST_PI = 3.14
AVOGADROS_CONSTANT = 6.02214179e23
```

Note that both of these variables are constants because their names begin with a capital letter.

RESOURCES

- Float - RubyDocs