Intro to Ruby

**String**- “Davian”

---The quotes mark the beginning and the end of the string, and are required.

**Method**- “Davian”.reverse , “Davian”.length

--Methods will always start with the dot and a usual English language word.

**Numbers and strings**- “Davian” \* 5 it will ultiply your name by 5 so “Davian DavianDavianDavianDavianDavian”

**New methods**- You cant use methods on numbers such as 40.reverse so instead you do 40.to\_s.reverse

* **to\_s** converts values to **s**trings.
* **to\_i** converts values to **i**ntegers (numbers.)
* **to\_a** converts values to **a**rrays.

**Arrays**- is pretty much a list of strings, or numbers [12, 47, 35]

**Arrays into methods**-  [12, 47, 35].mix

**Variables**- Ruby variables help store data Tickets = [12, 47, 35]

--Tickets is the variable, the array is the stored data which the variable is defining.

**Changing specific words in text:** Print poem means to show the text, poem is the name if the variable where the poem is inside of.

So changing words you do poem which is the variable name then a string inside an array [‘toast] and whatever you want to change it to =’honeybrew’

* poem[‘toast’] = ‘honeybrew’
* The two brackets are like a crosshairs used to line up precisely on a target. These brackets mean, "I am looking for \_\_\_\_ somewhere in here."

**How to do things to the complete poem:** poem.lines.to\_a.reverse and produced some Ruby magic.

First, you turned the poem into a list using lines.to\_a. The lines component decided the way the string should be split up, and then one of our "to" methods, to\_a, converted those splits into an Array. (**to\_a**rray.)

Different methods, such as bytes and chars can be used in place of lines. By using lines here, Ruby split the poem up according to each new line.

After that, you reverse'd your Array. You had each line prepared in advance. And then you reversed them. That's it!

**print poem.­lines.to\_a­.reverse.j­oin :**

Good show, my friend! The join method took that list of reversed lines and put them together into a single string. (Sure, you could have also just used to\_s.)

Review for this level!!:

* **Exclamation Points.** Methods may have exclamation points in their name, which just means to impact the current data, rather than making a copy. No big deal.
* **Square Brackets.** With these, you can target and find things. You can even replace them if necessary.
* **Chaining** methods lets you get a lot more done in a single command. Break up a poem, reverse it, reassemble it: poem.lines.to\_a.reverse.join.

**Hashes:** Hashes store related information by giving reusable labels to pieces of our data.

Books = {} 🡨 Known as a hash

Books[“The outsiders”] = :splendid

Books["NBA founder"] = :abys­mal

Then; Books.length will let you know that they’re 2 books in the hash that for some reason acts as a variable as well.

Books.keys will actually not only tell you the number but will list them out.

Keep in mind that hashes won't keep things in order. That's not their job. It'll just pair up two things: a **key** and a **value**. In your reviews, the key is the book's title and the value is the rating, in this case a **symbol**.

**Seven layers of OSI**

