# 2 Ruby: Introduction

Rails is a web application framework built with the Ruby programming language. Before you learn how to build apps with Rails, it's crucial to have a strong understanding of the Ruby language. Even more important than learning Ruby is learning the basic principles of programming. We'll teach you foundational programming principles with Ruby, thus killing two birds with one stone.

The exercises in this checkpoint cover:

- the basic principles of Ruby syntax,
- how to use strings,
- what variables are,
- how to program mathematical operations,
- the concept of nil, and
- defining methods

As you learn how to program you'll be introduced to many new terms. Here are a few terms that you'll see throughout the Roadmap.

- **Program** - a set of instructions that you give to a computer. Ruby is the language with which we'll write the instructions.
- **repl.it** - an online, interactive Ruby shell. It is a full-feature terminal emulator that lets you experiment with and execute Ruby code.
- **Ruby Interpreter** - is responsible for compiling and running the Ruby code.

Throughout the exercises you'll often want to "print" the results of your code to an output window. There are several ways to do this in Ruby. There is a command named `puts` that you can call before an object, like this:

```
puts "hello world"
```

The `puts` command (it's actually a **method**, but we'll cover that later) converts a value to a string (we'll define that below), and then prints it. There are some values that we don't want to convert to strings though, so `puts` won't work for us all the time. Instead, we'll use the `p` command. You can use `p` in the same manner as `puts`:

```
p "hello world"
```

> The difference is subtle, but `p` prints the true **value** of an object, rather than a string-ified version of the object. This will make more sense soon, but for the sake of simplicity, we'll use the `p` command throughout the exercises, as we'll want to see the **true values** of our code returned to the output window.

## Strings, Variables, Math, and Methods

In the following exercises, we'll introduce you to several new programming topics: strings, variables, math operations, and methods. We'll explain those concepts at a high level first, and then practice the detailed concepts in the exercises.

### Strings

"String" is the data type for words or phrases in Ruby and many other programming languages. Strings are delineated in Ruby using single or double quotes; the string is whatever is encapsulated in the quotation marks. The `"hello world"` in the examples above is a string, and could also be defined as such using single quotes: `'hello world'`.

### Variables

As in algebra, programming languages use variables to store and name values. In Ruby, a variable can store any value, and can be named any word or combination of words, with some exceptions we'll discuss later. In Ruby, it is conventional to name variables with lowercase letters joined by underscores:

```
my_variable_name = "I am a string"
p my_variable_name
```

> What do you expect would be output as a result of calling `p my_variable_name`?

Because Ruby is a "dynamically typed" language, we don't need to explicitly state that

`my_variable` is being assigned a string. We can assign whatever value we desire to any variable of any name.

### Math

Ruby supports all basic mathematical operations. That means we can do things like this:

```
first_num = 4
second_num = 6
p first_num + second_num
p first_num - second_num
p first_num * second_num
```

> What do you expect to be printed as a result of these operations?

The modulo operator – `%` – returns the integer remainder of a quotient. For example, when dividing 5 into 30, the remainder is 0 because 5 divides evenly into 30 six times. Division does not always work out evenly though. When dividing 8 into 30 the outcome is 3.75. The remainder of this division is calculated by multiplying .75 by 8 which results in an integer remainder of 6.

```
p 30 % 5
=> 0
p 30 % 8
=> 6
p 8 % 30
=> 8
```

Learn more about **modulo here**.

### Methods

Methods, which are often called "functions" in other languages, are very similar to mathematical functions. They are given inputs ("arguments"), perform a set of actions on/with those inputs, and then "return" something at the end. Along the way, a method can have many "side effects" — changes that are not explicitly returned but happen as a result of calling the method. We'll cover method syntax in the exercises.

Before you start the exercises below, read our **resource on writing at Bloc**. Then send your mentor a message in the discussion tab of this checkpoint.

- In just a few sentences, the message should describe a toaster (the kitchen appliance) in the terminology of a programming method.
  - What is/are its expected argument(s)?
  - What does it return?
  - What are its side effects?
  - What would happen if you fed it an unexpected argument, like a fork?

Once you've sent your message, try the exercises linked to below. Take notes as you complete the exercises. You can use **Gists** to take notes and share them with your mentor. Gists are a great place to take notes when you are programming because they allow you to easily format, search, and share information with others. You can also keep them private if you prefer.



## Exercises

- ~~Strings~~
- ~~Booleans, Symbols and Variables~~
- ~~Math Operations~~
- ~~Nil~~
- ~~Methods~~

# Solution Walk-Throughs

**Do not watch these videos until after you've attempted your own solutions.** If you struggle to complete any exercises, submit your best effort to your mentor *before watching a solution walk-through*. Submit your answers even if some tests don't pass, as it's important for your mentor to understand your thought process. Discuss difficult exercises with your mentor.

There are many possible solutions to the exercises. A walk-through should provide insight into the thought process of *one possible solution* per exercise. If your solution is different yet passes all the tests, do not change your solution to match what is seen in the video. Instead, discuss your thought process with your mentor.

1. **Strings Solutions**
2. **Booleans, Symbols, and Variables Solutions**
3. **Math Operations Solutions**

assignment completed

?