Ruby - Methods - Introduction

In our previous challenges, we have been using *methods* (`def method() .. end` construct) to abstract compound operations, perform data manipulations and learn various concepts of the language, without talking in much detail about the concept of *methods* themselves, and how they are useful for a programmer. In this set of challenges, we will explore the guiding principles behind methods, learn more features and how to use them efficiently in our programs.

In simpler terms, a *method* is a group of several expressions (block, so to say) that can be referred with a name, can be passed some arguments (input) and are associated with one or more objects.

If you have programmed before, you might notice that the description above sounds almost same as *functions* in other languages (e.g, Python) except the last part which talks about association with one or more objects. It might be a bit non trivial to comprehend since *Classes* haven't been introduced, but what it means is that these methods, even though they appear like global functions, are instead *private methods* of a root `Object` class on which they are implicitly defined and invoked automatically.

So, when you write -

```
def hello_world
   'Eha! Ruby'
end

> hello_world
'Eha! Ruby'
```

You are essentially adding a private method to `Object` class -

```
class Object
   private

   def hello_world2
      'Eha! Ruby'
   end
end

> hello_world2
=> 'Eha! Ruby'
```

This, however, is not the focus of this challenge. Instead, it was just to show you the true object nature of Ruby, and we'll return to it again later during our challenges on classes.

In this challenge, you need to write a method `prime?` that takes an argument and returns `true` or `false` depending on if the number is prime or not.

```
> prime? 3
true
> prime? 17
true
> prime? 22
false
```

Further reading

These methods, unlike *functions* in other object oriented programming language (e.g., Python) are not a *first-class citizens*, because they cannot be passed to other methods as arguments, returned by other methods, or assigned to variables.