Ruby - Methods - Keyword Arguments

In our previous challenge, we explored one way to pass a variable number of arguments to our methods. While it may seem handy feature to have, except few circumstances, you are never going to use that many variables for your method. Also, if your are passing several different types of variables to the method which then will be assigned to the array, it can be difficult for the programmer invoking the method to remember the proper order for those arguments.

Ruby allows you to (partially) mitigate this problem by passing a `Hash` as an argument or one of the arguments. For example, you have a method that takes a URI to download a file and another argument containing a `Hash` of other named options (proxy, timeout, active-connections etc.,)

```
def fetch_file(uri, options)
    if options.has_key?(:proxy)
        # do something
    end
end
```

The main problem with this technique, however, is that you cannot easily set default value for arguments (Read more). Since this construct is highly useful, Ruby 2 introduced *keyword arguments* which allows you to write -

```
def foo(x, str: "foo", num: 424242)
  [x, str, num]
end

foo(13) #=> [13, 'foo', 424242]
foo(13, str: 'bar') #=> [13, 'bar', 424242]
```

Also, introducing the *double splat* (**) which similar to it's counterpart collects all the extra named keywords as a hash parameter.

```
def foo(str: "foo", num: 424242, **options)
  [str, num, options]
end

foo #=> ['foo', 424242, {}]
foo(check: true) # => ['foo', 424242, {check: true}]
```

In this challenge, your task is to write a method `convert_temp` that helps in temperature conversion. The method will receive a number as an input (temperature), a named parameter `input_scale` (scale for input temperature), and an optional parameter `output_scale` (output temperature scale, defaults to Celsius) and return the converted temperature. It should perform interconversion between Celsius, Fahrenheit and Kelvin scale.

For example,

```
> convert_temp(0, input_scale: 'celsius', output_scale: 'kelvin')
=> 273.15
> convert_temp(0, input_scale: 'celsius', output_scale: 'fahrenheit')
=> 32.0
```

Note that the input values are lowercase version of corresponding scales.