Ruby - Methods - Variable Arguments

In our previous challenges, we explored some ways to pass arguments to our methods; however, they were limited in terms of the number of arguments we can pass to them. Using default parameters allows us to lower the number of arguments in some cases, but it's not helpful in situations where we need to pass a variable (and potentially very large) number of arguments.

Consider a method that computes the sum of numbers. Obviously, you wouldn't want to write different methods accounting for some variable number of arguments (e.g.: summing two numbers, three numbers, four numbers, etc.). This is where Ruby's * comes into play. Take a look at the following code:

```
def sum(first, *rest)
   rest.reduce(first) { |o, x| o + x }
end

> sum(1) # first = 1, rest = []
1
> sum(1, 2) # first = 1, rest = [2]
3
> sum(1, 2, 3) # first = 1, rest = [2, 3]
6
```

Prepending an asterisk (*), or *splat*, to a parameter assigns all of the values starting from that position in the method call to an array named $\boldsymbol{rest}$ inside the method. In this case, our method has at least one required parameter named $\boldsymbol{first}$, and then any subsequent values are assigned as an array to $\boldsymbol{rest}$.

Write a method named *full_name* that generates the full names of people given their first name, followed by some variable number of middle names, followed by their last name.

Sample Input 0

```
> full_name('Harsha', 'Bhogle')
```

Sample Output 0

```
"Harsha Bhogle"
```

Sample Input 1

```
> full_name('Pradeep', 'Suresh', 'Satyanarayana')
```

Sample Output 1

```
"Pradeep Suresh Satayanarayana"
```

Sample Input 2

```
> full_name('Horc', 'G.', 'M.', 'Moon')
```

Sample Output 2

"Horc G. M. Moon"