

```
def all_divisible_nums (num, divisor)
```

```
    return [] if num < divisor  
    return [num] if num == divisor
```

```
    if (num % divisor).zero?
```

```
        all_divisible_nums (num-1, divisor) << num
```

```
    else
```

```
        all_divisible_nums (num-1, divisor)
```

```
    end
```

```
end
```

all_divisible_nums(6, 3)

[3, 6]

```
def all_divisible_nums (num, divisor)
```

```
    return [] if num < divisor  
    return [num] if num == divisor
```

```
    if (num % divisor).zero?
```

```
        all_divisible_nums (num-1, divisor) << num
```

```
    else
```

```
        all_divisible_nums (num-1, divisor)
```

```
    end
```

```
end
```

all_divisible_nums(5, 3)

[3]

```
def all_divisible_nums (num, divisor)
```

```
    return [] if num < divisor  
    return [num] if num == divisor
```

```
    if (num % divisor).zero?
```

```
        all_divisible_nums (num-1, divisor) << num
```

all_divisible_nums(4, 3)

```

    else
      all_divisible_nums(num-1, divisor)
    end
  end
end

def all_divisible_nums(num, divisor)
  return [] if num < divisor
  return [num] if num == divisor
end

```

Diagram annotations:

- A blue arrow points from the `end` of the first block to the `def` line.
- A red bracket under `all_divisible_nums(num-1, divisor)` is labeled `[3]`.
- A red arrow points from the `all_divisible_nums(3, 3)` call to the `all_divisible_nums(num-1, divisor)` call in the first block.

```

if (num % divisor).zero?
  all_divisible_nums(num-1, divisor) << num
else
  all_divisible_nums(num-1, divisor)
end
end

```