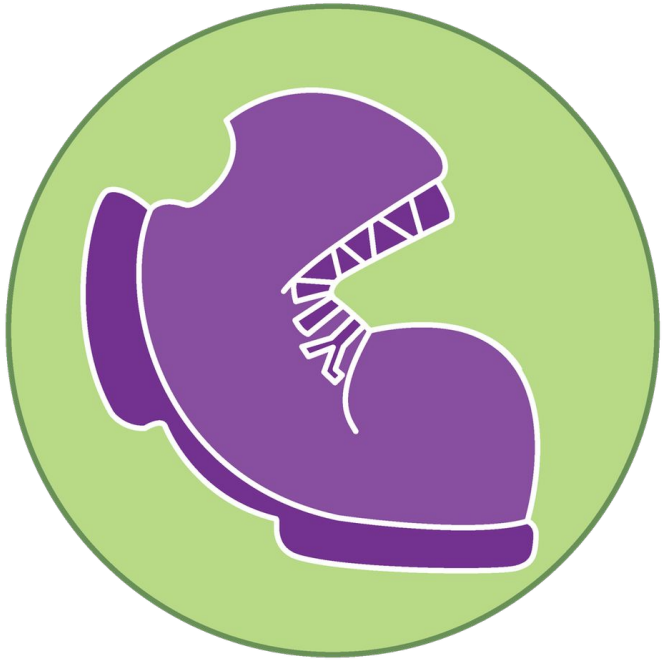
Functions Make Life Easier



@BootstrapWorld

There's Got to Be a Better Way!

I LOVE Green Triangles!

Open your editor and...

write the code to make a solid green triangle of size 50!

There's Got to Be a Better Way!

Make as many unique, solid, green triangles as you can in the next 2 minutes.

There's Got to Be a Better Way!

Look at the examples below and consider the following: What is constant? What stays the same? What is variable? What changes?

```
triangle(30, "solid", "green")

triangle(12, "solid", "green")

triangle(500, "solid", "green")
```

There's Got to Be a Better Way!

Our programming language allows us to define *values*. This lets us create "shortcuts" to reuse the same code over and over.

For example:

```
gt = triangle(65, "solid", "green")
```

This shortcut makes the same green triangle *every* time!

There's Got to Be a Better Way!

To make a shortcut that *changes*, such as creating solid, green triangles of different sizes, we need to define a *function*.

Suppose we want to define a function called `gt`. When we give it a number, it makes a solid green triangle of whatever size we give it.

I'm going to need a volunteer to help act out the process.

There's Got to Be a Better Way!

Our actor is playing the starring role of `gt`.

They expect a number and will produce an image.

Just as with any acting career, this one will begin with a rehearsal.

When I say, "*gt 5!*", you say, "*triangle 5 solid green!*"

Let's try it.

There's Got to Be a Better Way!

Let's Rehearse!

When I say...	gt says...
gt 5	triangle 5 solid green
gt 20	
gt 200	
gt 99	

Identifying and Describing the Pattern

How was `gt` so speedy at building green triangles?

How can we program the computer to be as smart as our volunteer?

In order to define a function, we need to identify what's changing and what stays the same. Once we've done that, we can define any function we'd like in the *Definitions Area*!

Identifying and Describing the Pattern

On the next slide, we'll take a look at some examples for `gt`.

As you look at the examples, consider the following:

- What's changing?
- How do we define our function?
- What would be a logical variable?

Identifying and Describing the Pattern

```
gt(5)   → triangle(5, "solid", "green")

gt(10)  → triangle(10, "solid", "green")

gt(25)  → triangle(25, "solid", "green")

gt(100) → triangle(100, "solid", "green")

gt(220) → triangle(220, "solid", "green")
```

Identifying and Describing the Pattern

If we keep everything that stayed the same and substitute `size` for the numbers that changed, it looks like this:

```
gt(size) → triangle(size, "solid", "green")
```

The way we write this in the editor is

```
fun gt(size): triangle(size, "solid", "green") end
```

Identifying and Describing the Pattern

Turn to The Great gt domain debate!.

Who is correct - Kermit or Oscar?

Why might someone think the domain for `gt` contains a Number and two Strings?

Identifying and Describing the Pattern

- Open the gt Starter File, and save a copy of your own
- Click Run
- Evaluate `gt(10)` in the Interactions Area. What do you get back?
- In one minute, how many different green triangles can you make?!
- How did making green triangles with `gt` compare to making them with your previous strategy?

Identifying and Describing the Pattern

Other people have other favorite shapes. We need to be able to meet their needs, too! What if we wanted to define a function `rs` to make solid red squares of whatever size we provide?

- Complete Let's Define Some New Functions!
- Add your new function definitions to your gt Starter File and test them out.
- Then do Let's Define Some More New Functions!
- And Describe and Define Your Own Functions!

Identifying and Describing the Pattern

- Why is defining functions useful to us as programmers?
- In math class we mostly see functions that consume numbers and produce numbers. What other data types did you see being consumed by these functions?