Using Subprograms to   
 Organize Your Code

**Algebra**

**Unit 3 Line Master 8a**

Part 1: Application that Calculates the Area of a Rectangle

Let’s revisit the application that determined the area of a rectangle and reorganize the code by creating subprograms.

A ***subprogram*** contains a small set of instructions to complete a task   
and is referenced and called to action by the main program.   
In Scratch, when we make a new block, we have made a subprogram.

Use the link to the application you used to determine areas of rectangles in Master 7:

<https://scratch.mit.edu/projects/805357000/editor/>

If you have a Scratch login, save the project in your Scratch   
account by selecting Remix at the top of the screen.   
A login is not required to work with the code, but you will not   
be able to save your changes without it.

This pseudocode shows how we will reorganize the application  
into 3 subprograms and a main program that calls them up.

*Obtain Input Subprogram*

|  |
| --- |
| subprogram obtainInput  output “Enter the length of the rectangle in  centimetres:”  **length** = user input  output “Enter the width of the rectangle in  centimetres:”  **width** = user input |

*Calculate Area Subprogram*

|  |
| --- |
| subprogram calculateArea  **area** = **length** \* **width** |

Let’s revisit the application that determined the area of a rectangle and reorganize the code by creating subprograms.

Using Subprograms to   
 Organize Your Code (cont’d)

**Algebra**

**Unit 3 Line Master 8b**

*Output Info Subprogram*

|  |
| --- |
| subprogram outputInfo  output “The area is “, **area**, “ square centimetres.” |

*Main Program*

|  |
| --- |
| output “I'll calculate the area of your rectangle.”  run obtainInput subprogram  run calculateArea subprogram  run outputInfo subprogram |

1. Start by creating a subprogram containing the code that requests information about length and width from the user.

* Graphical user interface, application, website, Teams

  Description automatically generatedTo create a subprogram, select   
  My Blocks, Make a Block.   
  Name the first subprogram obtainInput and click OK.
* Pull apart the blocks from the original program.

Drag the relevant ask and set blocks so they appear beneath the obtainInput subprogram as shown here.

**Graphical user interface, text, application, chat or text message

Description automatically generated**

Using Subprograms to   
 Organize Your Code (cont’d)

**Algebra**

**Unit 3 Line Master 8c**

1. Create a second subprogram called calculateArea.   
   Drag the set area calculation code to appear beneath it.

Graphical user interface, application

Description automatically generated

3. Create a third subprogram called outputInfo.   
 Drag the say block that states the calculated area to appear   
 beneath it.

Graphical user interface, text, application, chat or text message

Description automatically generated

4. Now you can adjust your main program to “call” the subprograms   
 into action, as shown below.

A picture containing text

Description automatically generated

* You have broken down the main program into smaller, more manageable steps. This means there is less code in the main program. It also helps to clarify what is happening in each part.
* If you need to debug or correct errors in the application, you can focus on one of the subprograms rather than having to look at all of the code at once.
* Try the application with dimensions you used in Part 1 of Master 7 to confirm it outputs the correct areas.

Using Subprograms to   
 Organize Your Code (cont’d)

**Algebra**

**Unit 3 Line Master 8d**

Part 2: Application that Calculates the Area of a Circle

Now that you know how to create subprograms, you can incorporate subprograms into the application you used to calculate the area   
of a circle in Master 7. Use this link to the application without subprograms:

<https://scratch.mit.edu/projects/805564875/editor/>

Create subprograms for each part of the application.   
Follow the pseudocode provided below. Be sure to name your subprograms based on the names used in the pseudocode.  
Check your revised program using diameters from Part 2 of Master 7.

*Obtain Input Subprogram*

|  |
| --- |
| subprogram obtainInput  output “Enter the diameter of the circle in  centimetres:”  **diameter** = user input |

*Calculate Radius Subprogram*

|  |
| --- |
| subprogram calculateRadius  **radius = diameter/2** |

*Calculate Area Subprogram*

|  |
| --- |
| subprogram calculateArea  **area** = **pi** \* **radius** \* **radius** |

*Output Info Subprogram*

|  |
| --- |
| subprogram outputInfo  output “The area is “, **area**, “ square centimetres.” |

*Main Program*

|  |
| --- |
| pi = 3.14  output “I'll calculate the area of your circle.”  run obtainInput subprogram  run calculateRadius subprogram  run calculateArea subprogram  run outputInfo subprogram |