Chapter 10: Validating Email Addresses

We are going create another application to validate an email address. This time, we are going to focus on making sure that the user supplies a name (at least two characters long) before the @ symbol and that they have entered a top-level domain (after the last dot) which must also be least two characters long. There is more to validating an email address, but this is an example of how you can use the split blocks to examine a string.

Later, we will have exercises and projects to validate further that a host is entered and that the email address starts with an alphabetic letter.

The first three steps are exactly like the Validate Email Practice. You can just modify that practice. Or create a new one

Step 1: Drag a Label from the Basic Palette onto the screen. Rename it LabelConfirmation. Clear the text property.

Step 2: Place a TextBox on the screen and rename it TextBoxEmail. Change the Hint property to Enter Email.

Step 3: Place a Button on the screen and rename it ButtonGo. Change the text property to Go.

Step 4: Open the Blocks Editor. The first thing we will need to do is create a Boolean variable that will store whether or not the email address is valid.

Step 5: Go to the Variables drawer and choose an initialize global to block. Change the name to emailValid.

Step 6: Go to the Logic drawer and find a false block, plug that into the emailValid variable.

Step 7: Now you will create a variable to hold the list that is returned when we split the email input. Go to the Variables drawer and choose an initialize global to block. Change the name to emailList.

Step 8: We need to plug in a create empty list block from the List drawer to make this a List variable.

In the following steps (9-31) you will create a procedure that validates the name.

Step 9: Go the Procedures drawer and choose a to procedure do block.

Step 10: Click on the name procedure and change the name to validateEmail. We will use this procedure to set the value of the emailValid to either true or false based on whether there is an @ symbol and that there is text before the @ symbol.

Step 11: The first step in the procedure is to set the emailValid variable to false. You can find the set global emailValid block by hovering the mouse cursor over the name emailValid in its initialization block. You will find the false block in the Logic drawer. Plug that set of blocks into the procedure.

Step 12: Now you will split the list at the first @ symbol. Like you did for emailValid, hover over the name emailList in its initialization block to find the set global emailList to block.

Step 13: Go to the Text drawer and select the split block, change it to split at first and plug it into the set global emailList to block from Step 12.

Step 14: Put the TextBoxEmail.Text block in the text argument slot and the @ symbol in the at argument slot.

Step 15: Go to the Control drawer and select an if-then block.

Step 16: Go to the Logic drawer select an and block.

Step 17: Plug the and into the if-then block.

Step 18: Now let's program the two conditions Is there a name in front of the @ symbol? and Is it more than one character long?

Step 19: Go to the Math drawer and create a greater than block.

Step 20: Go to the List drawer and select the length of list block.

Step 21: Hover the mouse cursor over the name emailList in its initialization block and get the get global emailList block.

Step 22: Go to the Math drawer and select a number block, change its value to 1.

Step 23: Configure the blocks

Step 24: Go to the Math drawer and find a greater than (>) block.

Step 25: Go to the Text drawer and find the length block.

Step 26: Go to the List drawer and find the select list item block.

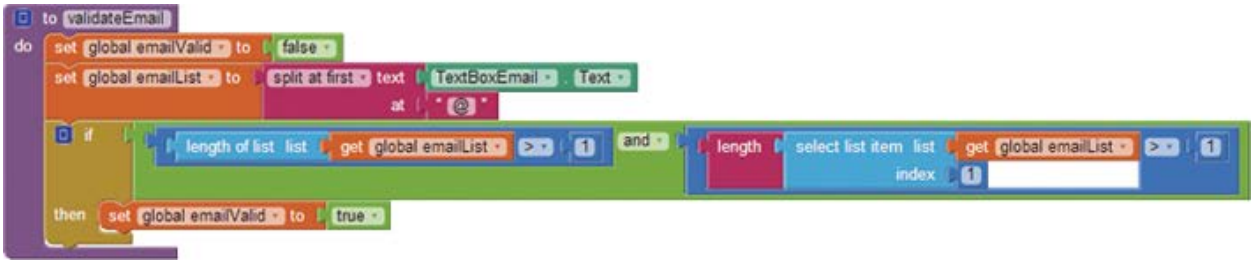
Step 27: Hover the mouse cursor over the name emailList in its initialization block and get the get global emailList block.

Step 28: Go to the Math drawer and select two number blocks, change their values to 1.

Step 29: Configure the blocks

Step 30: Plug both test conditions into the Logic and block

Step 31: If both test conditions are true, set the emailValid variable to true. Compare your work so far with the blocks below.



Part 2: Validate the Top-Level Domain

In the following steps (32-55) you will create the blocks and procedure to validate that the top level domain is at least two characters.

Step 32: Go to Variables and choose an initialize global name to block. Change the name to topDomain.

Step 33: Go to the Text drawer and find an empty text block plug it into the topDomain initialization block.

Step 34: Go to the Procedures drawer and choose a to procedure do block Change the name to validateTopDomain.

Step 35: The first step in the procedure is to set the emailValid variable to false. You can find the set global emailValid block by hovering the mouse cursor over the name emailValid in its initialization block, and you will find the false block in the Logic drawer. Plug that set of blocks into the procedure.

Step 36: Split the list at the first period. Like you did for emailValid, hover the mouse cursor over the name emailList in its initialization block to find the set global emailList to block.

Step 37: Go to the Text drawer and select the split block, and plug it into the set global emailList to block from.

Step 38: Plug the TextBoxEmail.Text block into the text argument slot and the period into the at argument slot.

Step 39: Go to the Control drawer and select an if then block.

Step 40: Go to the Math drawer and create a greater than (>) block.

Step 41: Go to the List drawer and find the length of list block.

Step 42: Hover the mouse cursor over the name emailList in its variable initialization block and click the get global emailList block.

Step 43: Go to the Math drawer and select a number block, change its value to 1.

Step 44: Configure the blocks

Step 45: Plug the set of blocks into the if then block that you created in Step 39, and then plug the if then block into the validateTopDomain procedure.

Step 46: To select the last item in the list, create a select list item block found in the List drawer.

Step 47: Find the get global emailList block by hovering the mouse cursor over the name emailList in its variable initialization block. Plug it into the list socket of the select list item block.

Step 48: Next, create a length of list block from the List drawer. Plug another get global emailList block into it, and plug these combined blocks into the index socket of the select list item block.

Step 49: Now we need to assign the last list item to our variable topDomain. Find the set global TopDomain to block by hovering the mouse cursor over the name topDomain in its variable initialization block. Plug the set of blocks from Step 48 into it.

Step 50: Now that we have the last item, let's use an if statement to test that it is the right length. Go to the Control drawer and select another if then block.

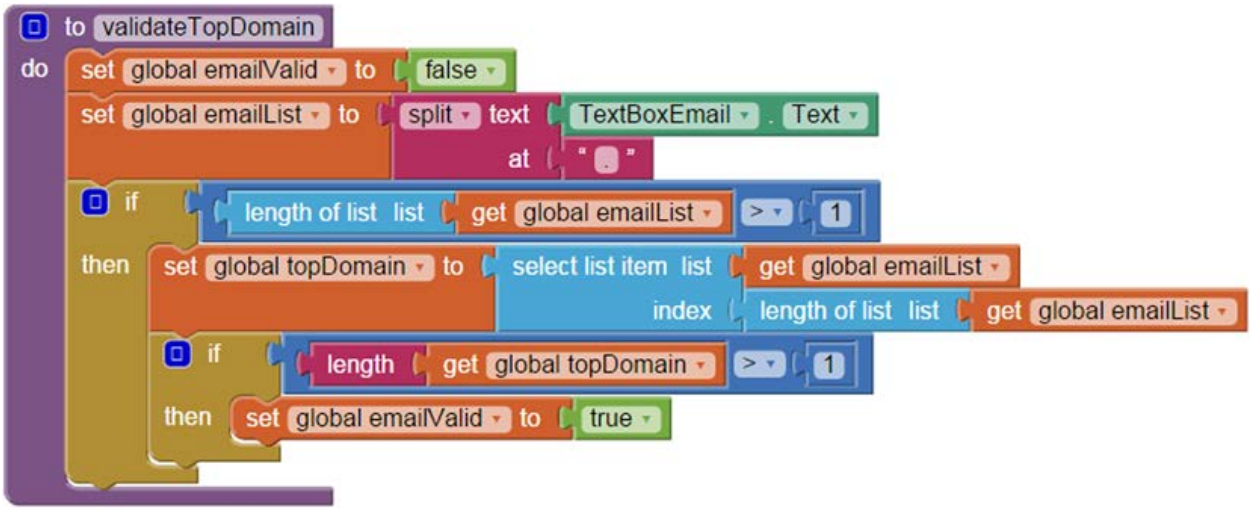
Step 51: You need to test that the length of topDomain is greater than 1. Use the Math greater than block and a number 1 block plugged into the right socket.

Step 52: Go to the Text drawer and find a length block.

Step 53: Hover the mouse cursor over the name topDomain in its variable initialization block to find the get global topDomain block. Plug that into the left slot of the greater than block.

Step 54: Place this set of blocks into the socket of the if then block.

Step 55: If this condition is true, set the emailValid variable to true. Compare your work so far with the blocks below.



Now that we have the procedures written, we will need to call them. We will use the ButtonGo.Click event handler. Each time the button is clicked we will perform the following steps

- Call validateEmail
- If emailValid is false, set the LabelConfirmation.Text to Invalid email
- If emailValid is true, call validateTopDomain
- If emailValid is false, set the LabelConfirmation.Text to Invalid email
- If emailValid is still true, set the LabelConfirmation.Text to Thank you

Step 56: Create the ButtonGo.Click event block from the ButtonGo drawer.

Step 57: Find the set LabelConfirmation.Text to block in the LabelConfirmation drawer.

You will need three of these. Set the first two to a text block with the value Please enter a valid email. Set the third to Thank you.

Step 58: Find the call validateEmail block in the Procedures drawer. Place it in the ButtonGo.Click event handler.

Step 59: Find the if then block in the Control drawer and the get global emailValid value block by hovering the mouse cursor over the name emailValid in its variable initialization block. Connect the two and place in the ButtonGo.Click event handler, under the block that calls the validateEmail procedure.

Step 60: Find the call validateTopDomain block in the Procedures drawer. Place it in the if then block.

Step 61: Create an if then else block from the Control drawer, and a get global emailValid variable block by hovering the mouse cursor over the name emailValid in its initialization block. Connect the variable to the if block and place them in the ButtonGo.Click event handler, under the block that calls the validateTopDomain procedure.

Step 62: Place the set LabelConfirmation.Text to Thank you block into the then socket from Step 61.

Step 63: Place the other set LabelConfirmation.Text to blocks in the two else slots.

Step 64: Run and test your app on your device or emulator. Test valid and invalid scenarios.