Functional Area	**Test Name**	**Test Steps**	**Expected Results**	**Actual Results**
Set up of Micro:bit	Set up success.	1. Connect Micro:bit using USB. 2. Press the 'START' button to connect Micro:bit via web serial.	All buttons are now clickable except for 'START' and 'INTERRUPT'.	*Developer:* expected *Target Audience*: expected
	Set up with no Micro:bit connected	1. Press the 'START' button without connecting Micro:bit using USB.	A pop-up should tell you 'No Paired Serial Devices Available'.	*Developer:* expected
Unclickable buttons	With code running	1. Connect the Micro:bit using the 'START' button. 2. Press 'RUN' at the top of the screen. 3. Click on 'START', 'FLASH', 'RUN', 'REBOOT'. 4. Click on 'HELP'. 5. Click on 'INTERRUPT'.	After step 3, nothing should happen. After step 4, the duck should appear. After step 5, the code should be interrupted and all buttons except 'START' and 'INTERRUPT' are clickable.	*Developer:* expected *Target Audience*: expected
Flashing code to Micro:bit	With an error	1. Connect the Micro:bit using the 'START' button. 2. Type code in the text editor that contains an error. 3. Press the 'FLASH' button.	The code flashes to the connected Micro:bit, but it will only display the error. The helpful duck should appear to give advice on how to deal with the error.	*Developer:* expected *Target Audience*: expected
	Without an error	1. Connect the Micro:bit using the 'START' button. 2. Type code in the text editor that does not contain an error. 3. Press the 'FLASH' button.	The code should run on the Micro:bit and work as expected.	*Developer:* expected *Target Audience*: expected
Editing code	While running on the Micro:bit	1. Connect the Micro:bit using the 'START' button. 2. Clear the editor of all code except 'from Micro:bit import *'. 3. Press 'INSERT FRAGMENT' underneath the first 'While loop' example in the Python Language Features tutorial. 4. Press 'RUN' at the top of the screen. 5. Try to edit the code in the editor.	The code should not be able to be edited while running on the Micro:bit	*Developer:* expected *Target Audience*: expected

	Before running for the first time	1. Connect the Micro:bit using the ‘START’ button. 2. Attempt to change the code in the editor.	The code should be able to be changed before running.	*Developer:* expected
	After running to completion	1. Connect the Micro:bit using the ‘START’ button. 2. Clear the editor of all code except ‘from Micro:bit import *’. 3. Press ‘INSERT FRAGMENT’ underneath the second ‘While loop’ example in the Python Languages tutorial. 4. Press ‘RUN’ at the top of the screen. 5. After the code has been run to completion, try to edit the code in the editor.	The code should be able to be changed after being run to completion.	*Developer:* expected
	After running and being interrupted	1. Connect the Micro:bit using the ‘START’ button. 2. Clear the editor of all code except ‘from Micro:bit import *’. 3. Press ‘INSERT FRAGMENT’ underneath the first ‘While loop’ example in the Python Languages tutorial. 4. Press ‘RUN’ at the top of the screen. 5. Press ‘INTERRUPT’ 6. Try to edit the code in the editor.	The code should be able to be changed after running and being interrupted.	*Developer:* expected
	After running and terminating with a SyntaxError	1. Connect the Micro:bit using the ‘START’ button. 2. Clear the editor of all code except ‘from Micro:bit import *’. 3. Press ‘INSERT FRAGMENT’ underneath the second ‘While loop’ example in the Python Languages tutorial. 4. Change the final line to ‘print(x’ rather than ‘print(x)’. 5. Press ‘RUN’ at the top of the screen.	After step 5, you should see the duck and a SyntaxError message. At step 6, the code should be able to be changed.	*Developer: expected*

		6. Try to edit the code in the editor.		
Autocomplete	After typing ‘a’	1. Connect the Micro:bit using the ‘START’ button. 2. Type ‘a’ into the editor 3. Press tab	After step 2, an autocomplete box should pop up containing ‘accelerometer ’ and ‘audio’. After step 3, ‘accelerometer’ should appear in your editor.	*Developer:* expected
	After typing ‘T’	1. Connect the Micro:bit using the ‘START’ button. 2. Type ‘T’ into the editor 3. Press tab	After step 2, an autocomplete box should pop up containing ‘True’. After step 3, ‘True’ should appear in your editor.	*Developer:* expected
The helpful duck appears	Call on duck without an error	1. Connect the Micro:bit using the ‘START’ button. 2. Press the ‘HELP’ button. 3. Press ‘My code doesn’t do what I want it to do’. 4. Run through a path.	The helpful duck should appear to give advice with whatever is needed.	*Developer:* expected *Target Audience*: expected
	Call on duck with NameError, without following a tutorial	1. Connect the Micro:bit using the ‘START’ button. 2. Change line 5 to say ‘Display.scroll(‘Hello, World!’)’ 3. Press the ‘RUN’ button.	After step 3, the duck should appear with the error message: Error on line 5: NameError: name ‘Display’ isn't defined.	*Developer:* expected
	Call on duck with a SyntaxError, without following a tutorial	1. Connect the Micro:bit using the ‘START’ button. 2. Change line 5 to say ‘display.scroll(‘Hello, World!’)’ 3. Press the ‘RUN’ button.	After step 3, the duck should appear with the error message: Error on line 5: SyntaxError: invalid syntax.	*Developer:* expected
	Call on duck with an ImportError, while following a tutorial	1. Connect the Micro:bit using the ‘START’ button. 2. Clear the editor of all code except ‘from Micro:bit import *’. 3. Press ‘INSERT FRAGMENT’ underneath a piece of code in the Python Language Features tutorial. 4. Edit the code to make it have an error by replacing the word ‘Micro:bit’ with ‘Micro:bit’.	After step 3, the code should appear in the editor. After step 5, the duck should appear, the error should be displayed, and the line with the error should be highlighted. After step 6, the difference between your code and the tutorial code should be highlighted within the duck’s speech bubble.	*Developer:* expected *Target Audience*: expected

		5. Press the 'RUN' button. 6. Navigate through the duck, first pressing 'An error message is displayed', telling it nothing is helping until it compares your error with the tutorial.		
	Call on duck with a NameError, while following a tutorial	1. Connect the Micro:bit using the 'START' button. 2. Clear the editor of all code except 'from Micro:bit import *'. 3. Press 'INSERT FRAGMENT' underneath the first 'While loop' example in the Python Language Features tutorial. 4. Edit the code to make it have an error by replacing the function display with displa. 5. Press the 'RUN' button. 6. Navigate through the duck, first pressing 'An error message is displayed', telling it nothing is helping until it compares your error with the tutorial.	After step 3, the code should appear in the editor. After step 5, the duck should appear, the error should be displayed, and the line with the error should be highlighted. After step 6, the difference between your code and the tutorial code should be highlighted within the duck's speech bubble.	*Developer:* expected
	Call on duck with a TypeError, while following a tutorial	1. Connect the Micro:bit using the 'START' button. 2. Clear the editor of all code except 'from Micro:bit import *'. 3. Press 'INSERT FRAGMENT' underneath the first 'While loop' example in the Python Language Features tutorial. 4. Edit the code to make it have an error by replacing the integer 500 with the string '500'. 5. Press the 'RUN' button. 6. Navigate through the duck, telling it nothing is helping until it compares	After step 3, the code should appear in the editor. After step 5, the duck should appear, the error should be displayed, and the line with the error should be highlighted. After step 6, the difference between your code and the tutorial code should be highlighted within the duck's speech bubble.	*Developer:* expected

		your error with the tutorial.		
The helpful duck disappears	Make the duck disappear by pressing the red ‘X’	1. Press the ‘HELP’ button. 2. Press the red ‘X’.	After step 2, the duck should disappear.	*Developer:* expected *Target Audience*: expected
	Make the duck disappear by pressing ‘Goodbye’ at the end of the flowchart	1. Press the ‘HELP’ button. 2. Press ‘My code doesn’t do what I want it to do’. 3. Press ‘A list has changed even though I didn’t change it’. 4. Press ‘Ah, that’s it!’ 5. Press ‘Thanks Duck, bye for now!’	After step 5, the duck should disappear.	*Developer:* expected
	After making the duck disappear by pressing the red ‘X’ halfway through a path, make the duck reappear	1. Press the ‘HELP’ button. 2. Press ‘My code doesn’t do what I want it to do’. 3. Press ‘A list has changed even though I didn’t change it’. 4. Press the red ‘X’. 5. Press ‘HELP’ again.	After step 5, the duck should start back from the beginning of the flowchart. Rather than the slide it was closed from.	*Developer:* expected *Target Audience*: expected
Testing code from tutorial	Running code directly from tutorial	1. Connect the Micro:bit using the ‘START’ button. 2. Press a ‘RUN EXAMPLE’ from within the tutorial.	After step 2, the example should run on the Micro:bit and behave as expected. The code isn’t expected to stay on the Micro:bit when disconnected from the computer.	*Developer:* expected *Target Audience*: expected
	Inserting a fragment, then flashing	1. Connect the Micro:bit using the ‘START’ button. 2. Clear the editor of any code except ‘from Micro:bit import *’. 3. Press ‘INSERT FRAGMENT’ underneath a piece of code in the Python Language Features tutorial. 4. Press the ‘FLASH’ button. 5. Disconnect the Micro:bit.	After step 3, the code fragment should appear in the editor. After step 4, the code should be flashed to the Micro:bit, behave as expected. After step 5, the program should continue to run on the Micro:bit once reconnected.	*Developer:* expected