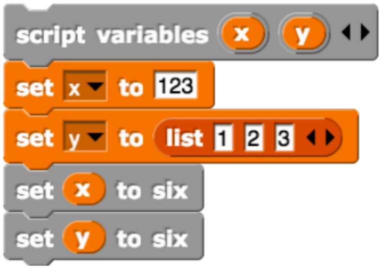
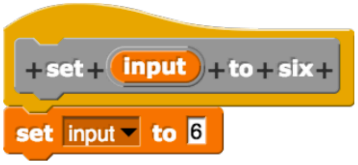
Discussion 5: Mutability

What are the values of the script variables x and y after the given script finishes running?

a.



```
script variables (x) (y)
set x to 123
set y to list 1 2 3
set (x) to six
set (y) to six
```

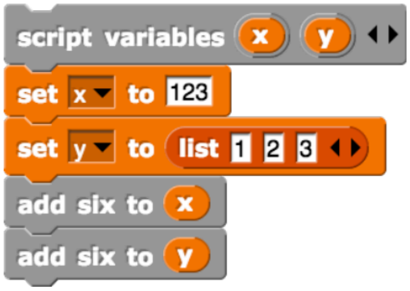


```
+set+(input)+to+six+
set input to 6
```

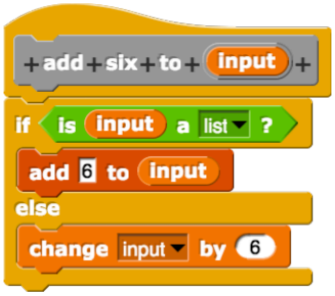
x: 123

y: [1,2,3]

b.



```
script variables (x) (y)
set x to 123
set y to list 1 2 3
add six to (x)
add six to (y)
```



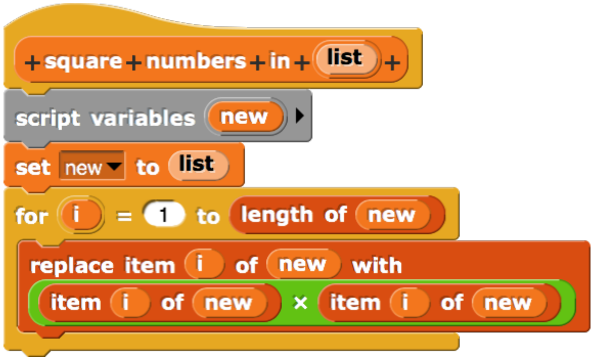
```
+add+six+to+(input)+
if is (input) a list ?
    add 6 to (input)
else
    change input by 6
```

x: 123

y: [1,2,3,6]

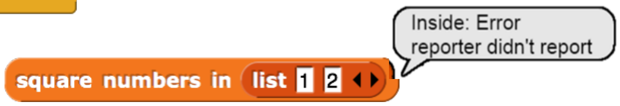
Challenge: Errors Galore

The following questions are based off this block:



```
+square+numbers+in+(list)+
script variables (new)
set new to (list)
for (i) = 1 to length of (new)
    replace item (i) of (new) with
    (item (i) of (new)) × (item (i) of (new))
```

a. When running this block, we get the following error: What caused this error, and how can we fix it?

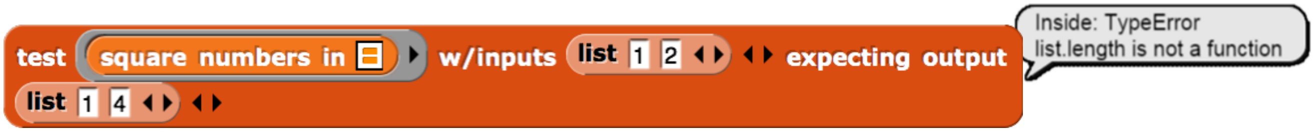


```
square numbers in list 1 2
```

Inside: Error reporter didn't report

The error is due to not reporting at the end of the block definition. We can append [report (new)] after the for loop and the block will work.

b. Now, that we've fixed it, we try to test our code, but we get another error:



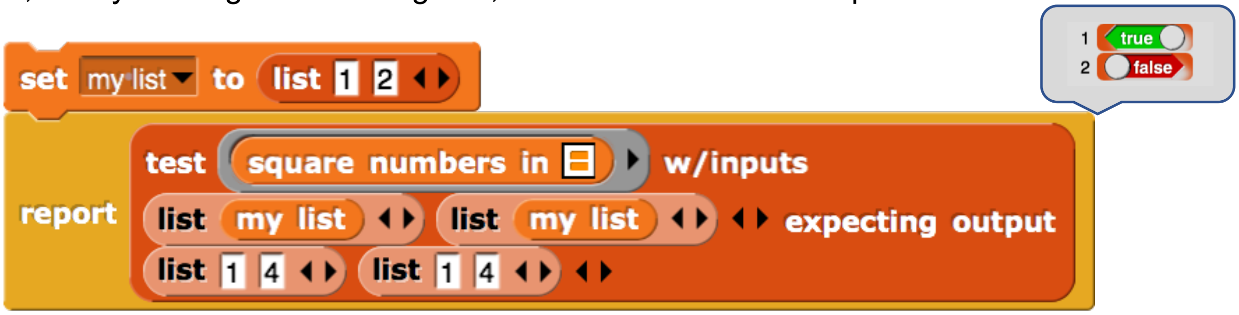
```
test (square numbers in ⊟) w/inputs (list 1 2) expecting output
(list 1 4)
```

Inside: TypeError
list.length is not a function

How can we fix this?

The input list (list [1] [2]) must be nested in a list.

c. Now, we try running the following test, but it doesn't work as expected:



```
set my list to (list 1 2)
report test (square numbers in ⊟) w/inputs
    (list (my list)) (list (my list)) expecting output
    (list 1 4) (list 1 4)
```

1 true
2 false

Why does it report this, and how can we fix it?

The first time we call (square numbers in []) on (my list) we are mutating (my list) so it becomes (list [1] [4]).

When we call (square numbers in []) on (my list) again we then get (list [1] [16]) which is not equal to the

expected output (list [1] [4]). To fix this inside the (square numbers in []) block we can set (new) to (copy of (list)).