Variables & Loops

Intro to CS

Driver Name:		Navigator Name:	

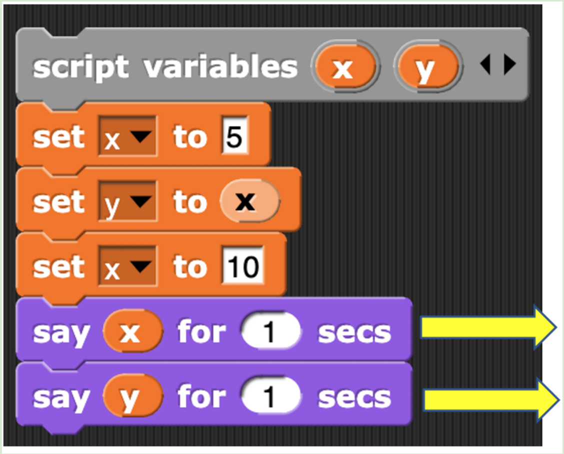
Getting Started

DRIVER ONLY:

1. Log into Snap and start a new project. Save the program as **U2L0-Loops**.
2. Grab a piece of scrap paper for tracing.

NAVIGATOR ONLY: Have this lab open, and ready for reference. Provide guidance on how the driver should proceed when necessary.

Q1. Complete a trace table to trace the values of **x** and **y** in this script:



```
script variables (x) (y)
set x to 5
set y to (x)
set x to 10
say (x) for (1) secs
say (y) for (1) secs
```

Predict what the sprite will say for both of the **say** statements; **fill in the boxes** below with your predictions!

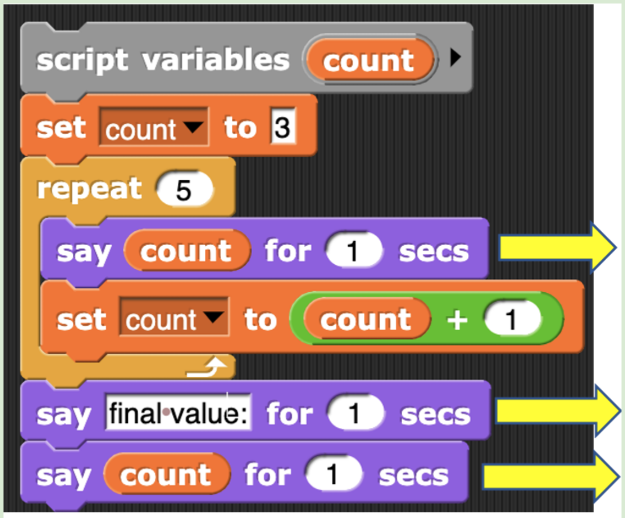
HINT! When you set one variable ("x") to another variable ("y"), you are setting the first variable, "x", to whatever *value* is stored in the second variable, 5 and *not* the variable itself.

Q2. Take the time to **build** the algorithm above in Snap ***exactly*** as shown and execute it to confirm your predictions!

Were you correct? If not, read the *Answer & Explanation* to see why, then **explain** your mistake!

Answer & Explanation

Q3. This example was done together in class. Recall your earlier predictions.



```
script variables (count)
set count to 3
repeat 5
    say count for 1 secs
    set count to count + 1
say final value: for 1 secs
say count for 1 secs
```

The sprite will **say** the current value of count each time through the **repeat loop**, but what will the value of count be *each* time?

Capture your predictions below!

`final value:`

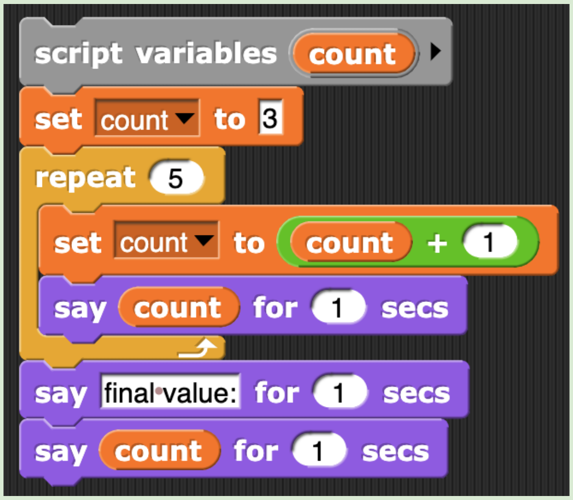
Q4. Take the time to **build** the algorithm above in Snap ***exactly*** as shown and execute it to confirm your predictions! Were you correct? If not, read the *Answer & Explanation* to see why, then **explain** your mistake!	*Answer & Explanation*

Pair Programming Swap

→ DRIVER & NAVIGATOR SWITCH SEATS ←

NEW Driver Name:		NEW Navigator Name:	

Q5. Here's the same algorithm as the one above, except the **set** and **say** commands are ***reversed*** inside the **repeat statement**:	**Predict:** How will *reversing* the two commands inside the **repeat statement** affect what the sprite says?



```
script variables (count)
set count to 3
repeat 5
    set count to (count + 1)
    say count for 1 secs
say final value: for 1 secs
say count for 1 secs
```

Capture what you think the sprite will say now!

```
final value:
```

Q6. In Snap, **adjust** the algorithm you built in step **21** so it matches the algorithm above; execute it to confirm your predictions!

Were you correct? If not, read the *Answer & Explanation* to see why, then **explain** your mistake!

Answer & Explanation

IMPORTANT VOCABULARY & IDEAS!

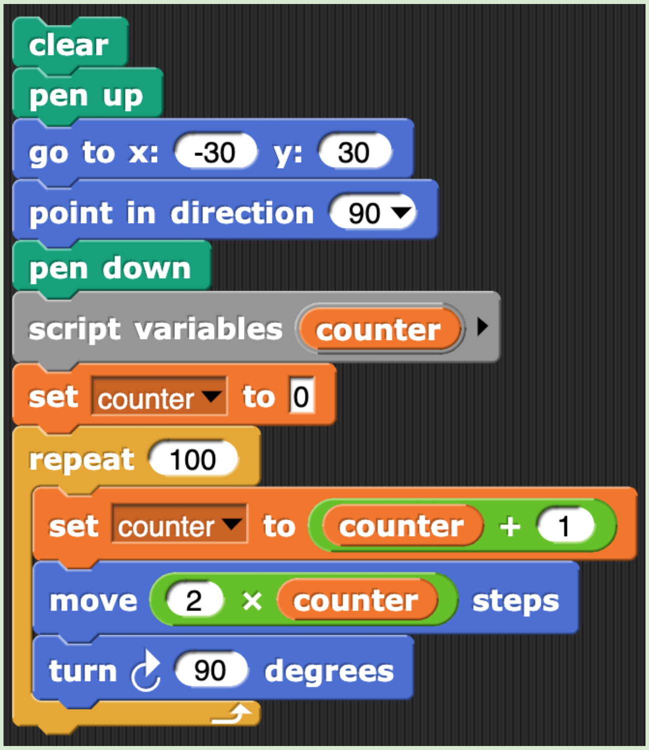
In the algorithm above, the variable **count** is being used as a **counter variable**, which is used to track *which iteration* the loop is in. Each time through a loop, a counter variable ***increases by 1***.

Q7. Build an algorithm using a **loop** and a **counter variable** that will say the numbers "2, 4, 6, 8, 10, 12, 14."

Insert a script pic of your algorithm.

Q8. Analyze this algorithm; what do you *think* it will draw? (this is tricky; give it your best guess!)

Consider how the **counter variable** will affect what is produced on the stage.



```
clear
pen up
go to x: -30 y: 30
point in direction 90
pen down
script variables counter
set counter to 0
repeat 100
    set counter to (counter + 1)
    move (2 × counter) steps
    turn ↻ 90 degrees
```

Now **build** it and see what it draws! Let's call this shape a "**squiral**" (***squ**are spi**ral***; yes, this is a totally made up word!)

Q9. Insert a **stage pic** that shows the "squiral":	
Q10. Explain why the "squiral" *spirals outward*:	

Unsure? Here are some things you can do to 'un-stick' yourself:

- ☐ Trace the value of **counter** in a trace table (maybe only for the first few iterations…)
- ☐ Rubber duck debugging 🐥
- ☐ Consult a neighboring pair
- ☐ Change the values of inputs. Predict how the squiral will change, and verify your hypothesis in *Snap!*

Pair Programming Swap

→ DRIVER & NAVIGATOR SWITCH SEATS ←

NEW Driver Name:		NEW Navigator Name:	

Q11. In the current algorithm, the **counter variable** counts *up* from 0 to 100 by *adding* 1 each time through the **repeat** loop; figure out a way to get it instead to **count *down* from 100 to 0**.

Make the change to your algorithm, and run it to see what happens!

Check your code :)

Insert a script pic of your updated algorithm that counts *down*:	
Q12. What happens to the "squiral" when you count down instead of count up? Why?	

Q13. Try changing the **turning angle** in the algorithm; *instead of 90*, try other numbers such as **43, 92**, **126**, **175**, etc. *Describe what happens!*	
Q14. Try changing the "multiplier" in the 2 × counter reporter block; *instead of 2*, try values like **4**, **3**, **2.5**, **1.5**, and **0.8**. *Describe what happens!*	
Q15. Create a cool looking "squiral" and insert a stage pic to the right:	

Q16. Now, insert the script pic of the algorithm that created your squiral in Q15.	

💾 Save your work.

STOP! It's checkpoint time 😎

Complete this checkpoint: **https://forms.gle/NAh8nojzUXDscCyYA**

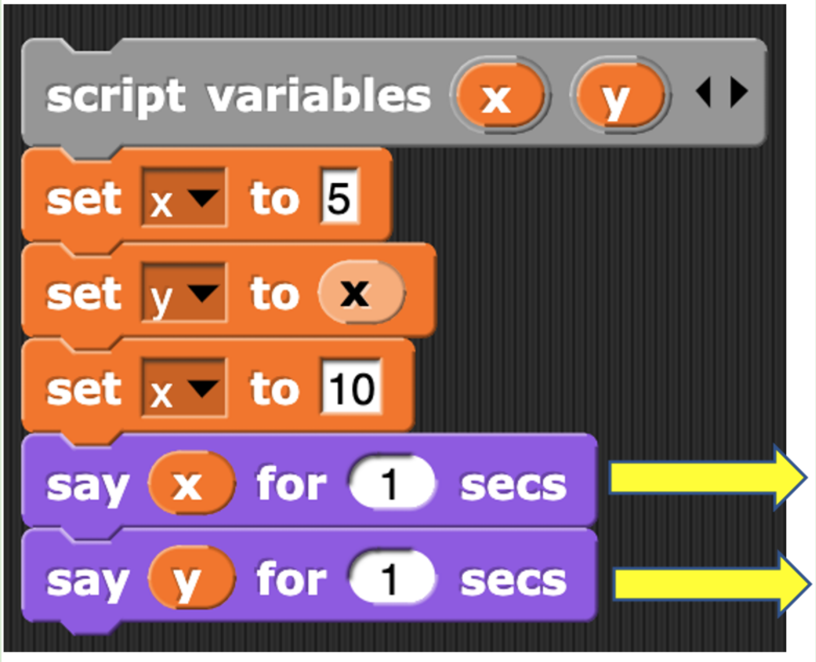
Submit in Google Classroom!

Turn in

Hints

Q1 Explanation (back)

Carefully analyze this algorithm:



```
script variables x y
set x to 5
set y to x
set x to 10
say x for 1 secs
say y for 1 secs
```

Predict what the sprite will say for both of the **say** statements; **fill in the boxes** below with your predictions!

10

5

It says 5 here, NOT 10!

Explanation:

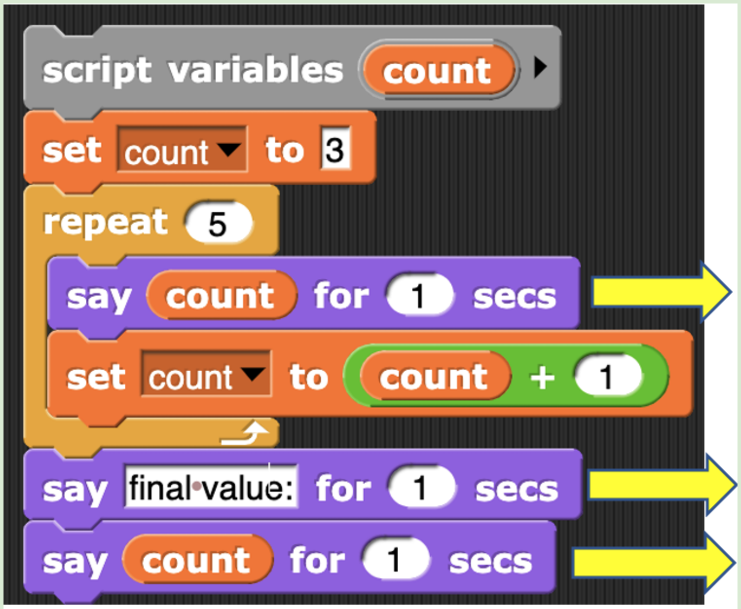
x is set to 5, then y is set to x's *value*, which is 5.

So now x and y are both 5. Then when x gets set to 10, *only* x gets set to 10 -- y is still 5.

A common misconception is that when you update x to be 10, y *also* updates to 10 since it was set to x in the previous line; but in reality, y is set to x's value, which is 5 -- not x itself. y doesn't know anything about x! It only knows the value it was given, which was 5.

Q4 Explanation (back)

Carefully analyze this algorithm:



```
script variables count
set count to 3
repeat 5
    say count for 1 secs
    set count to (count + 1)
say final value: for 1 secs
say count for 1 secs
```

The sprite will **say** the current value of count each time through the **repeat statement**, but what will the value of count be *each* time?

Capture your predictions below!

3	4	5	6	7

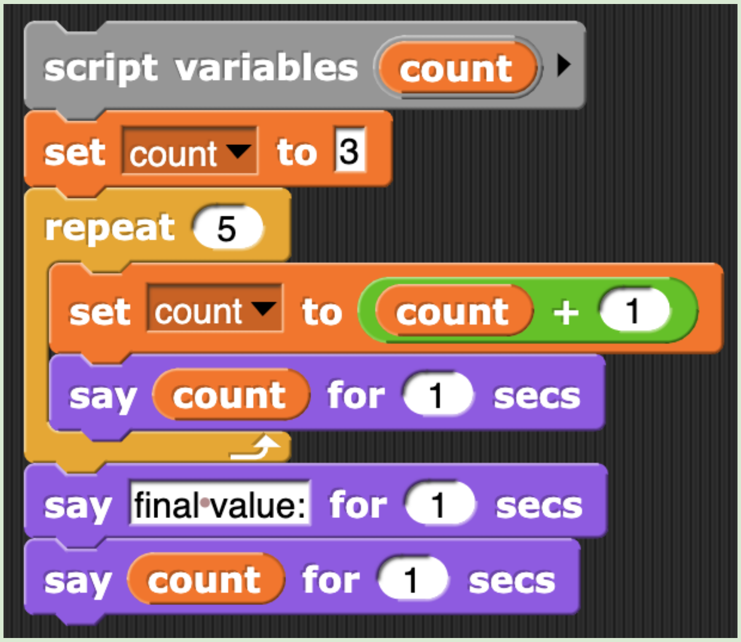
Each time through the **repeat** loop, **count** gets set to the value of **whatever its current value is**, **plus 1**; the first time through, its value is 3, so it gets set to 3 + 1 = 4; the second time through, its current value is 4, so it gets to 4 + 1 = 5, and so on, until the 5th time through, when its current value is 7, so it gets set to 8 (***after*** the sprite says “7”), and the loop ends, at which point the sprite says “final value:” followed by its current value of **8.**

`final value:`

8

Q6 Explanation (back)

Here's the same algorithm as the one above, except the set and say commands are ***reversed*** inside the repeat statement:



```
script variables count
set count to 3
repeat 5
    set count to (count + 1)
    say count for 1 secs
say final value: for 1 secs
say count for 1 secs
```

How does *reversing* the two commands inside the repeat statement affect what the sprite says?

Capture what you think the sprite will say now!

4	5	6	7	8

Everything runs the same way, except the sprite announces the current value of count after it gets increased by 1 (in the previous problem, it stated the value *before* it got increased by 1).

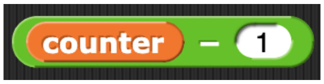
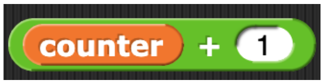
Therefore, in the final time through the loop, count gets set to 7 + 1 = 8, ***then*** the sprite says "8", and then the loop ends, so the sprite says "final value:" and then "8" (again).

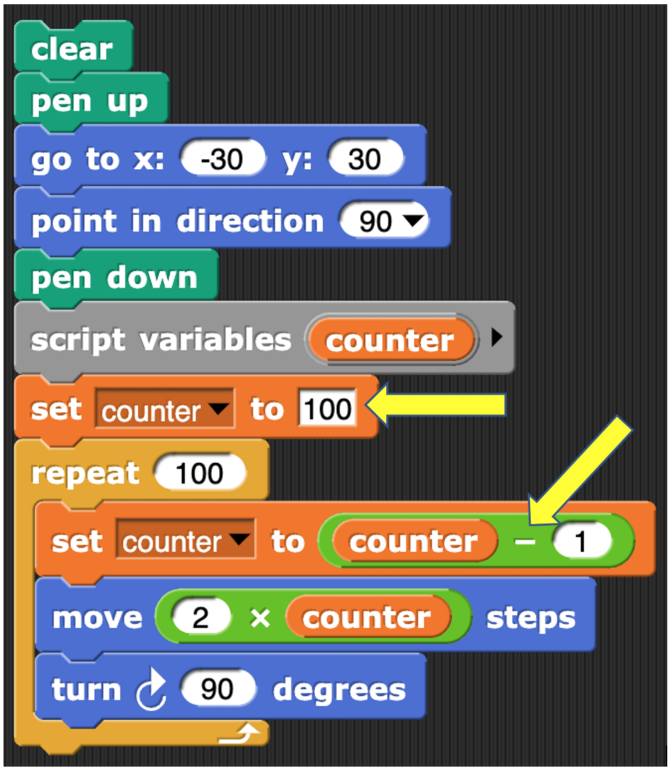
```
final value:
```

8

Q11 Explanation (back)

***Two* changes are needed to make it count *down* from 100 to 0 by 1 each time:**

- Set **counter** to 100 rather than 0
- Use  `counter − 1` rather than  `counter + 1`



```
clear
pen up
go to x: -30 y: 30
point in direction 90
pen down
script variables counter
set counter to 100
repeat 100
    set counter to counter − 1
    move 2 × counter steps
    turn ↻ 90 degrees
```