Teacher: Iulian J Irimina

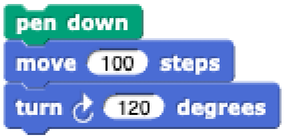
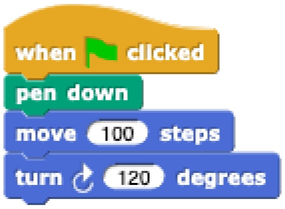
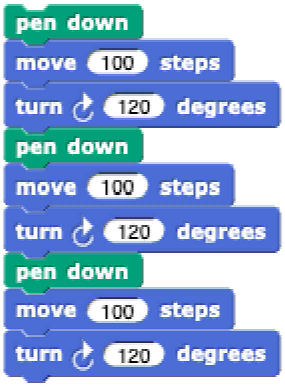
Unit: 1 Intro to Programming, Modern art with programing
Level: AP CSP, 10th grade

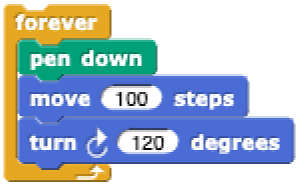
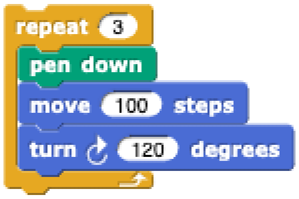
AIM: Students use control structures to control program repetitions.
Essential Question: [Danielson 1c] How do we keep track of program repetitions in order to control the flow of the code?
Standards: [Danielson 1c] Develop a program that effectively uses control structures in order to create a computer program for practical intent, personal expression, or to address a societal issue.

Materials: [Danielson 1d, 1e, 3c] computers, Snap code editor, paper/pencil

Do Now/Warm-up Activity (5-8 minutes): [Danielson 3d]
(Reinforce prior learning, connecting activity to the day's lesson, activity should be engaging)

Today we are practicing drawing shapes in various ways using code. What would the following code snippets generate if you click on it once in Snap editor? Write your answer in the table below.

Code snippet	**Output**
 pen down move 100 steps turn ↻ 120 degrees	It produces a___ because ____
 when 🟩 clicked pen down move 100 steps turn ↻ 120 degrees	
 pen down move 100 steps turn ↻ 120 degrees pen down move 100 steps turn ↻ 120 degrees pen down move 100 steps turn ↻ 120 degrees	

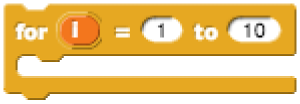
 forever pen down move 100 steps turn ↻ 120 degrees	
 repeat 3 pen down move 100 steps turn ↻ 120 degrees	
Now your goal is to draw a hexagon using code. Explain how you would do that and what blocks you would use. (no code required)	

Students check with each first in pairs.
Students and the teacher go over the do now together and clarify any questions related to the do now. Students explain preferences on which blocks they would use to accomplish the last drawing task. Students discuss ways to repeat tasks in a program.

Mini Lesson – Step by Step Procedures (12 minutes): [Danielson 1a , 3a]
Teach/Model (You do): Teacher models and demonstrates while talking aloud her/his thing process, steps, etc.

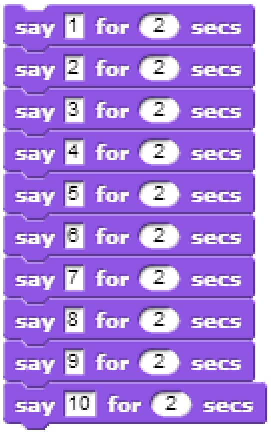
How can we control the program repetition?
For example, how can I create a timer that goes from 1 to 10? - teacher demo

----
Sometimes the script inside a loop needs to know which time through the loop it's in (first, second, etc.).



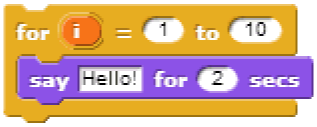
```
for i = 1 to 10
```

You can use to keep track, and you can use its counter (the default name is i) in the repeated script. For example, the **for** block lets you simplify long scripts like:



```
say 1 for 2 secs
say 2 for 2 secs
say 3 for 2 secs
say 4 for 2 secs
say 5 for 2 secs
say 6 for 2 secs
say 7 for 2 secs
say 8 for 2 secs
say 9 for 2 secs
say 10 for 2 secs
```

to

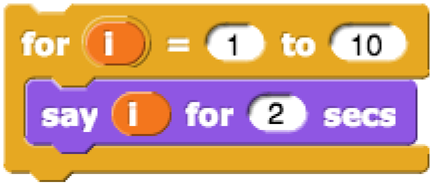


```
for i = 1 to 10
    say Hello! for 2 secs
```

Each time the for block runs the script inside, it changes the value of the counter by 1, beginning with the first input number and ending after the second.

Guided Practice *(We do)*: [Danielson 3c, 3d]

1. Build this script that makes the sprite say the numbers 1 through 10.



```
for i = 1 to 10
    say i for 2 secs
```

   a. Then modify it so that the sprite says 0, 2, 4, 6, 8, ... up through 30.

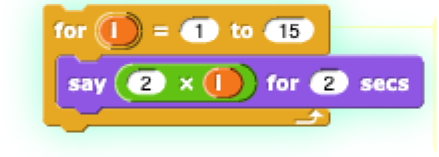
   b. Discuss your solutions with another pair.

Work Period *(They do)* (minutes): [Danielson 1e, 2b, 2c]

Approx 30 min

Task #1
What will the following code snippet produce? Create a table and put all the values of i. What

do you notice about each iteration? What does i represent here?



```
for (i) = (1) to (15)
    say ((2) × (i)) for (2) secs
```

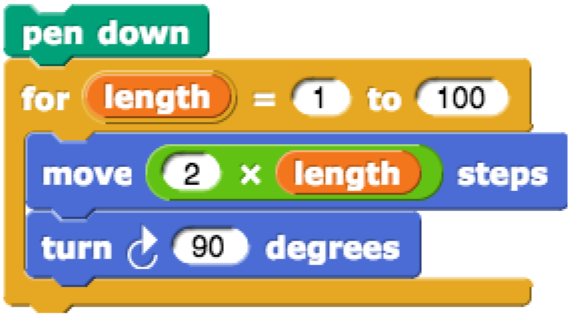
i	value

Task #2

Experiment with spirals.

a. Build this script and try it out:

This design got the nickname "squiral" because it's a square spiral.



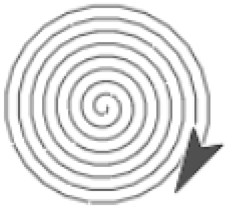
```
pen down
for (length) = (1) to (100)
    move ((2) × (length)) steps
    turn ↻ (90) degrees
```

b. Turn and talk to your partner. Make sure you can explain why the squiral spirals outward.

c. Try switching the order of the 100 and the 1 in the for block in the squiral script. What is the result?

d. Try changing the turning angle in the squiral script to other numbers such as 92, 126, etc.

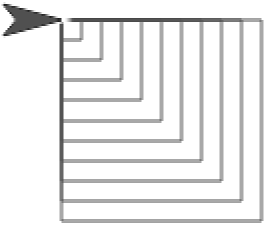
e. Change the inputs to turn and move to get as close as you can get to a smooth spiral:



f. Explain the variable length and its value(s) in the program

g. What question do you have about the tasks #1 and 2 so far? What worked as intended and what didn't work well?

Task #3
When you have finished your task 1, then move on to this task. Think on how you would create the following shape. Discuss with your partner and jot down notes about your choice of blocks.

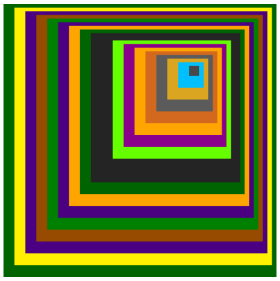


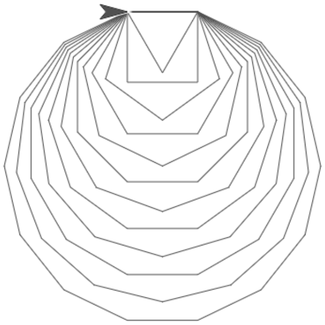
Differentiation/extra : [Danielson 1b, 1c, 3e]

***Support*:** Teacher-student team support. Students benefit from the partner work, notes and feedback from the computer.

***Strategic*:** Students discuss/diagram/draw before coding.

***Enrichment*:** Try coding any of the shapes shown here or you may work on your assortment of shapes. Code starter provided





Share (5 minutes): [Danielson 1c]
Students share choices.

Summary: [Danielson 3c, 3d, 4a]

Controlling the repetitions in a program can happen in various ways. Each choice is good in relation to the intended goal. Using variables as opposed to hard coded values enables us to write a 'better' program as the variables can be redefined and manipulated in various ways.

Homework/Follow-up Activity: [Danielson 3d]
More practice with simple tracing variables in a loop.