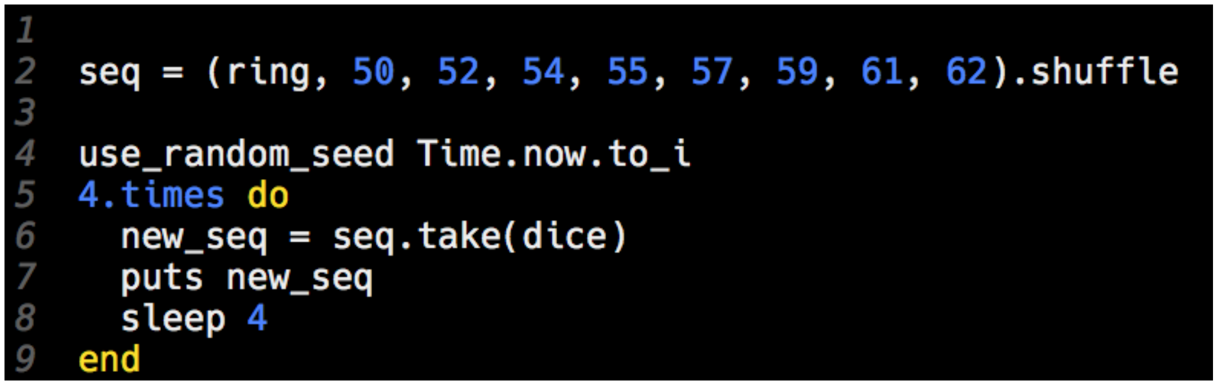
Sonic Pi Generative Music Unit Plan
Lesson # 4 - Looping through data structures of different lengths

Lesson Objectives
Students will be able to iterate through data structures of different lengths in a loop structure.
Suggested Duration
1 period (45 minutes)
NYS Computer Science and Digital Fluency Learning Standards
7-8.CT.9 Read and interpret code to predict the outcome of various programs that involve conditionals and repetition for the purposes of debugging.
Vocabulary
N/A
Assessments
• Assess ____. Check for the ability to: - accurately explain what each line is doing in a block of code   - use the **.length** method to accurately iterate through all indices of a data structure

Do Now
Have students log into Peardeck: Lesson 4 - Different Length Data Structures



```

seq = (ring, 50, 52, 54, 55, 57, 59, 61, 62).shuffle

use_random_seed Time.now.to_i
4.times do
  new_seq = seq.take(dice)
  puts new_seq
  sleep 4
end
```

In peardeck slide, Students should explain what is happening in each line of this block of code.

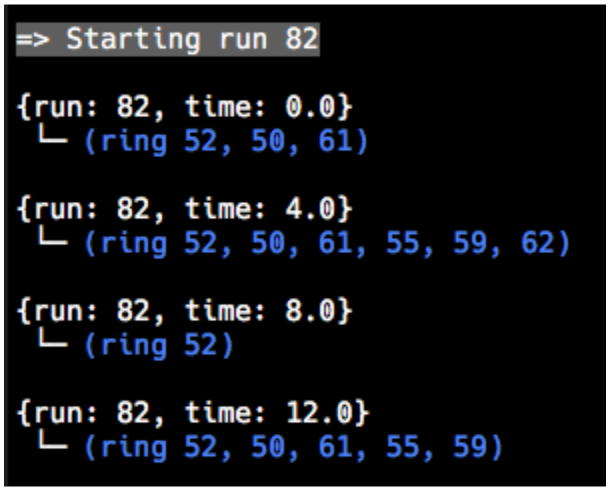
Lesson

Part 1 - Making Predictions

1. After completing Do Now, have students write down predictions about what is going to happen in this code in peardeck slide

2. Have students run this code in Sonic Pi. After the code has run, they should write down each sequence of notes that appears in the console.

<u>**Example of Note Sequences in Console**</u>



```
=> Starting run 82

{run: 82, time: 0.0}
 └─ (ring 52, 50, 61)

{run: 82, time: 4.0}
 └─ (ring 52, 50, 61, 55, 59, 62)

{run: 82, time: 8.0}
 └─ (ring 52)

{run: 82, time: 12.0}
 └─ (ring 52, 50, 61, 55, 59)
```

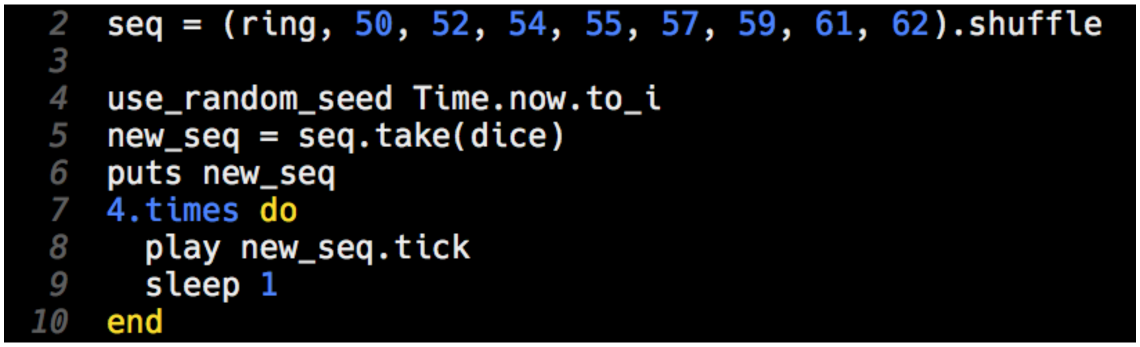
3. Have students write observations in peardeck.
   Share comments which explain that you get a different amount of numbers each time.

4. In peardeck, have students choose which part of the code is causing us to get a different amount of numbers each time

   Answer: ***new_seq = seq.take(dice)***
   .take will choose a certain number from the front of the sequence and dice will alway return a random number between 1 and 6

Part 2 - Identifying Issues with loops #1

1. Have students copy and paste this code from the Peardeck slide into Sonic Pi



```
seq = (ring, 50, 52, 54, 55, 57, 59, 61, 62).shuffle

use_random_seed Time.now.to_i
new_seq = seq.take(dice)
puts new_seq
4.times do
  play new_seq.tick
  sleep 1
end
```

2. If we wanted to play through each of these sequences of different lengths using a .times do/end block, there would be a problem.

   Give students 1-2 minutes to discuss and predict what a potential problem would be. Have students share out responses. Asks students to clarify by explaining which part of the code might cause the problem.

3. Have students make observations in the Peardeck about what happens.
   Remind them to listen and watch the output in the console.
   Students should write their observations in the Peardeck

4. Share out responses.
   Possible responses: ***When the sequence has less than 4 numbers, it repeats some of the numbers.***
   When the sequence has more than 4 numbers, not all the numbers are played

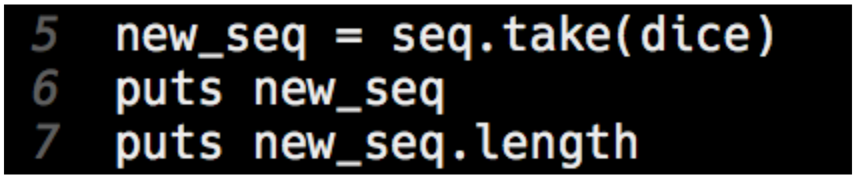
5. Explain that since the loop block happens 4 times, it will always play 4 notes, but this is a problem because each time there is a different amount of notes and there is only a 1 out of 6 chance it will be exactly 4 note.

   We want to find a way that it will always play all the numbers in the sequence, no matter how long the sequence is.

Part 3 - .length - Finding the length of a data structure

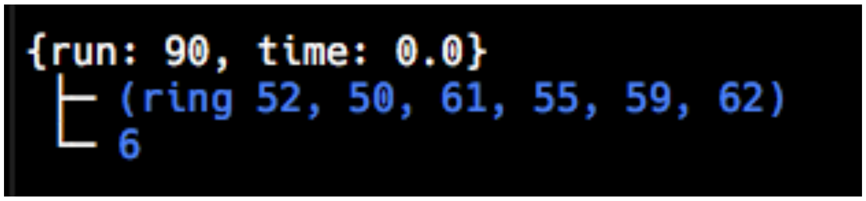
1. Introduce ***.length***
   If we put ***.length*** at the end of a variable containing a data structure (like new_seq), it will return the number of notes in the sequence.

   Demonstrate this by adding ***puts new_seq.length*** to the current code in Sonic Pi



```
new_seq = seq.take(dice)
puts new_seq
puts new_seq.length
```

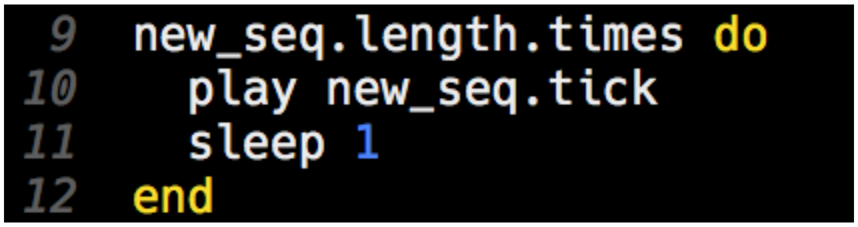
   Have students look a the console results (Examples will vary)



```
{run: 90, time: 0.0}
 ├─ (ring 52, 50, 61, 55, 59, 62)
 └─ 6
```

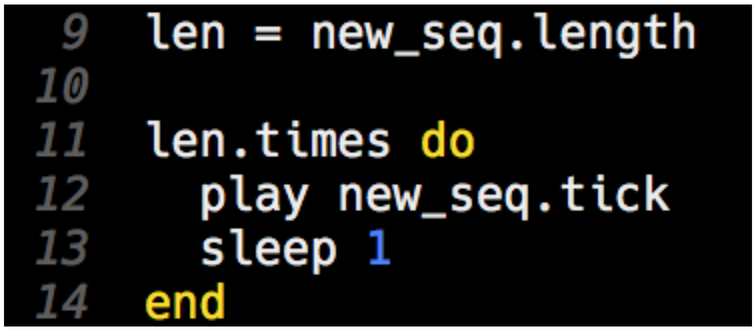
   We see the actual number sequence which has 6 number and below we just see the number six which is what is returned when we use the .length method with a data structure.

2. We can then treat new_seq.length as a variable which holds the value of the length of the sequence of notes and add it to our .times do/end block



```
new_seq.length.times do
  play new_seq.tick
  sleep 1
end
```

   Tell students that if this feels too complex for them, they can create a new variable to store the .length of the sequence and use that with the .times do/end block



```
len = new_seq.length

len.times do
  play new_seq.tick
  sleep 1
end
```

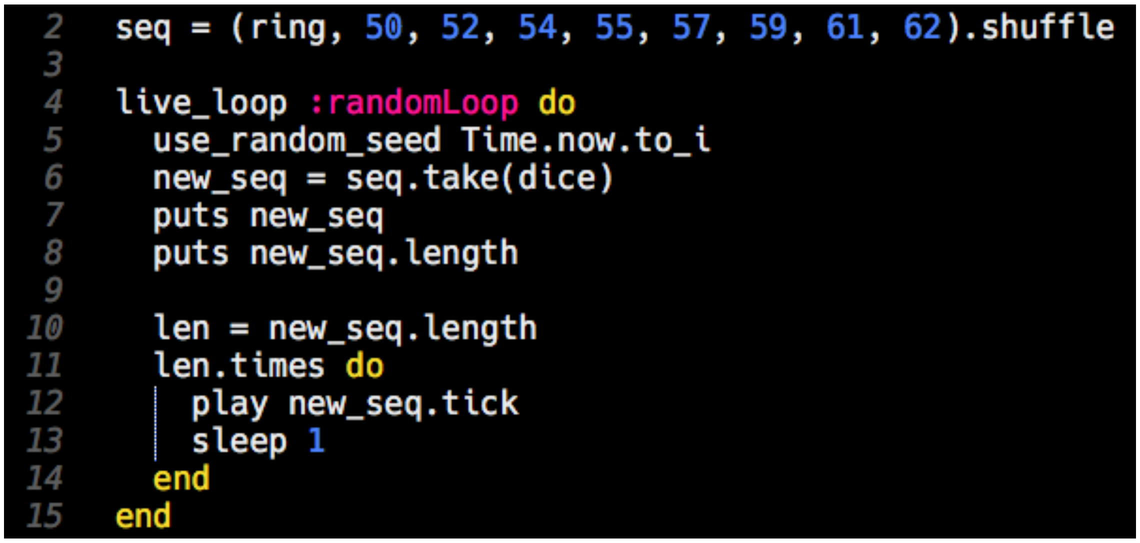
3. Have students run the code multiple times to see that each time it plays the exact number of times as are in the sequence

Part 4 - Identifying Issues with loops #1

1. If we want to create a loop of different length sequences, we can put everything into a live loop and each time through the loop it will choose a new sequence length.

   Have students wrap their code inside of a live loop

   Code Example



```
seq = (ring, 50, 52, 54, 55, 57, 59, 61, 62).shuffle

live_loop :randomLoop do
  use_random_seed Time.now.to_i
  new_seq = seq.take(dice)
  puts new_seq
  puts new_seq.length

  len = new_seq.length
  len.times do
    play new_seq.tick
    sleep 1
  end
end
```

2. Have students run this code and observe the output in the console.

They should look at what is printed in the console as well as the actual notes that are being played.
There is a problem with what we want to happen and what is actually happening.
Give students time to see if they can identify the problem.
Write their prediction in the Peardeck slide

3. If students are struggling to find the issues, provide them with the following hints

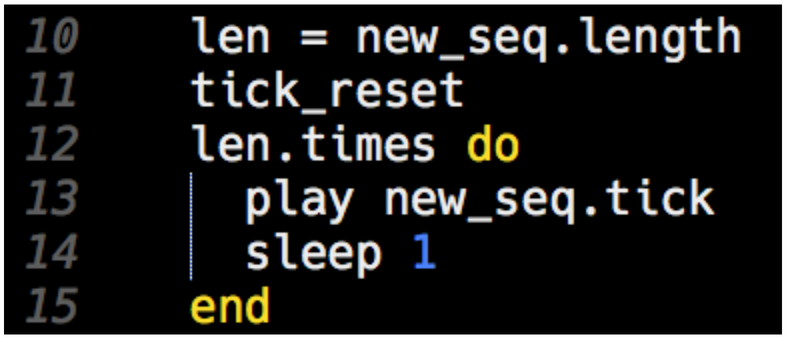
Hints:

- Every time the console prints out the note sequence and sequence length, we are at the beginning of the .times do/end block which means we should be at the beginning of the sequence again.
- Before we added the live loop, every time we ran the code, it started with the same number.

4. **Problem:** The sequence doesn't always start on the first note in the sequence.

5. Because we are using .tick, the count of tick continues to count up and will not always line up with the first note in the ring.

To resolve this, we can just add a ***tick_reset*** before the .times do/end loop
This way the new sequence will always start at the beginning of the sequence



```
    len = new_seq.length
    tick_reset
    len.times do
      play new_seq.tick
      sleep 1
    end
```

Wrap Up/Assessment

Assignment: Create a live loop which generates different lengths of note sequences that uses a ring method to change the length of the sequence

Students should copy and paste code into Peardeck slide

Checklist: Lesson 4 - Data Structure Assignment Checklist Assessment Tool

Data Structure Assignment Checklist

Each part is worth a total of 2 points.
2 - Accurately completes requirement
1 - Requirement is attempted but not completed accurately
0 - Does not include requirement

______ Uses true randomness

______ Uses data structure method to select values

______ Uses method to select random number of values from data structure

______ Stores manipulated data structure in variable

______ Uses loop to iterate through all values of data structure

_______ = Total Score out of 10