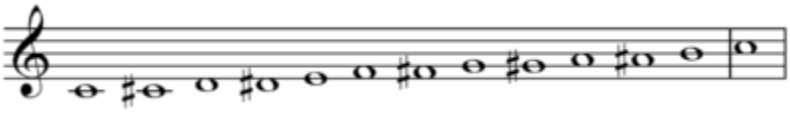
Unit 2: For Loops | **Skill Builder 3: Loop through the musical notes**

In this third lesson for Unit 2, you will learn about the relationship between the frequencies of the musical scale and write a program to play those notes that musicians have used for many centuries.

Objectives:

- Explain the 'twelfth root of two' relationship of the musical scale
- Write a program that plays successive notes in a scale



Teacher Tip: The staff above shows 13 notes. The first and last note are the same note (C) in two different octaves. There are 12 'steps' (spaces between the notes). In music theory these are referred to as half-steps, half-tones, or semitones. This tuning is known as '12-tone equal temperament' and the common ratio of these frequencies is $2^{1/12}$, the twelfth root of 2.

A Little Music Theory

Musical notes are determined by the frequency of a vibrating object such as a speaker, drum head, or string as in a guitar or piano. The notes of the musical scale have a special mathematical relationship. There are 12 steps in an octave. If a note has frequency **F**, then the very next note has frequency **F** $\times \sqrt[12]{2}$.

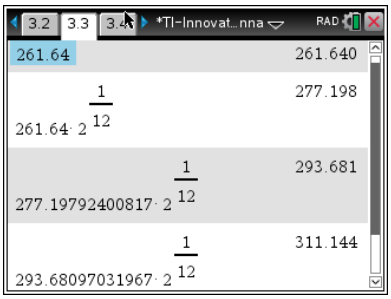
Multiplying a note's frequency by $\sqrt[12]{2}$ or $2^{1/12}$ (the twelfth root of 2) twelve times results in a doubling of the original frequency, so the last note in the octave (or the first note in the next octave) has frequency of $F \times (2^{\frac{1}{12}})^{12} = 2 \times F$. For example, if a note has a frequency of 440 Hz, the note an octave above it is 880 Hz, and the note an octave below is 220 Hz.

The human ear tends to hear both notes an octave apart as being essentially "the same", due to closely related harmonics. For this reason, notes an octave apart are given the same name in the Western system of music notation - the name of a note an octave above C is also C. The intervals between these notes are called 'semitones'.

In this project we'll take advantage of the $2^{1/12}$ principle to generate the 12 notes in an octave.

Middle C has a frequency of 261.64 Hz. An octave above Middle C, also known as Treble C, has a frequency 2×261.64 Hz or 523.28 Hz. There are 12 steps (semitones) between these two notes, and each step is $2^{1/12}$ times the note before it.

In the image to the right, we entered 261.64 into the Calculator app. Then, in the next statement, the value is multiplied it by $2^{1/12}$.



The handheld supplied **Ans** at the beginning (not shown) because the multiplication symbol requires something in front of it. Just pressing **enter** repeatedly creates the sequence of answers shown.

We will incorporate this repetitive principle into our program. If you continue the progression, the twelfth answer will be 523.28, exactly two times the starting value, because $F \times (2^{\frac{1}{12}})^{12} = 2 \times F$.

Teacher Tip: This can get complicated if students are not familiar with powers and roots. Having a piano or keyboard handy to demonstrate the principle on the instrument helps to illustrate the sounds of notes an octave apart and the 12 semitones in an octave (if you include the black keys!).

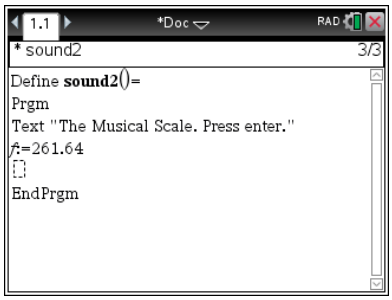
Setting up the Program

1. Start a new program, and name it sound2.
2. Add the **Text** command, add opening and closing quotation marks, and type the text "The Musical Scale. Press enter."
3. Assign the starting frequency, 261.64 to the variable *f*.

There are two assignment operators: ***:=*** *and* → *(sto). You can use either one:*

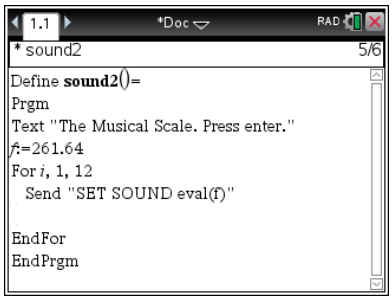
f:=261.64 *or* 261.64→ *f*

4. This variable will represent each of the 12 notes in the scale.



Setting up the For Loop

1. Add a **For** loop that goes from 1 to 12 (for the twelve notes).
2. Add **Send "SET SOUND** statement from the **Hub** menu
3. Add **eval( )** for the variable ***f*** as shown.

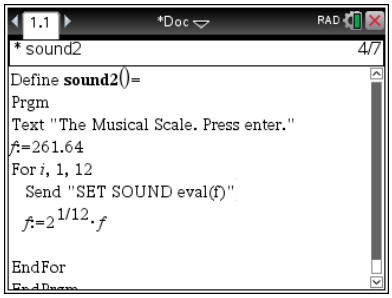


Evaluating the Frequency

1. Multiply ***f*** by $2^{1/12}$, and store the result back into ***f***:

   f := f * 2^(1/12)

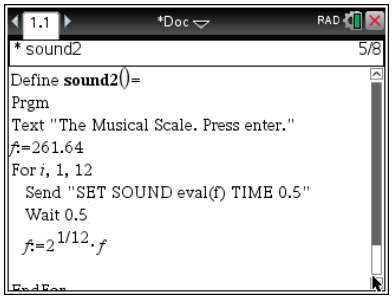
   - This statement takes the current value of ***f*** and changes it to the next higher note's frequency on the scale.
2. Run (**ctrl+R**) the program. Did you hear anything? Is it what you expected?



Modifying the Program

Try adding the **TIME** parameter to the **Send "SET SOUND** command, and be sure to add an equivalent **Wait** statement to the program to let each note finish playing.

If a new command is received by the TI-Innovator Hub before it finishes its last task, then the device will process the new command instead of finishing the current one. Syncing the handheld with the TI-Innovator Hub is up to the programmer.



When you run this program you will not hear the usual 'do-re-mi-fa-sol-la-ti-do' sequence. That's only eight notes. The others correspond to the black keys on a piano. How can you get the program to only play the *correct* eight notes?

Consider adding a **DispAt** statement to show the frequency being played.