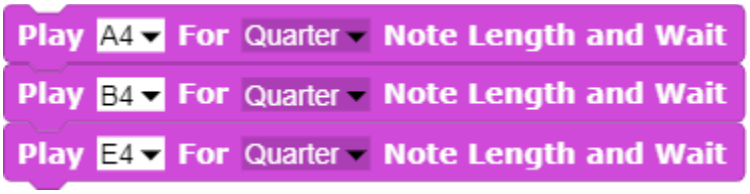
11. A Tune Recorder / Player

Glen Bull, Jo Watts, and Joe Garofalo

Previous modules illustrated how musical notes and chords can be assigned to keys on the computer keyboard. Sequences of notes were created by snapping **Play** code blocks together.



In this module, rather than snapping blocks of code together to create musical sequences, a *TuneScope* recorder will be created to record sequences of notes entered by pressing keys on the computer keyboard.

Electronic keyboards can be purchased that can be used to enter notes that are recorded on the computer.

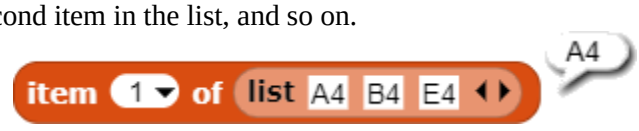
Since a computer keyboard is available on every computer at no additional cost, the computer keyboard will be used as a substitute for an electronic piano keyboard in this module.

Topic 11.1 Playing a List of Notes

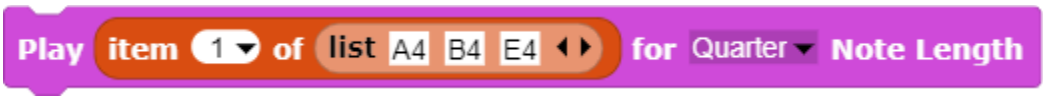
When the computer keys are pressed, musical notes assigned to computer keys will be recorded in a list. A *TuneScope Player* will be used to play the tunes recorded in this manner.

The same type of computer loop used to play chord sequences in previous modules can also be used to play a series of notes. To use a loop to play a sequence of notes, first place the notes in a list in the order in which they will be played. (The **List** code block is found in the *Variables* section of the *Command Palette*.)

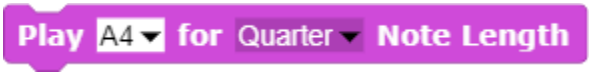
In order to play the notes, a method of accessing individual notes in the list is required. The **Item** code block (also found in the *Variables* section of the *Command Palette*) can be used to access the notes. For example, **Item 1** can access the first item in the list (the note A4), **Item 2** can access the second item in the list, and so on.



Therefore, the code block **Play Item 1 of List**:



Is effectively the same as the code block **Play A4**:



In the next section, lists will be used to play an entire sequence of notes with a single **Play** code block.

Essential Knowledge

CS Principle 3.10 Lists

Application of Essential Knowledge

In the *Drum Machine* module, lists were used to represent drum sequences. In this module, the application is similar. However, rather than using an "X" to represent a drum hit, each item in the list represents a musical note. In the same way that a drum sequencer code block was constructed to play drum patterns, a tune player can be constructed to play lists of notes.

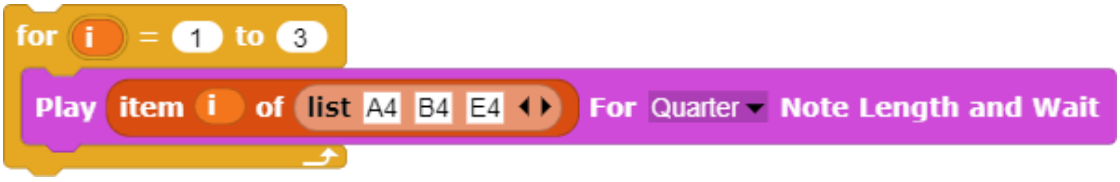
Exploration 11.1 Playing a List of Notes

Create a list of three to six notes. Combine the **Play** and the **Item** code blocks with the list to play one of the notes in the list.

Topic 11.2 Using a Loop to Play a List of Notes

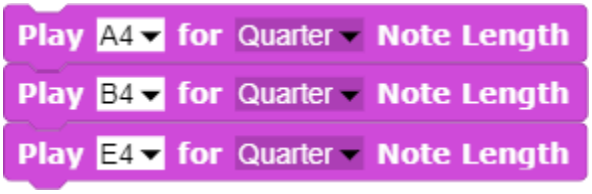
If the list is placed in a loop, a single **Play** code block (shown in purple in the illustration below) can be used to play an entire list of notes. (Note: The **Play** code block is found in the *Sound* section of the *Code Block Palette*.)

The **For** loop has an *index* represented by the letter "i". The value of the index is automatically increased by 1 during each iteration of the **For** loop.



The index " *i* " (depicted in orange in the illustration) has a value of "1" during the first iteration of the loop. This value is increased to "2" during the second iteration of the loop and to "3" during the third (and final) iteration of the loop.

The use of a list in combination with a loop makes it possible to use a single **Play** code block in place of multiple **Play** code blocks snapped together in this manner.



The use of a list to play a sequence of notes is more efficient than using a separate **Play** code block to play each note.

Essential Knowledge

CS Principle 3.10 Lists

Application of Essential Knowledge

In previous modules, a loop has been combined with a list to play sequences of chords and sequences of drum patterns. In this example, the same concept is employed to play lists of notes.

Exploration 11.2 Using a Loop to Play a List of Notes

Use a **For** loop to play the list of notes that you previously created.

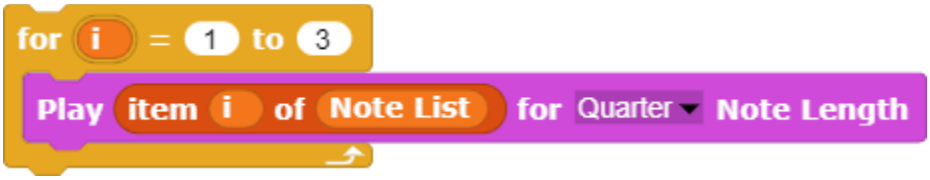
Topic 11.3 Storing a List of Notes in a Variable

A short list of notes can be placed directly into the input slot of the **Play** code block. For a longer list of notes, it can be convenient to assign a name to a list of notes. This can be done with a *variable*. Create a variable named *Note List* and assign the list of notes to it.

(Note: Recall that in previous modules, a checkbox to the left of the variable name under the *Command Palette* was used to display a table with the items in the list that has been assigned to the variable.)

Once the list of notes has been assigned to the variable *Note List*, the oval block displaying the variable name can be dragged into the input slot of the **Item** code block, replacing the actual list. Creation of the variable *Note List* will make it easier to add additional items to the list of notes in the future.

The variable *Note List* can be used in place of the actual list of notes. When the list is dozens of notes or even hundreds of notes long, use of a variable to represent the list of notes makes the code more readable.



Use of meaningful variable names is crucial to effective programming. A meaningful variable name makes the code more readable.

Essential Knowledge

- **CS Principle 3.1** Variables and Assignments
- **CS Principle 3.10** Lists

Application of Essential Knowledge

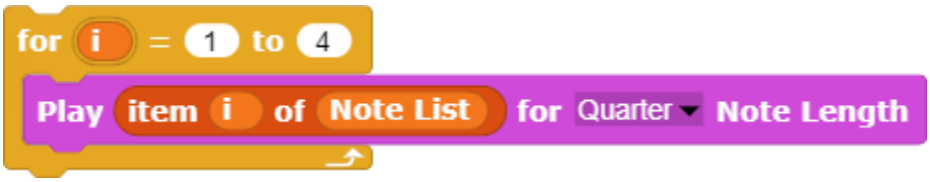
Replacement of the list of notes with the variable name Note List makes the code easier to read and understand. The code block would become increasingly unwieldy as the length of the note list increased (to encompass an entire song, for example). Use of the variable name Note List addresses this potential issue.

Exploration 11.3 Storing a List of Notes in a Variable

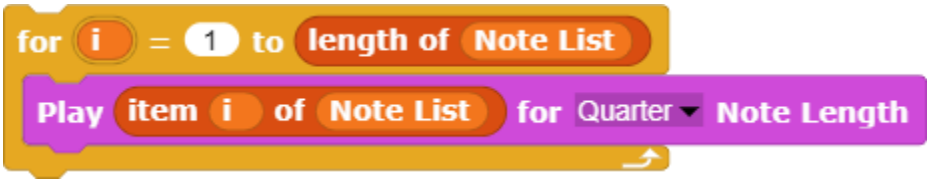
Create a variable and assign the previously created list of notes to the variable. Then use the variable in combination with a loop to play the list of notes.

Topic 11.4 Playing Lists of Varying Lengths

The number of notes in *Note List* changes as new notes are added to create and play a tune. When the number of notes in the list changes, the For loop must also be revised. For example, if a fourth note is added to the list, the code block **For I = 1 to 3** must be replaced with **For I = 1 to 4**.



Use of the **Length of List** block in place of a specific number such as “3” or “4” ensures that the loop will continue to work properly even as the number of notes changes.



Essential Knowledge

CS Principle 3.10 Lists

Application of Essential Knowledge

The **Length of List** code block was previously used in a similar manner to play lists of chords of varying length.

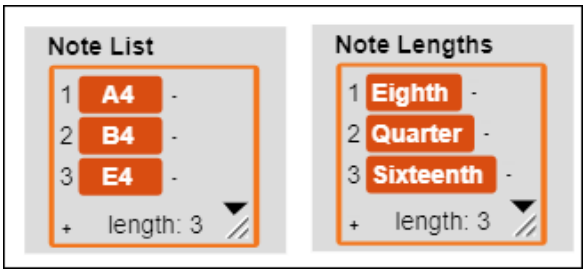
Exploration 11.4 Playing Lists of Varying Lengths

Continue to add additional notes to *Note List* to develop a musical tune. Use the updated loop with the **Length of List** block to play the revised list of notes.

Topic 11.5 Playing Lists with Notes of Varying Durations

All of the notes in previous examples have been the same duration – a quarter note. However, a second variable, *Note Lengths*, can be created and used to specify the duration of each note. (This variable could have also been named *Note Duration* because that also would have been a meaningful name.)

At this point, two lists have been created – a list of notes and a parallel list of note durations.



The **Item** code block is used to access the note length in the **Play** code block in a manner similar to the way in which the **Item** block is used to access the notes in the list of notes.

Exploration 11.5 Playing Lists with Notes of Varying Durations

Create a parallel list of note lenghts to be used in combination with the previously created list of notes. Then play the revised list of notes.

Topic 11.6 Creating a Tune Player

The previously developed code can now be used as the basis for a custom code block, a **Tune Player**.

The **Tune Player** code block plays the notes in *Note List* using the note lengths stored in the variable *Note Lengths*.

Creation of a custom code block, **Tune Player**, means that it is no longer necessary for the programmer to keep track of the specific details of the loop that is used to play the notes.

Essential Knowledge

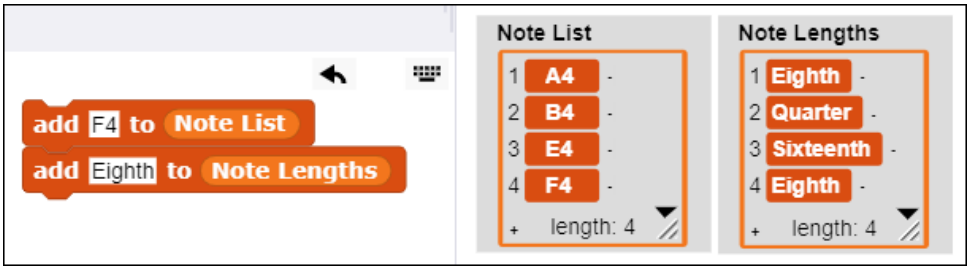
CS Principle 3.13 Developing Procedures

Exploration 11.6 Creating a Tune Player

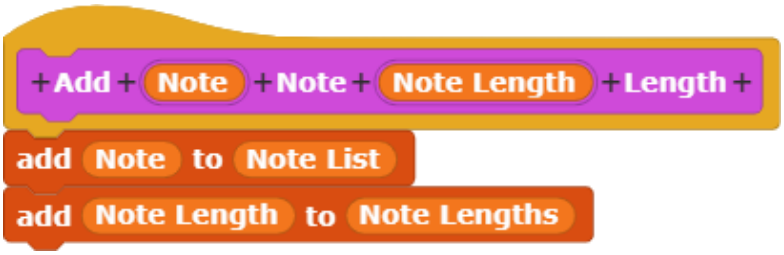
Use the custom **Tune Player** code block to play the previously created list of notes.

Topic 11.7 Adding Notes to the Note List

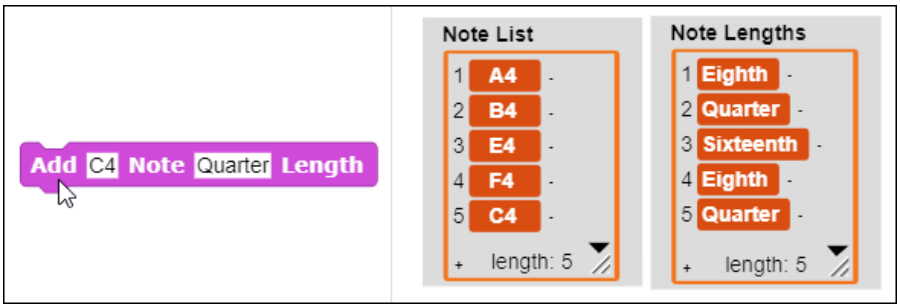
The **Add Thing to List** code block (found in the *Variables* section of the *Command Palette*) can be used to add notes and note lengths to the note lists.



Once these code blocks have been tested and verified to work as anticipated, they can be used to construct an **Add Note** custom code block.



This custom block adds a note with a specified length to the *Note List* and *Note Lengths* lists respectively.



Essential Knowledge

- CS Principle 3.10 Lists
- CS Principle 3.13 Developing Procedures

Application of Essential Knowledge

The custom **Add Note** code block extends the flexibility of the tune layer by providing a convenient way to add notes (and corresponding durations) to the *Note List* and the *Note Lengths* lists respectively.

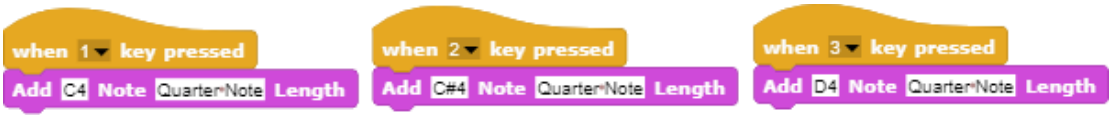
Exploration 11.7 Creating a Tune Player

Use the custom **Tune Player** code block to play the previously created list of notes.

Topic 11.8 Recording Notes Entered via the Computer Keyboard

The **When Key Pressed** block assigns an action to a key on the computer keyboard. In this instance, the action adds a note to the *Note List*. A piano keyboard template (shown in the illustration below) can be used to identify keys on the computer keyboard that correspond to musical notes. Note: Controlling data access in this way reduces the possibility of a typing error.

In this piano keyboard template, the number key "1" corresponds to the musical note "C" on the piano keyboard, the number key "2" corresponds to the musical note "C# on the piano keyboard, the number key "3" corresponds to the musical note "D" on the computer keyboard, and so forth.



The assignment of musical notes to keys on the computer keyboard makes it possible to add new notes to the list by pressing keys on the computer keyboard. There are other possible layouts that could be created depending on the individual needs of each user. For example, the musical note "C" could be assigned to the letter "C" on the computer keyboard. The ability to customize code blocks gives users complete flexibility to customize code blocks in ways that reflect individual preferences.

Exploration 11.8 Recording Notes Entered via the Computer Keyboard

Use card stock to create a *Piano Key* template and use it as an overlay for your computer's keyboard. Then create code in the manner described above that records musical notes entered on the computer keyboard and enters them into *Note List*.

Topic 11.9 Varying Octaves

An effective computational thinking strategy is to solve the simplest instance of a more complex problem as a "proof of concept." In this instance, all of the notes are assigned to the fourth octave. Now that a basic method for recording notes has been developed as a proof of concept, the note recorder can be refined by adding the capability to vary the octave of recorded notes.

The octave of notes can be varied by creating a variable named *Octave*. The variable can initially be set to a default value of "4".

The following code blocks enable the octave to be increased or decreased by pressing the up and down arrows on the computer keyboard

With this refinement of the code, pressing a key selects the musical note. For example, pressing the "1" key on the computer keyboard, supplies the note "A" as an input to the **Add Note** procedure. The **Add Note** code block then uses the **Join** code block (found in the *Operators* section of the *Command Palette*) to combine the note name with the octave. Therefore, if the value of the variable *Octave* is "4", pressing the "1" key on the computer keyboard will add the note "C4" to the note list.

A piano keyboard has seven octaves. A computer is not constrained by the physical layout of a piano keyboard. Therefore, the number of octaves can be set to the range desired by the user. The refinement of the code shown below resets the variable *Octave* to "1" when its value exceeds "7", and resets the variable to "7" when its value is less than "1". This constrains the numerical value of the variable *Octave* to the numbers one through seven.

Essential Knowledge

- CS Principle 3.1 Variables and Assignments
- CS Principle 3.3 Mathematical Expressions
- CS Principle 3.6 Conditionals

Application of Essential Knowledge

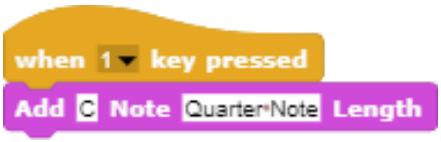
In the preceding example, the new code accesses a variable that is external to the code block. As programs become more complex, it can be necessary to document these kinds of details as the program grows.

Exploration 11.9 Shifting Octaves

Recreate your previously created list of notes, but shift all of the notes up or down an octave. Use the custom **Tune Player** code block to play the new list of notes and listen for the differences between the old and the new lists.

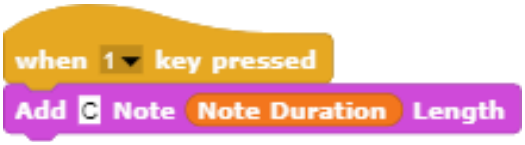
Topic 11.10 Varying the Duration of Notes

A method similar to that used to vary the octave of notes can also be used to vary the duration of notes. Until now, all of the notes have been assigned a fixed duration of one quarter note.



That limitation can be addressed by creating a new variable, *Note Duration*. The *Note Duration* variable can be adjusted using the **Set Variable** code block in the same manner as previous examples.

Creating a variable for the note duration enables this parameter to be adjusted so that the notes are not always the same length.



Once the *Note Duration* variable has been created and incorporated into the key press code as an input to note length, computer keys can be used to control this variable in a manner similar to the way in which the up and down arrow keys were used to control the value of the *Octave* variable.

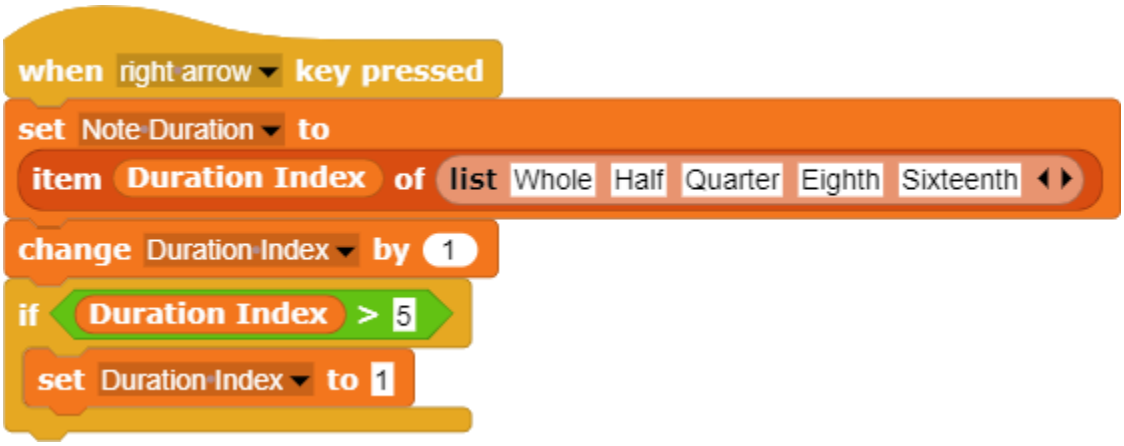
A list of the note lengths is accessed to select the duration of notes. A variable named *Duration Index* will be used to determine which item in this list is selected.

The *Duration Index* is initially set to "1" (pointing to the first item in the list, *Whole* note).

The left and right arrow keys will be used to adjust the duration selected. When the right arrow key is pressed, the *Duration Index* is used to select the corresponding item in the list of note durations.

The *Duration Index* is advanced each time the arrow key is pressed. Therefore, the next time the arrow key is pressed, the *Note Duration* will be set to the next item in the list (a *Half* note, in this instance).

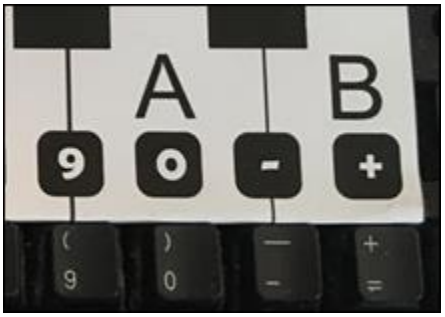
Eventually the end of the list will be reached. When this occurs, the *Duration Index* is reset to "1" so that it points to the beginning of the list again.



This enables the duration of each note to be adjusted by pressing the right arrow key. A parallel left arrow procedure enables the left arrow to decrease the *Duration Index.* Consequently, the *Note Duration* can be quickly adjusted by moving back and forth through the list of note durations using the arrow keys.

The Case of the Equal (“=”) Key

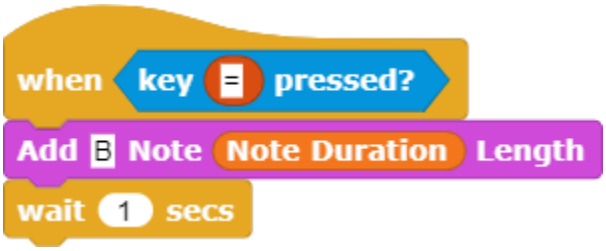
The **When __ Key Pressed** code block includes the minus (“-”) and plus (“+”) keys in its drop-down menu but not the equal (“=”) key.



There are several possible work-arounds. Since the equal (“=”) key is not available in the drop-down menu of the **When __ Key Pressed** code block, the plus (“+”) key (directly above the equal key) could be substituted instead.

One drawback of this work-around is that the *Shift* key must be held down to access the plus (“+”) key. Some users may not recognize or remember that the shift must be held down to access the plus (“+”) key. It is desirable to design a user interface that does not require users to make these kinds of adjustments.

While the code is slightly more complex, the following method can be used to access the equal “=” key directly. The **Key Pressed?** code block (found under the *Sensing* section of the *Code Block Palette*) can be placed in a **When < >** code block. The orange *Identity* code block (shown as an oval within the blue **Key Pressed?** Block) is then used to insert an equal (“=”) sign into the **Key __ Pressed** code block. (Note: The orange Identity code block can be obtained from the Big Num library of the Snap! code library extensions.)



The resulting code will insert the note “B” repeatedly as long as the equal (“=”) key is held down. To avoid instances of multiple notes being unintentionally added to the *Note List*, a **Wait** code block can be added at the end. If a **Wait** code block is included in the code, only a single note will be added to the note list as long as the user does not hold down the key for more than one second. (A longer wait can be added if necessary.)

Essential Knowledge

CS Principle 3.1 Variables and Assignments

Application of Essential Knowledge

The example in the preceding section continues to extend the way in which variables are used and applied. In this instance, two additional variables – *Note Duration* and *Duration Index* – are used to enable the user to vary the durations of notes recorded.

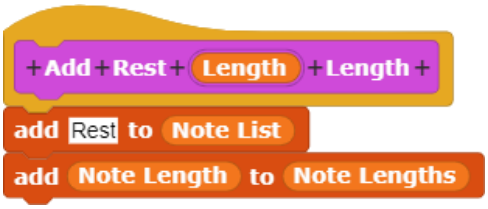
Exploration 11.10 Changing Note Lengths

Use the blocks in this chapter to create a *Note List* and a *Note Lengths* list. Use at least three different note lengths. Use the custom **Tune Player** code block to play the new list of notes and listen for the differences between the note lengths.

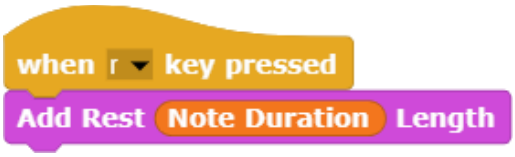
Topic 11.11 Incorporating a Rest

Music often includes times during which no notes are played. The musical term for this is a *rest*. The **Rest** code block incorporates a delay of a specified duration when no notes are played.

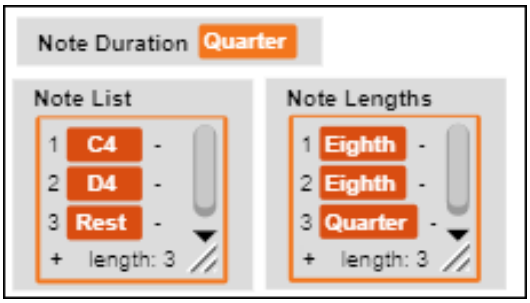
The **Add Rest** code block is similar to the **Add Note** code block except that it adds a rest instead of a note to the note list. A corresponding duration (Half, Quarter, Eighth, etc.) is added to the *Note Lengths* list.



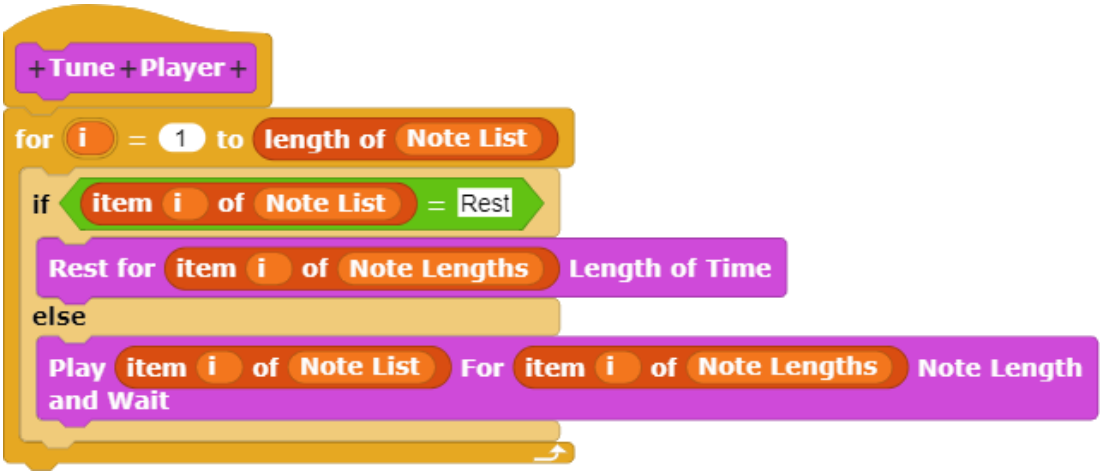
The **Add Rest** code block could be assigned to any key on the computer keyboard. In this instance, it has been assigned to the "r" (*for rest*) key.



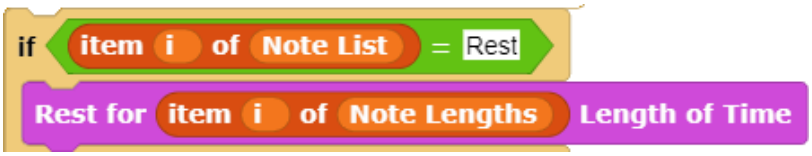
Pressing the “r” key on the computer keyboard will then insert the word “Rest” into the note list.



After the capability for inserting a rest into the *Note List* is implemented, the **Tune Player** needs to be told how to process a rest when it encounters one.



This is accomplished through incorporation of an **If … Else** code block. This code block says to **Rest** for a specified amount of time if the word “Rest” is encountered in the *Note List*.



Otherwise, the specified note in the *Note List* is played in the same fashion as before.

Essential Knowledge

- CS Principle 3.1 Variables and Assignments
- CS Principle 3.6 Conditionals
- CS Principle 3.10 Lists
- CS Principle 3.13 Developing Procedures

Exploration 11.11 Adding Rests

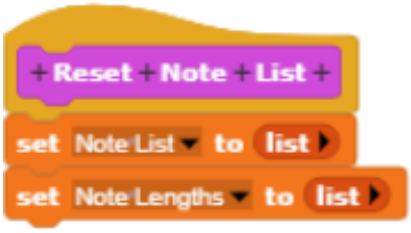
Update the Tune Player so that rests can be incorporated into the lists of notes.

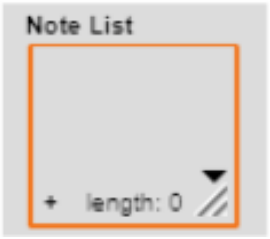
Topic 11.12 Customizing the Player

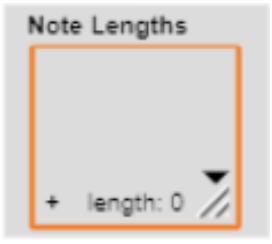
TuneScope enables music tools to be customized. The methods used to create the *Tune Recorder / Player* can also be used to customize it For example, the space bar could be used to start the *Tune Player*.

A code block to reset the note list might also be useful.

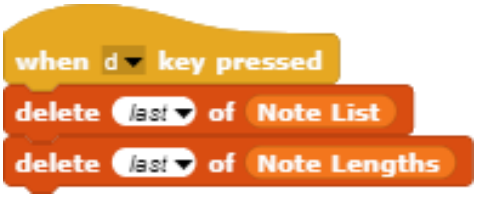
This code block sets both the *Note List* and the *Note Lengths* list to empty lists.



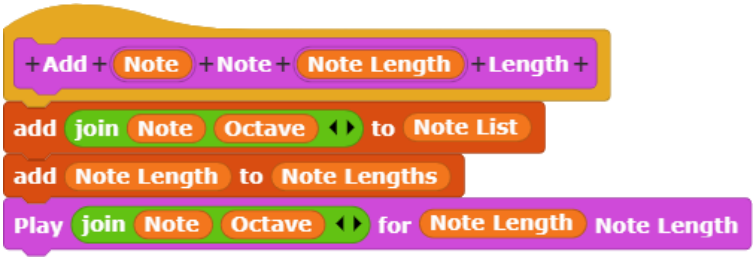




To correct errors, the ability to delete a note from the *Note List* and the *Note Lengths* list without completely resetting them would also be useful. This provides the capability to edit the list of notes by removing notes and trying different ones. In this example, when the “d” (for “Delete”) key is pressed on the computer keyboard, the last note in the list is deleted from *Note List*. The last note length in the *Note Lengths* list is also deleted.



When a note is added to the note list, it could be helpful if the *Tune Recorder* also played that note. Adding a **Play** code block at the end of the **Add Note** procedure provides instant feedback when a note is added.



Other enhancements can be incorporated to customize the code blocks and interface to match your individual preferences.

Essential Knowledge

- CS Principle 3.10 Lists
- CS Principle 3.13 Developing Procedure

Exploration 11.11 Customize the Player

Create any of the blocks in this chapter and add them to your Tune Player. Try to think of other features that might make the Tune Player easier to use and how you might create them.

Further Explorations

1. Use *Tune Recorder* to record a short series of musical notes (i.e., a half dozen to a dozen notes). You can either (1) compose your own sequence of notes through experimentation and exploration or (2) copy an existing series of notes from an existing tune. If you choose the second option, be sure to credit the source.
2. Write the lyrics for a two-line jingle to accompany the recorded musical sequence.
3. Use the *Tune Player* to play the recorded sequence of notes.
4. Sing or chant the lyrics of the jingle that you created while playing the sequence of notes.
5. Create a video of your performance.

Extensions

- What are the challenges of using the *Tune Player* / *Recorder* to create music?
- Are there additional features that you think would be useful?
- Do you feel that you know enough to create new features using *TuneScope* code blocks?