



Unit 1: Getting Started with the TI-Innovator™ Hub

Skill Builder 3: Input and Sound

In this lesson, you will learn about two ways to make sound on the TI-Innovator Hub and use variables to store and retrieve values.

Objectives:

- Produce a sound on the TI-Innovator Hub
- Use variables to store entered values
- Use the **input()** function and the **int()** function
- Play musical notes

The TI-Innovator Hub houses a small speaker, called 'SOUND'. The speaker is on the bottom of the hub. There is no amplifier, so the volume is very low. It is designed that way so that your classroom does not get too noisy.



1. Start a new Python file using the **Hub Project** template. Ours is named SOUNDA since SOUND is already taken.

```

EDITOR: SOUNDA
PROGRAM LINE 0004
# Hub Project
from ti_system import *
from time import *
-
  
```

Teacher Tip: Filenames like LIGHT, COLOR and SOUND are 'taken' by TI-developed modules. They are hidden, protected files that control the TI-Innovator Hub devices.

Python is case sensitive: x and X are two different variables. All keywords are lowercase. Using a capital letter in a keyword like 'If' results in an error.



10 Minutes of Code – Python

TI-84 PLUS CE PYTHON WITH THE TI-INNOVATOR™ HUB

- 2. You will write two **input** statements to get values to create a sound. These values will be stored in two variables, **f** and **t**. Write two statements like this:

```
f = input("Frequency? ")
t = input("Time? ")
```

We will do this step-by-step next...

- 3. Type the lowercase letter **f** (press **[alpha] [F]**) and the **=** sign on a blank line below the bottom import statement. Use the **[sto>]** key for the **=** sign.

```
f =
```

Leave your cursor to the right of the **=** sign for the next step.



```
EDITOR: SOUNDA
PROGRAM LINE 0004
# Hub Project
from ti_system import *
from time import *
f=_
```

Teacher Tip: We use the variable name **'f'** for brevity. Multi-character names (like 'frequency') are allowed and recommended but require more keystrokes. Uppercase variables are different from lowercase variables: **'F'** and **'f'** are two DIFFERENT variable names.

- 4. Now get the **input()** function by selecting it from **<Fns...> I/O**

```
f = input( )
```

```
EDITOR: SOUNDA
Func Ctl Ops List Type I/O Modul
1:print()
2()
3:eval()
4:str.format() string format
```



5. Selecting **input()** from the menu pastes the word **input()** into your program after the = sign.

Inside the parentheses, type the “prompt” for this input statement in quotation marks.

To type the prompt “frequency? “, turn on alpha-lock by pressing [2nd] [alpha].

The quotation mark is on the [+] key. The question mark (?) is on the [-] (negation) key next to the [enter] key. The space character is on the [0] key. Be sure to use the closing quote as well, the same as the opening quote.

```

EDITOR: SOUND A
PROGRAM LINE 0004
# Hub Project
from ti_system import *
from time import *
f=input("frequency? ")

```

On the next line make another, similar **input** statement for the variable **t** which stands for *time*.

Teacher Tip: Python allows the use of either single quotes or double quotes. Double quote is easier to find on the keypad. Mixing single and double quotes is not permitted. Single quote is [2nd] [+].

6. Below the two **input** statements, add another **import** statement to access the sound functions on the TI-Innovator Hub.

To paste the **import** statement:

Press [math] and select **ti_hub... Hub Built-in devices... Sound...**

To get the **sound.tone(,)** function:

Press [math] again and select the new **Sound...** menu item at the bottom of the list to get the **tone(freq, time)** function.

```

EDITOR: SOUND A
PROGRAM LINE 0007
# Hub Project
from ti_system import *
from time import *
f=input("frequency? ")
t=input("time? ")
import sound
sound.tone(, )

```

Tip: you can press [up-arrow] to quickly get to the bottom of any menu. immediately.

Teacher Tip: Normally, **import** statements appear at the top of a program as a matter of proper style and clarity for reading but **import** statements can appear anywhere in the code. The effect of this particular example shows up when the program is run: The two input statements will be executed, but if there is an issue with the TI-Innovator Hub (not attached, for example) then the **import sound** statement causes an error. If the **import sound** statement were at the top of the code with the others then the error message will appear *before* the input statements are processed and execution halts right away.

The **import** statements for the TI-Innovator devices do some processing of their own. They are not passive instructions. Upon execution of the **import sound** (and other devices) statement there is active code ‘under the hood’ that makes sure that the TI-Innovator Hub is attached and ready to go.



- The two arguments for the tone function are the two variables, **f** and **t** which have been assigned values using the **input** functions.

Since these are just single letters you can press **[alpha]** and the proper letter key for each. Be sure **f** is before the comma and **t** is after the comma inside the parentheses as shown.

```

EDITOR: SOUNDA
PROGRAM LINE 0007
# Hub Project
from ti_system import *
from time import *
f=input("frequency? ")
t=input("time? ")
import sound
sound.tone(f,t)_

```

- <Run>** the program. At the prompts, enter a frequency of 440 and enter 2 for the time.

And then... oops! A 'runtime error'! This error occurs because the **input** functions return values that the sound function cannot use: they are strings, **"440"** and **"2"**, rather than numeric values, **440** and **2**. You need to convert each string value to a numeric value using either **float()** (for a decimal number) or **int()** (for an integer). We will correct this error next. But first...

```

PYTHON SHELL
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "SOUNDA.py", line 7, in <module>
  File "sound.py", line 33, in tone
TypeError: can't convert 'float' object to str implicitly
>>> |

```

- Programming errors come in three flavors:

Syntax errors - errors in typing (spelling/grammar/structure).

Runtime errors – errors during program execution, such as division by zero, incorrect data types, or Hub not present.

Programmer errors – errors in logic or expressions that cause the program to produce incorrect results. These are the hardest errors to correct because they happen in the programmer's head.

ERROR

- To convert the values of **f** and **t** to integers use the **int()** function found on the **<Fns...>** **Type** menu. Insert the two statements:

```

f = int( f )
t = int( t )

```

As shown in the image, these two statements belong *after* the **input** statements and before the **sound.tone** statement. Order matters!

You can also use **float()** in place of **int()** if you expect to enter decimal values.

*Note that the **import sound** statement has been moved up to be with the other import statements. This is more common in Python coding.*

```

EDITOR: SOUNDA
PROGRAM LINE 0008
# Hub Project
from ti_system import *
from time import *
import sound
f=input("frequency? ")
t=input("time? ")
f=int(f)
t=int(t)
sound.tone(f,t)

```



Teacher Tip: Python has very loose ‘type’ rules. You do not need to declare a variable’s type before it is assigned a value . A variable can contain any data type like integer, float, string, or others and a variable can change type ‘on the fly’ as in the above example where strings are converted to integers.

11. <Run> the program again and enter 440 for frequency and 2 for time again. You will hear a tone of 440Hz for 2 seconds from the speaker on the bottom of the TI-Innovator Hub. The Shell prompt is displayed on the calculator screen since the program is done. Select <Tools> and press [enter] to **Rerun Last Program** to try other frequencies (between 0 and 8000 Hz) and times. *Can you hear all the frequencies?* Try a ‘low’ frequency value like 3 or 6. What do you hear?

```

PYTHON SHELL
>>> # Shell Reinitialized
>>> # Running SOUNDA
>>> from SOUNDA import *
frequency? 440
time? 2
>>> |
  
```

Note: If you do not see the shell prompt >>> after a few seconds, then press the [on] key to ‘break’ the program.

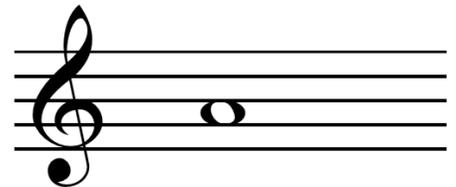
Teacher Tip: Very low frequencies (less than 10?) will cause a buzzing sound from the speaker because it is vibrating very slowly. As frequency increases the tone increases in pitch. At very high frequencies no sound is heard because the human ear cannot process those frequencies (but dogs can). There is an upper limit of 8000Hz built into the software.

12. Playing Notes...

sound can also play musical notes using their names:

sound.note(“A4”,2)

plays the note A in the 4th octave.



13. To copy the SOUNDA program, from the Editor or the Shell...

- select <Files>, and, in the File Manager screen,
- point to the SOUNDA filename,
- select <Manage> Replicate Program...
- name the copy SOUNDDB and press [enter] or select <Ok>.

You are taken immediately into the Editor with the code for SOUNDDB showing.

```

FILE MANAGER
REPLICATE PROGRAM
Current Name:SOUNDA
Name=SOUNDB

Allowed
- Up to 8 characters
- First character:A-Z
- Remaining characters:A-Z 0-9 _
  
```



14. Make three changes in the SOUNDB program as shown:
- Change “frequency?” to “note?” in the `f = input()` statement.
 - #comment the `f=int(f)` statement by placing the cursor at the beginning of that line and typing `[2nd] [3]` for the # sign because the note function needs its first argument to be a string, not an integer.
 - Change `sound.tone` to `sound.note` by deleting ‘tone’ and typing ‘note’ in its place (without quotes).

Note: You can delete the line `f=int(f)`, too, but it’s good practice to use a comment in case you may need the statement in the future.

```

EDITOR: SOUNDB
PROGRAM LINE 0009
# Hub Project
from ti_system import *
from time import *
import sound
f=input("note? ")
t=input("time? ")
#f=int(f)
t=int(t)
sound.note(f,t)

```

15. After making all three changes, <Run> the program. At the **note?** prompt type **a4** without quotes as shown. Enter **2** for the time again.

Do you hear the same sound as before?

A4 is the note A in the 4th octave (A above middle C) and has a frequency of 440Hz. Try other notes (letters A,B,C,D,E,F,G) (uppercase or lowercase) in octaves (1,2,3,4,5,6,7) Remember, just Select <Tools> and press [enter] to Rerun Last Program to try other frequencies

```

PYTHON SHELL
>>> # Shell Reinitialized
>>> # Running SOUNDB
>>> from SOUNDB import *
note? a4
time? 2
>>> |

```

There are two other functions on the **Sound...** menu that include a ‘tempo’ argument. Give them a try and figure out what ‘tempo’ means.

Teacher Tip: ‘tempo’ is the sound equivalent of ‘blink’ for light and color. A tempo value of 3 will make the speaker ‘beep’ 3 times a second for time seconds. Example:

```
sound.tone(f, t, 5)
```

will ‘pulse’ the speaker 5 times a second for **t** seconds.

Note A8 causes an Innovator error (a beep and a red-light blink), not a programming error.

```
sound.note("a8", 3) generates an error inside the Innovator hub.
```