



In this lesson you will generate random computer music.

Objectives:

- Use the **for** loop to control the number of notes
- Use the random number generator to create random musical notes

In this unit you used **for** loops to control light, colors, and sounds. In this application, you will create a program to play computer-generated random sound or music. This challenge can take three approaches: a) play purely random tones (frequencies), b) play random notes using their special frequencies, or c) play random notes using their names (in a list). You will also use random durations (timings) for each tone/note. And, for the icing on the cake, each note can create a different color using the color LED.

Teacher Tip: If this is a class project, expect lots of different approaches.

The instruction to students results in random frequencies or tones. Notes have special frequencies and names. See the end Teacher Tip for more 'musical' effects.

1. Make a new Python Hub Project (this one is named MUSICB) and import **color** and **sound** from the [math] ti_hub... menu.

You will also need a function that can produce 'random' numbers. These functions are part of the standard Python commands but are found in a separate module that the Hub Project template does not import.

Press [math] **random...** and add the statement

from random import *

to your collection of **import** statements at the top of your code.

```

EDITOR: MUSICB
PROGRAM LINE 0007
# Hub Project
from ti_system import *
from time import *
import color
import sound
from random import *
-
Fns... a A # Tools Run Files

```

2. Write an **input** statement to enter the number of sounds to play (**n**).

Convert the value of **n** to an integer.

Use **n** in a **for** loop to play the random sounds.

```

EDITOR: MUSICB
PROGRAM LINE 0011
# Hub Project
from ti_system import *
from time import *
import color
import sound
from random import *

n=input("number of notes?")
n=int(n)
for i in range(n):
+-
-
Fns... a A # Tools Run Files

```



- Use the **randint()** function found in **[math] random...**

r = randint(,)

The variable **r** is assigned a *random integer* from *min* to *max* for later use. *min* and *max* will be replaced with numbers. But, before you enter those numbers, consider the next step...

- r** represents a sound frequency. Not all frequencies are 'audible'. Very small frequencies and very large frequencies should be avoided because they are outside our hearing range.

Recall that when working with music in the previous lesson you used frequencies in the hundreds, so when choosing min and max keep that in mind.

Add another random variable **t** (for time) and use the **uniform()** random number generator also found on **$random...$**. This returns a random *decimal* number between *min* and *max* so some notes will play for part of a second. Your choice of *min* and *max* here will depend on how long you want each note to last.

```

EDITOR: MUSICB
random module
random
1:from random import *
2:random()
3:uniform(min,max)
4:randint(min,max)
5:choice(sequence)
6:randrange(start,stop,step)
7:seed()
Esc

```

```

EDITOR: MUSICB
PROGRAM LINE 0014
import color
import sound
from random import *

n=input("number of notes?")
n=int(n)
for i in range(n):
**
**r=randint(,)
**t=uniform(,)
**
**
Fns... a A # Tools Run Files

```

Teacher Tip: the value of **r** could be **uniform(,)** also.

- Next make the sound using **sound.tone()** and enter the variables representing frequency and time, the variables **r** and **t** respectively.

Add the **sleep()** function to pause the program while the tone is playing. *For how long should the program wait?*

```

EDITOR: MUSICB
PROGRAM LINE 0016
n=input("number of notes?")
n=int(n)
for i in range(n):
**
**r=randint(,)
**t=uniform(,)
**sound.tone(,)
**
**
**sleep()
**
Fns... a A # Tools Run Files

```

Teacher Tip: **sound.tone(r , t)**

sleep(t)



6. How about using the color LED? Remember that there are three color channels, red, green, and blue, that are limited to the values 0 to 255. You can have the LED light up in purely random colors using the `randint(,)` function for each channel. For example:

`red = randint(0 , 255)`

or you can make the colors depend on the frequency `r` or the time `t` or both. But be careful about going 'out of range' beyond 255.

The screen to the right is *not* the complete program! You need to fill in the proper elements in each function.

```

EDITOR: MUSICB
PROGRAM LINE 0018
for i in range(n):
**
**r=randint(,)
**t=uniform(,)
**sound.tone(,)
**
**
**red=randint(0,255)
**color.rgb(red,,)a
**sleep()
**
Fns... | a A # | Tools | Run | Files

```

Teacher Tip: Possible solution for color:

Using `%` (mod) guarantees a value in the range 0...255 for the color channels.

`color.rgb(r*t%256,r*t%256,r*t%256)`

which produces various brightness of a white light.

Challenge: How about random 'notes' for music rather than just tones?

- a) Using frequency:

```

n=randint(0,59)
freq=55*2**(n/12)
sound.tone(freq,time)

```

- b) Using note names:

```

Make a list of note names*: NoteList=["c1","d1","e1"...]
R=randint(0,len(NoteList)-1)
sound.note(notelist[R], time)

```

Possible solution:

```

# Hub Project
from ti_system import *
from time import *
import color
import sound
from random import *

n=input("number of notes?")
for i in range(n):
    r=randint( 50, 1000)
    t=uniform( .2, 1)
    sound.tone(r,t)
    red=randint( 0, 255)
    green=randint( 0, 255)
    blue=randint( 0, 255)
    color.rgb(red, green, blue)
    sleep(t)

```



10 Minutes of Code – Python

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

UNIT 2: APPLICATION

TEACHER NOTES

* Creating a long list of note names is more easily done on a computer: if the program is started on the calculator, use **TI-Connect CE™** to transfer the program to the computer as a .py file. Open the .py file in a Python editor or any text editor. Save the .py file and transfer it back to the calculator using **TI-Connect CE**.