



Unit 2: for loops with the TI-Innovator™ Hub

Application: Computer Music

In this application, you will learn to 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.

1. Make a new Python Hub Project.

You will need a function that can deliver random numbers. This tool is part of the standard Python commands, but it is found in a separate module that the Hub Project template does not import.

Press **menu > Random** and add the statement

from random import *

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

```

1.7 1.8 1.9 ▶*Unit 2 Py...ops RAD [X]
*u2app.py 9/19
# Unit 2 Application
#=====
from ti_hub import *
from math import *
from time import sleep
from tiplotlib import text_at,cls
from ti_system import get_key

from random import *
#=====

```

2. Use the **esc** key to terminate the program.

Press **menu > TI Hub > Commands** and select the statement

while get_key() != "esc":

Be sure to indent the rest of the statements.

When you run this program press the **esc** key to end the program.

```

1.7 1.8 1.9 ▶*Unit 2 Py...ops RAD [X]
*u2app.py 13/21
from ti_hub import *
from math import *
from time import sleep
from tiplotlib import text_at,cls
from ti_system import get_key

from random import *
#=====
while get_key() != "esc":
  <<block

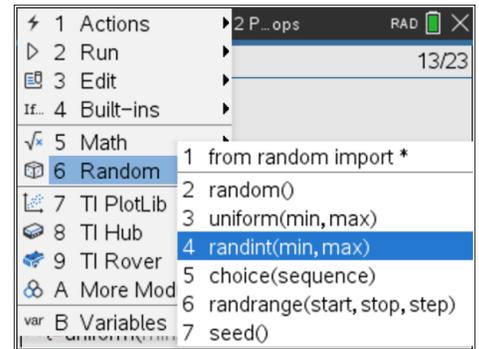
```



- Use the **randint()** function found in **menu > Random**:

r = randint(min, max)

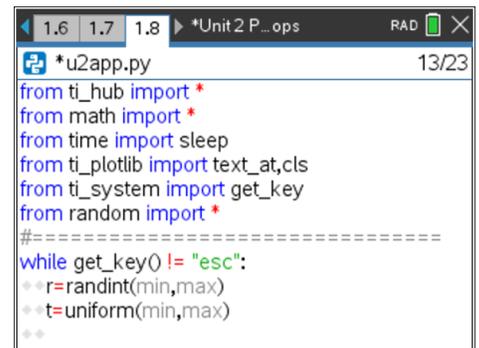
The variable **r** stores a *random integer* for later use. *min* and *max* will be replaced with numbers. But, before you enter those numbers, consider the next step.



- 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.

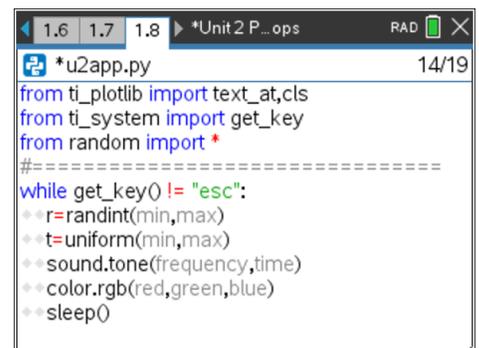
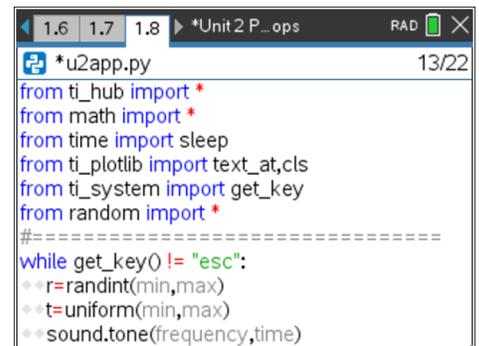
Recall that, when working with music in the previous lesson, you used frequencies in the hundreds, so when choosing *min* and *max* keep that in mind.

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



- Next create the sound using **sound.tone()** and replace the prompts frequency and time with your variables **r** and **t**, respectively.

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





10 Minutes of Code - Python

TI-NSPIRE™ CX II WITH THE TI-INNOVATOR™ HUB

UNIT 2: APPLICATION

STUDENT ACTIVITY

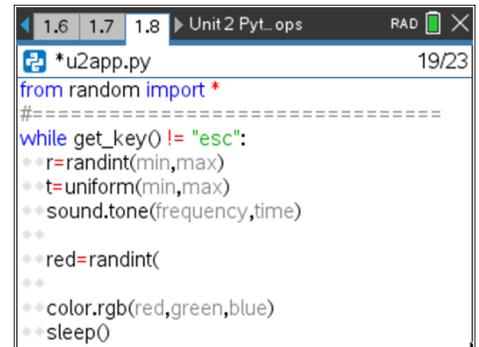
6. How about the color LED? Remember that there are three color channels, red, green, and blue, that are limited to the values 0 to 255. You can have the LED light up in purely random colors using the `randint(0,255)` function for each channel

```
red = randint(0,255)
```

or you can make the color depend on the frequency `r` or the time `t` or both.

Be careful about going 'out of range': beyond 255.

The screen to the right is *not* the complete program!



```
Unit 2 Pyt... ops 19/23
*u2app.py
from random import *
#=====
while get_key() != "esc":
    r=randint(min,max)
    t=uniform(min,max)
    sound.tone(frequency,time)
    red=randint(
    color.rgb(red,green,blue)
    sleep()
```