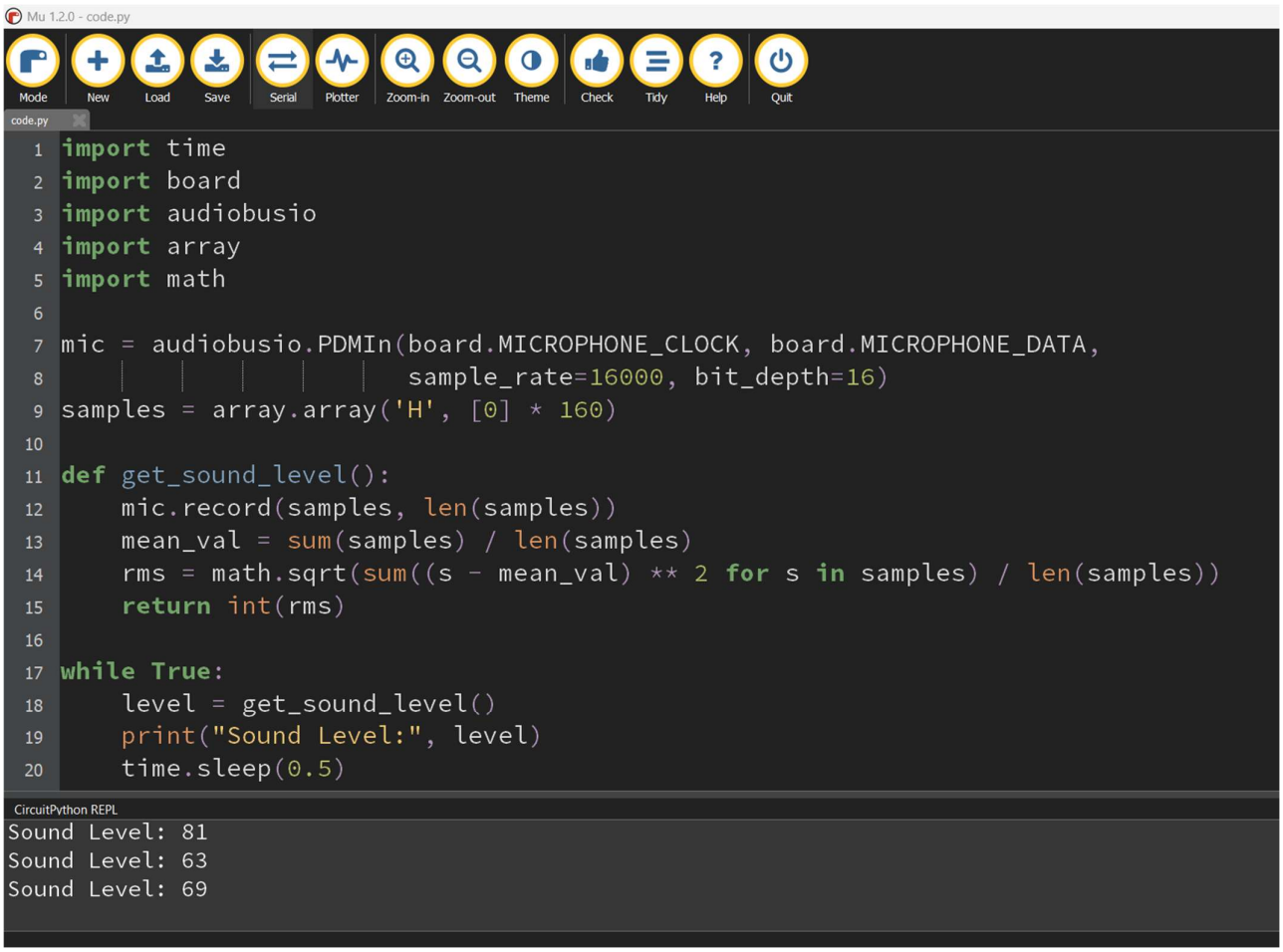
Getting Sound sensor value from CPX

- In exercise 1 we saw how to define a function. In this exercise also, we use a function to read sound sensor values.



- Save this file as code.py to the CPX drive.
- Look at the REPL (bottom part of Mu) to see the light value printed.
- You will see values like: Sound value: 1456, 102, 5000, etc.
- Make sound closer to CPX and see the value

Explanation for sound sensor function:

Step 1: Importing Required Modules

import time

```
import board
import audiobusio
import array
import math
```

These modules are needed for:

- working with time delays
- accessing CPX hardware pins
- using the microphone
- storing sound data
- performing mathematical calculations

Step 2: Setting Up the Microphone

```
mic = audiobusio.PDMIn(
    board.MICROPHONE_CLOCK,
    board.MICROPHONE_DATA,
    sample_rate=16000,
    bit_depth=16
)
```

This code activates the **built-in microphone** on the CPX.

- sample_rate = 16000 means the microphone listens 16,000 times per second.
- bit_depth = 16 determines how detailed each sound reading is.

This setup allows the CPX to accurately capture sound data.

Step 3: Creating a Storage Area for Sound Samples

```
samples = array.array('H', [0] * 160)
```

This line creates a list named samples that can store **160 sound readings** at a time.

Each reading is a number that represents the sound detected by the microphone.

Step 4: Defining the Function

```
def get_sound_level():
```

This function:

- records sound from the microphone
- processes the sound data
- returns a single number that represents how loud the sound is

Recording Sound Data

```
mic.record(samples, len(samples))
```

- The microphone records sound and stores the values inside the `samples` array.
- This happens **every time the function is called**.

Calculating the Average Sound Level

```
mean_val = sum(samples) / len(samples)
```

- This calculates the average of all recorded sound values.
  It represents the background sound level.

Calculating Loudness (RMS Value)

```
rms = math.sqrt(
    sum((s - mean_val) ** 2 for s in samples) / len(samples)
)
```

- This calculation measures how much the sound varies from the average.
  Greater variation means **louder sound**.
- This value is a reliable way to measure sound intensity

Returning the Sound Level

```
return int(rms)
```

- The function returns the sound level as a whole number.
- After this, all variables inside the function are cleared from memory.

Calling the Function

```
while True:
    level = get_sound_level()
```

Each time this line runs:

- the function starts fresh
- new sound values are recorded
- a new sound level is calculated

Adding a Delay

```
time.sleep(0.5)
```

- The program waits for half a second before repeating.
  This keeps the output readable and stable.