

Mission 5: Micro Musician

Student Workbook



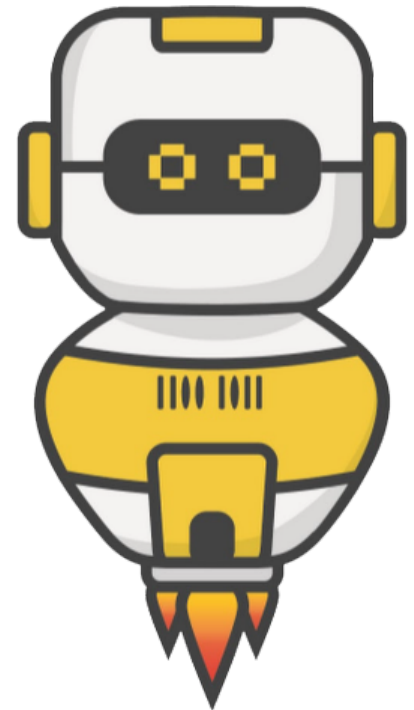


Mission 5: Micro Musician ✓

Play music and sounds with the CodeX and learn about code readability.

How about some music?

Computers and music go great together! This project brings together coding, electronics, and music. The CodeX has a built-in speaker, and there are lots of built-in tunes to play,



Go to the Mission 5 Log and fill out the Pre-Mission preparation.



Mission 5: Micro Musician

Musicians often use computers to help create music.



- Drum Machines
- Keyboard synthesizers
- Recording and Mixing with Digital Audio Workstation (DAW) Software

Mission 5: Get started

- Go to <https://sim.firialabs.com/> and log in.

- Go to Mission 5 

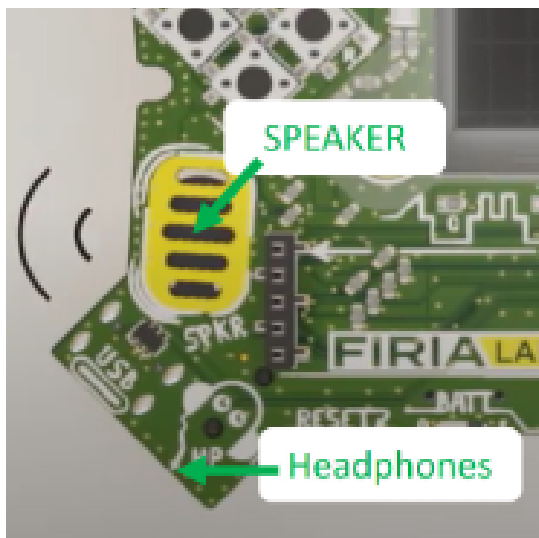
- Click **NEXT** and start Mission 5.

Objective #1: Sound Outputs

There are two ways to listen to sound on the CodeX.

- Built-in speaker
- Plug in headphones

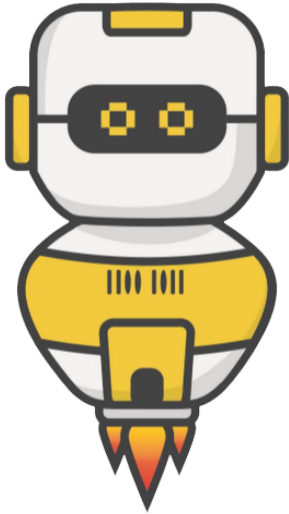
The CodeX uses a **codec chip** to change digital information into audio sound waves.



With code you can:

- Play sound files
- Beep tones
- Control volume
- And more!

Objective #1: Sound Outputs



DO THIS:

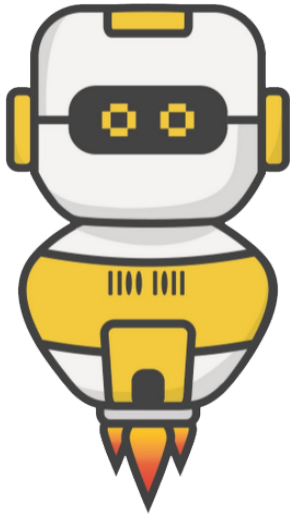
- Click on `audio` to add it to your toolbox
- Scroll down in the toolbox until you find `CodeX Sounds`
- Click on it and find the table with all the CodeX built-in sounds
- In your Mission Log, write down the names of sounds that you want to try

Example Audio Functions

The audio functions of the CodeX are exposed in the `audio` object from the `codex` module.

Try playing an MP3 file from the `CodeX Sounds` collection:

Objective #1: Sound Outputs

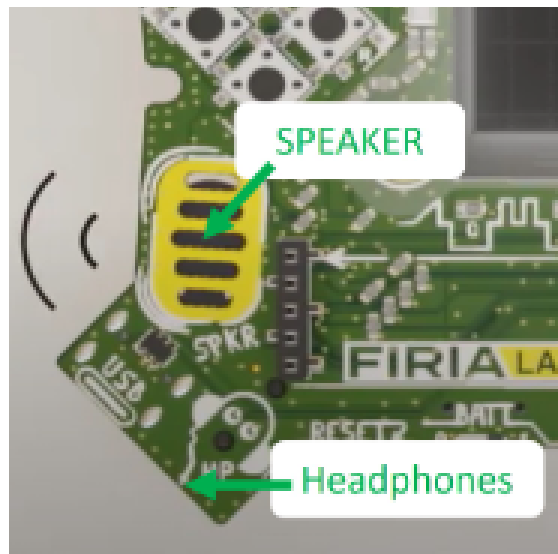


DO THIS:

Find the speaker and headphone jack.

DO THIS:

- Close the instruction panel
- Use camera controls to rotate the CodeX in the scene
- Click on the speaker
- Click on the headphone jack



Objective #2: Micro tunes

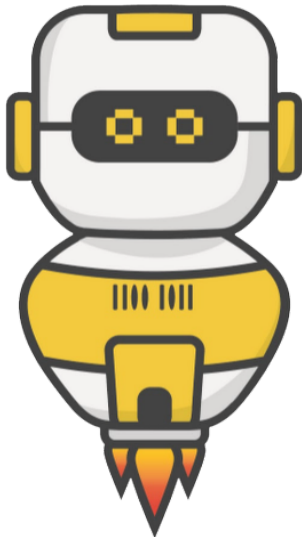
Now it is time to write code to play some sounds.

Start by playing an mp3 file.

- An mp3 is just an audio file in the mp3 format
- The CodeX has a few sample mp3 files already loaded

Here is an example:

```
from codex import *  
audio.mp3("sounds/welcome")
```



DO THIS:

- Create a new file called **Music1**
- Type code to play an mp3 file
- You can use the one in the example, or pick a file from the sounds your are interested in trying
- Run the code

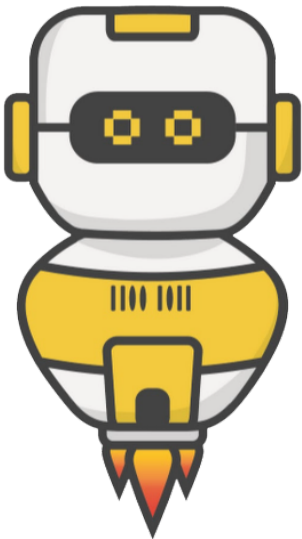
```
from codex import *  
audio.mp3("sounds/welcome")
```

Objective #3: Clean codes

Good code is easy to read by people.

As your programs get longer, you can do a few things to keep them readable:

- Use blank lines to separate sections of code
 - The computer ignores blank lines
- Add comments that explain what the code does
 - The computer ignores comments



DO THIS:

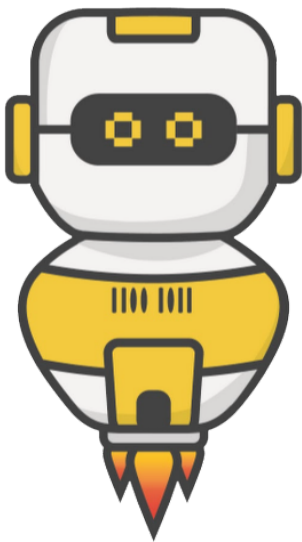
- Add a blank line to your code
- Run the code

```
from codex import *  
  
audio.mp3("sounds/welcome")
```


Objective #4: Once more, with feeling

You don't want the screen to be blank when the sounds are playing.

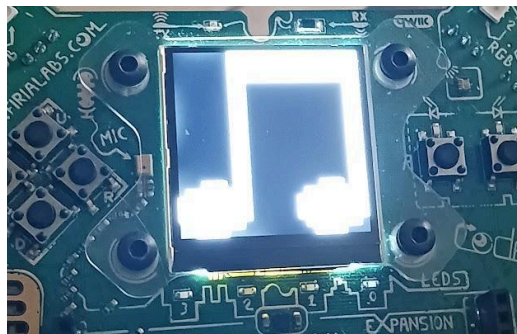
- You can use **display.show()** with an image
- Use this statement BEFORE playing the mp3 file



DO THIS:

- Add `display.show(pics.MUSIC)`
- Change the audio file to “sounds/africa”
- Run your code
- Answer the question on the Mission Log

```
Music1 x
1 from codex import *
2
3 display.show(pics.MUSIC)
4 audio.mp3("sounds/africa")
5
```



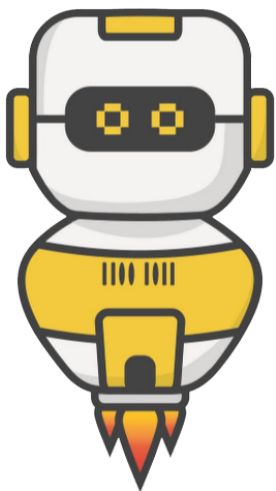
Objective #5: Comments

Making code readable to people was already mentioned.


You already learned about blank lines.


Now let's find out about comments.


```
Music1 x
1 # This is what a comment looks like
2 from codex import *
3
4 # You can add a comment anywhere
5 # It explains what the code does
6 display.show(pics.MUSIC)
7 audio.mp3("sounds/africa")
8
```




DO THIS:

- The instruction panel defines two words:
 - **Readability**
 - **Comments**
- Write the definitions in your Mission Log
- Click on  to add it to the toolbox

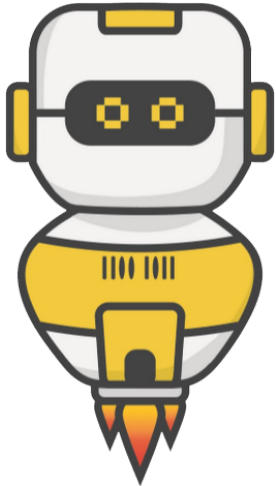
 **Readability:** Making code easy to understand for *humans*.

- Use descriptive variable names
- Use  **Comments** - notes in the code about what you're doing

In Python, anything that follows a `#` to the end of the line

... is a  **comment**, meaning it is *ignored* by the computer.

Objective #5: Comments

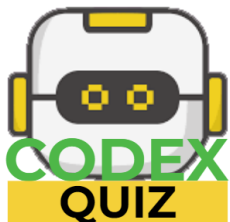


Add three comments to your code

DO THIS:

- Add a comment with your name at the top of your code (line 1)
- Add a comment before the `display.show()` statement (line 4)
- Add a comment before the `audio.mp3()` statement (line 6)
- Run the code

```
Music1 x
1 # Student Name
2 from codex import *
3
4 # Display MUSIC pic
5 display.show(pics.MUSIC)
6 # Play Africa song
7 audio.mp3("sounds/africa")
8
```



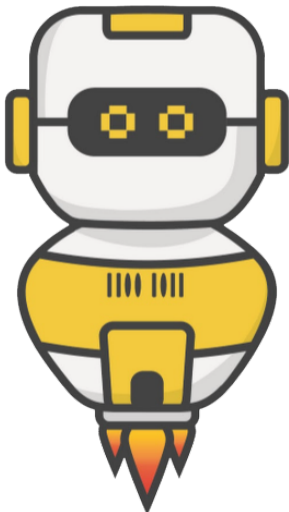
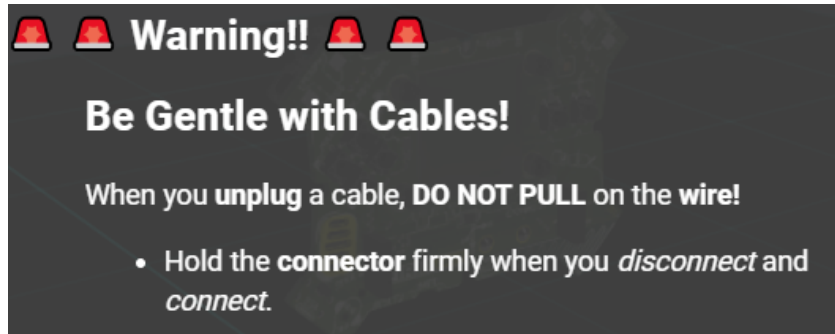
Mission Quiz: Sounds and Readable

Test your skills by taking the quiz.

Objective #6: Portable mp3s

After the code is running on the CodeX, you can go unplugged.

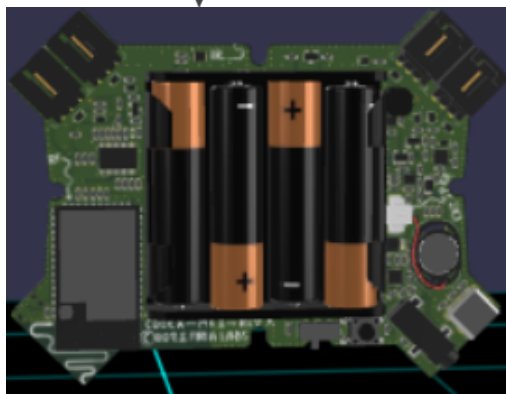
- After you run a program, it is loaded on the CodeX
- You can unplug the CodeX from the computer and run on batteries



DO THIS:

- Close the instruction panel
- Use camera controls to rotate the CodeX in the scene
- Click on the BATT switch

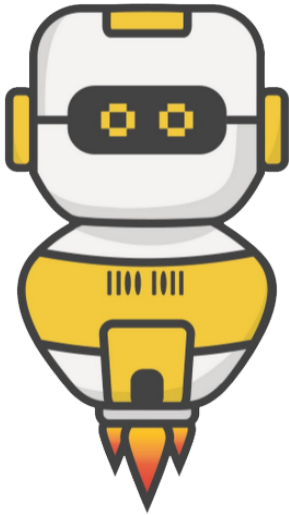
(OPTIONAL)



- Put batteries the CodeX
- Unplug from the computer
- Flip the BATT switch to position 1
- Enjoy your CodeX “unplugged”!

Mission Complete

You have completed the fifth mission.



Do this:

- Read your “Completed Mission” message
- Complete your Mission 5 Log
 - Post-Mission Reflection
- Get ready for your next mission!

Post-Mission Reflection

What are two ways you can hear sound from the CodeX?

1) _____

2) _____

What are two ways to make your code readable to people?

1) _____

2) _____

What are two ways you want to use sound or audio files in a program?

1) _____

Wait! Before you go ... Clear the CodeX

Go to FILE -- BROWSE FILES

Select the “Clear” file and open it

Run the program to clear the CodeX

Okay. Now you can go.