WAKE TECH
Biotechnology, Engineering, & Skilled Technologies

Introduction to Python Programming

BAT-212: BAT Logic and Programming



This material is based upon work supported by the National Science Foundation Advanced Technical Education grant program, A New Technician Training Program for Advanced Building Technologies, DUE-2000190.

The opinions, findings, and conclusions or recommendations expressed are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Introduction to Python Programming© 2024 by Wake Technical Community College is licensed under CC BY-NC-SA 4.0.
To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-sa/4.0/

Introduction to Python Programming

OBJECTIVES

Upon completion of this activity the student will be able to:

- Connect to and run a python program on the Circuit Playground Express.
- Edit a Python program with the Mu IDE.

PARTS AND EQUIPMENT

- Circuit Playground Express board.
- Computer
- Access to internet for Mu IDE download.

REFERENCES

The following are immediately relevant to the lab:

- https://learn.adafruit.com/welcome-to-circuitpython
- https://codewith.mu/en/about
- https://learn.adafruit.com/adafruit-circuit-playground-express
- https://learn.adafruit.com/circuitpython-made-easy-on-circuit-playground-express/first-things-first
-

The following are general references:

- https://greenteapress.com/thinkpython/html/thinkpython002.html#toc5
- https://docs.python.org/3/tutorial/

BACKGROUND

We are using the Circuit Playground Express Developer board to learn the basics of Python programming. We will be using a version of Python called CircuitPython that has added features over standard Python to allow interaction with hardware inputs and outputs. Our focus in these labs is learning Python rather than wiring and circuitry, but using the board makes it more fun (we hope) and easier to see the results of your programs.

PROCEDURES

Installing the software

Important Note: The version numbers listed below are probably not the current versions you will find when you go to the websites. This lab will give the general form of what is available

and should work, but the software is updated frequently and sometimes code that worked previously will not work in the current version.

- Plug your board into the USB port of the computer. If the board does not show up as a drive on your computer, you will need to load Circuit Python to the board with the following steps:
  - Download CircuitPython from here: https://circuitpython.org/board/circuitplayground_express/
  - Load CircuitPython to the board, following the instructions here: https://learn.adafruit.com/welcome-to-circuitpython/installing-circuitpython Instructions are given here about resetting the board so that the bootloader is detected.
- If you are using a lab computer, the Mu editor should already be installed. Run the Mu Editor application. If it is the first time, it can take a while to set up. If you are using your own computer, or wish to install on a home computer, please go to https://learn.adafruit.com/welcome-to-circuitpython/installing-mu-editor and follow the instructions there.

Running code

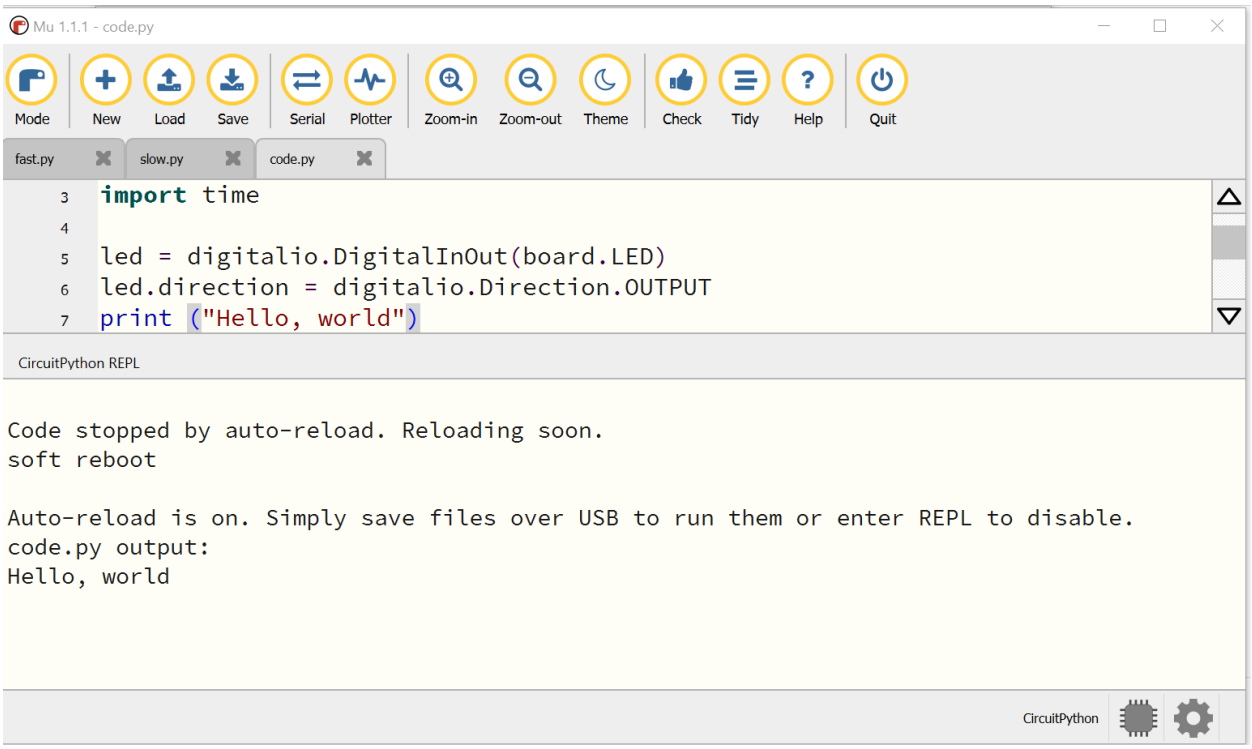
- Go to https://learn.adafruit.com/adafruit-circuit-playground-express/creating-and-editing-code and copy the code from the window in the middle of the page to the Mu Editor.
- Save to the board using the save button on the top tool bar of Mu. Save to *code.py*. When you save the code, it will start running on the board and will make the small red LED flash (pin D13). The function time.sleep() controls the blink rate.

The code that runs on the board is saved to *code.py*. You can save programs under other names, but to run them, they must be copied to *code.py*.

- Open a new file in the editor and paste the same code in again. Change the time.sleep argument in the new file to 1.0 for both instances. Save the file under the name *slow.py*. Notice that the blink rate did not change. Now under the directory, close *code.py*, double right-click on *slow.py* and rename the copy to *code.py*. You should see the flash rate change.
- In the current *code.py* file, add the following line before the *while*:

  print("Hello, world")

- Open the serial monitor using the serial button on the editor.
- Save *code.py*
- Your window should look something like this:



If you do not see the 'Hello, world' output, try saving again. This reruns the code. Since the print statement is not in the while loop, it runs only once when the code first loads.

Including Libraries

Go to https://circuitpython.org/libraries and download the bundle for CircuitPython 7.x.x. Unzip the bundle to the D: drive on the class computers, or to an appropriate place on your own computer. Note that there can be trouble with path length, so place it somewhere high in the directory structure. We will copy only the libraries we are using to the lib folder on the CIRCUITPY drive. Go to https://learn.adafruit.com/welcome-to-circuitpython/circuitpython-libraries to learn more about using Circuit Python libraries.

Decision Statements Program 1

- Go to the directory where you unzipped the bundle. Inside that directory you will find an *examples* directory. In that directory, find the file *circuitplayground_button_a.* Copy it to the CIRCUITPY drive.
- Use the "Load" button to bring this file up in the Mu editor. Notice that the code includes the line "while True:" which means the code will loop forever.
- Save the code as code.py and see what the code does.
- Now copy the file *circuitplayground_button_b* to the CIRCUITPY drive. Save the code as code.py and see what the code does.
- How is the button_b code different from the button_a code?
- Let's change this code so that it acts like an S-R latch. Remember that an S-R latch will hold the last active state, so if SET was last asserted, the output will be held as SET until

RESET is asserted. Then the output will be held as RESET until SET is asserted again. Use button_a as the SET (turning on the LED) and button_b as the RESET (turning off the LED). Do you want to use an *elif* or an *else* statement?

- **Comment your code well and submit.** No other lab report required.
    - Header comments should include:
        - Attribution for code you used as basis for your code.
        - List of inputs (such as pushbutton)
        - List of outputs (such as red LED)
        - Summary of what the code does
        - Anything the user should be aware of in terms of assumptions or possible difficulties.
    - In-line comments should provide explanation of each significant process in the code (this is based on your judgement; there is a balance between too much and not enough).

Decision Statements Program 2

- Go back to the directory where you unzipped the bundle, into the *examples* directory. In that directory, find the file *circuitplayground_light* and copy it to the CIRCUITPY drive. This code will read the light sensor on your board.
- Use the "Load" button to bring this file up in the Mu editor. Again notice that the code includes the line "while True:" which means the code will loop forever.
- Save the code as code.py and see what the code does. You will need to open the Serial monitor to see the print statements. Take note of the range of light values. Use the light on your phone to get the high end of values.
- We will implement hysteresis using the if-elif-else structure. Remember that hysteresis means that the element turns on at a one level of the input and turns off at a different level. Choose two values in the range. I used 70 and 40 – just pick 2 levels that you can easily differentiate and get as output. If the light level is greater than your top value, turn the red LED on. If it drops below the low value, turn the LED off. For in-between values, it should keep whatever was the last value.
- Test the operation of your code and demonstrate to the instructor.
- **Comment your code well and submit.** Using the commenting guidelines given for the Program 1 above.