



JULY 28, 2023

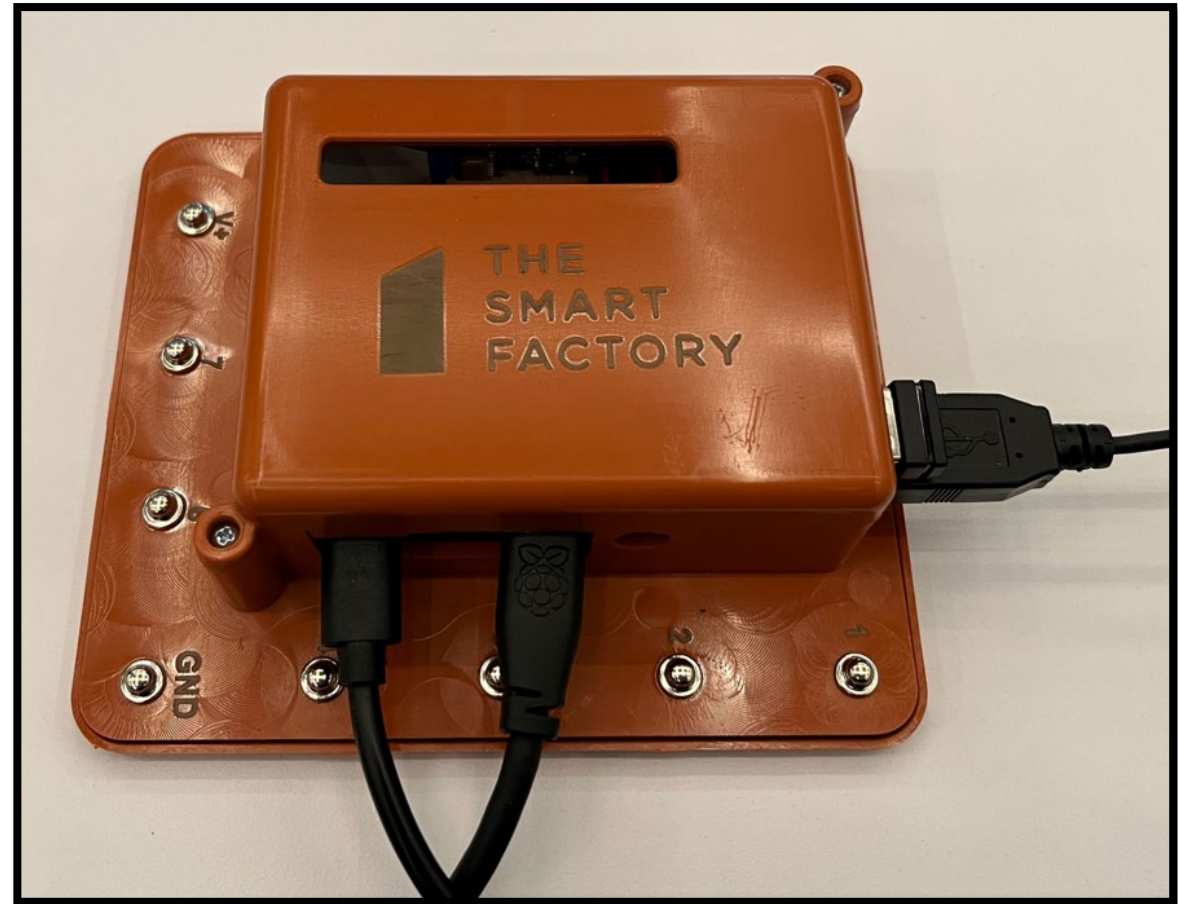
# Smart Factory Believers

**EMPOWERING THE STEM  
WORKFORCE OF TOMORROW**

**AUTOMATED CAMERA REPAIR**

# Purpose

In order to use the camera included in your housing, you will need to execute the steps on the following slides.



# Required Resources

- Pi Housing
- HDMI to Micro HDMI cord
- USB-C power supply
- Keyboard
- Mouse
- Monitor
- Wi-fi connection





# Initial Setup

All cords need to be connected to the housing before it is connected to power. Keyboard, mouse, HDMI from monitor, then finally power. After a few moments, the pi should power on and you will see the desktop for the pi displayed on the monitor.

If the screen does not turn on, unplug the power and plug it back in. Ensure all other cords are plugged in prior to powering on the unit.



# Open the Web Browser

In the top left-hand corner of the desktop is the web browser icon. When you open the web browser it should open a window on the desktop.

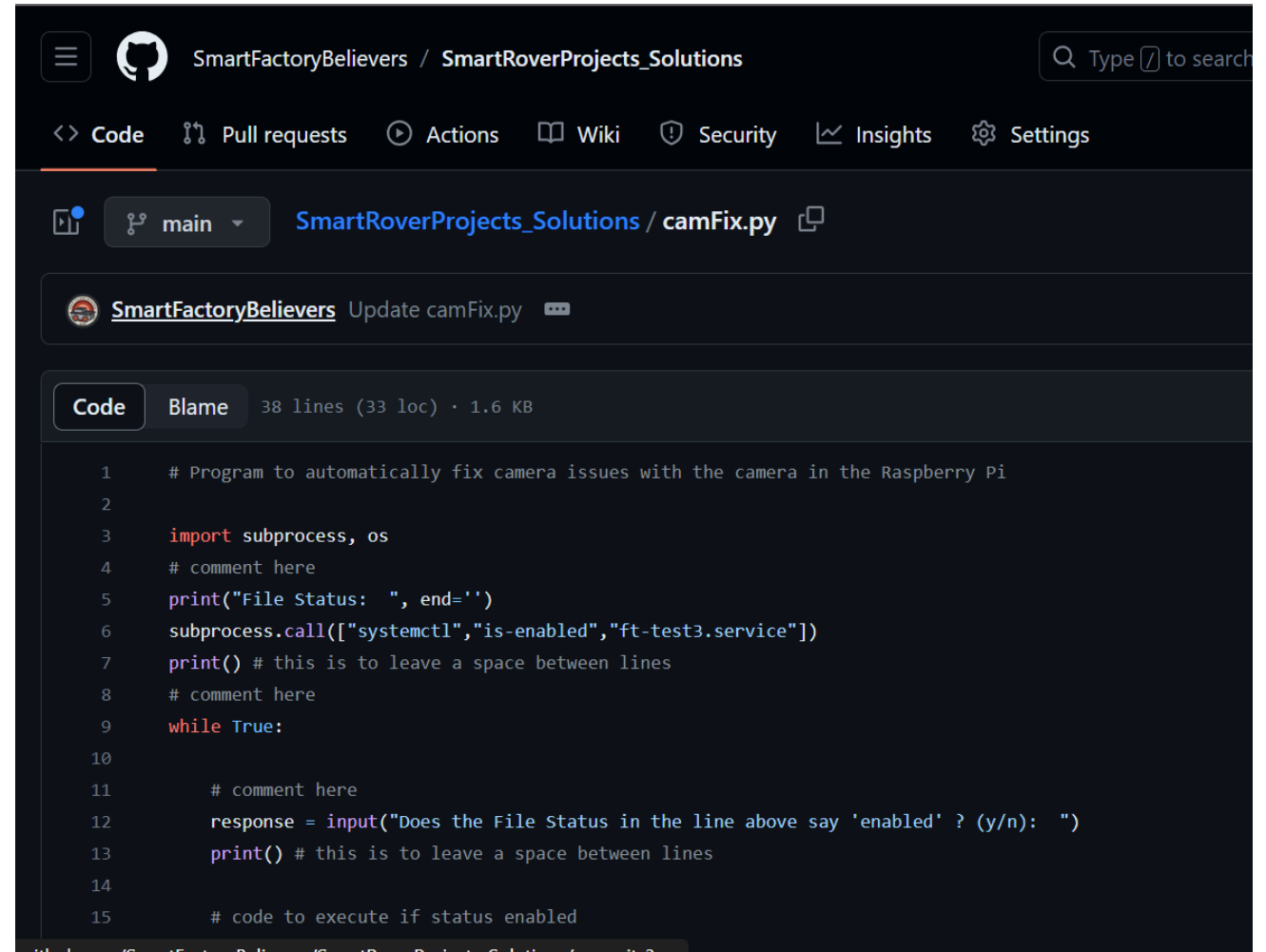
Once the browser window is open, go to <https://github.com/SmartFactoryBelievers>

Under repositories, navigate to **“SmartRoverProjects\_Solutions” > camFix.py**



# Copy camFix.py

Highlight all of the code (lines 1-38) and copy. Once the code is copied, minimize the browser window to return to the desktop.



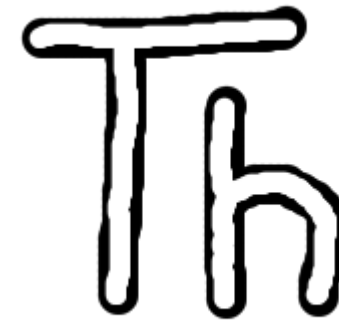
The screenshot shows a GitHub repository page for 'SmartFactoryBelievers / SmartRoverProjects\_Solutions'. The file 'camFix.py' is selected, showing its code. The code is a Python script for automatically fixing camera issues on a Raspberry Pi. It includes comments and uses the subprocess module to execute systemctl commands. The code is displayed in a dark-themed editor with line numbers from 1 to 15 visible.

```
1 # Program to automatically fix camera issues with the camera in the Raspberry Pi
2
3 import subprocess, os
4 # comment here
5 print("File Status: ", end='')
6 subprocess.call(["systemctl", "is-enabled", "ft-test3.service"])
7 print() # this is to leave a space between lines
8 # comment here
9 while True:
10
11     # comment here
12     response = input("Does the File Status in the line above say 'enabled' ? (y/n): ")
13     print() # this is to leave a space between lines
14
15     # code to execute if status enabled
```

# Open Thonny IDE

On the desktop is the Thonny IDE “Th” icon. This opens the python code editor.

Click the icon to open Thonny IDE.

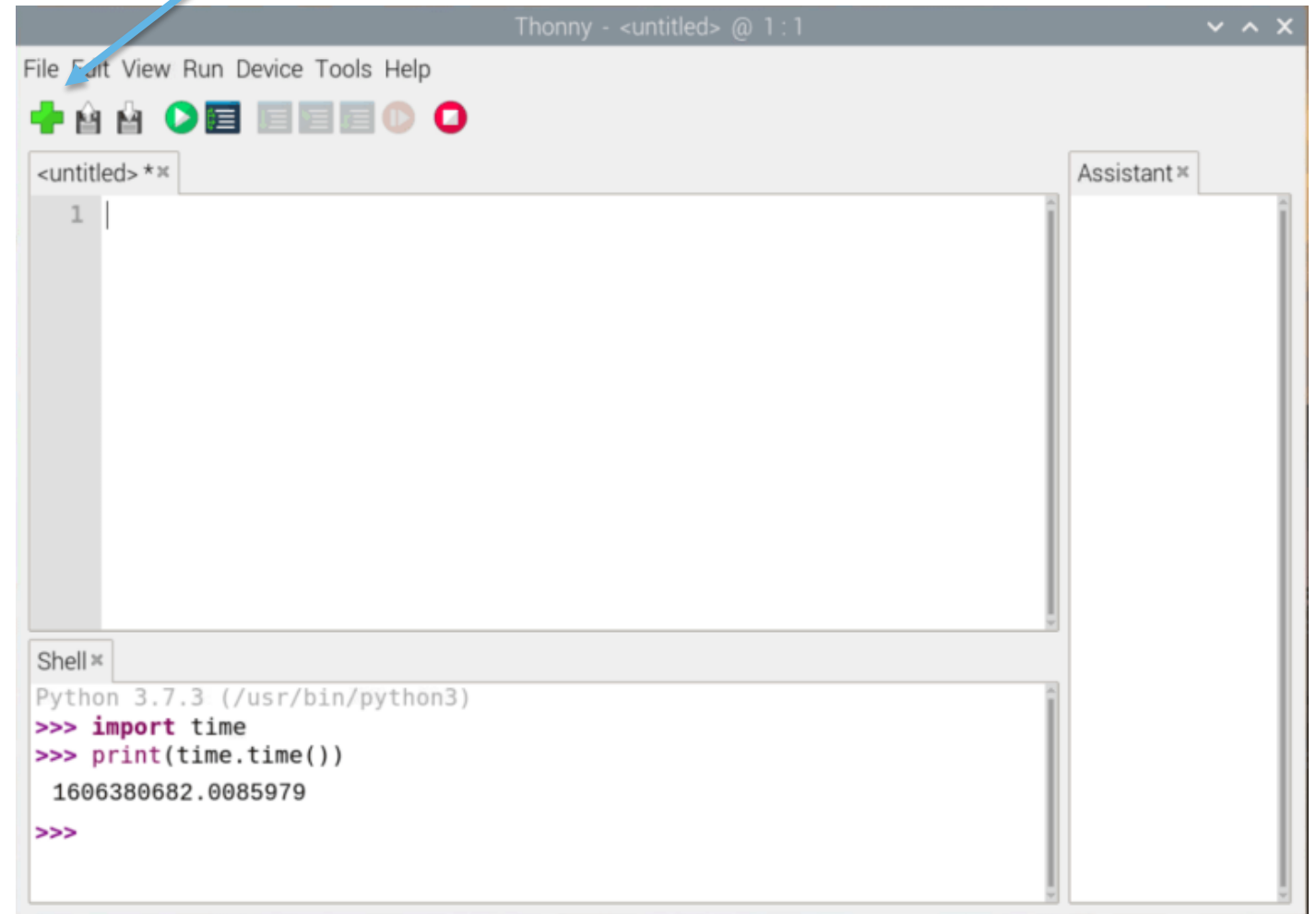


# Create a new code file

Create new file

In the top-left corner, click the green plus sign “+” to create a new code file.

Paste in the code from Github, rename the file “camFix.py”, and save the file.





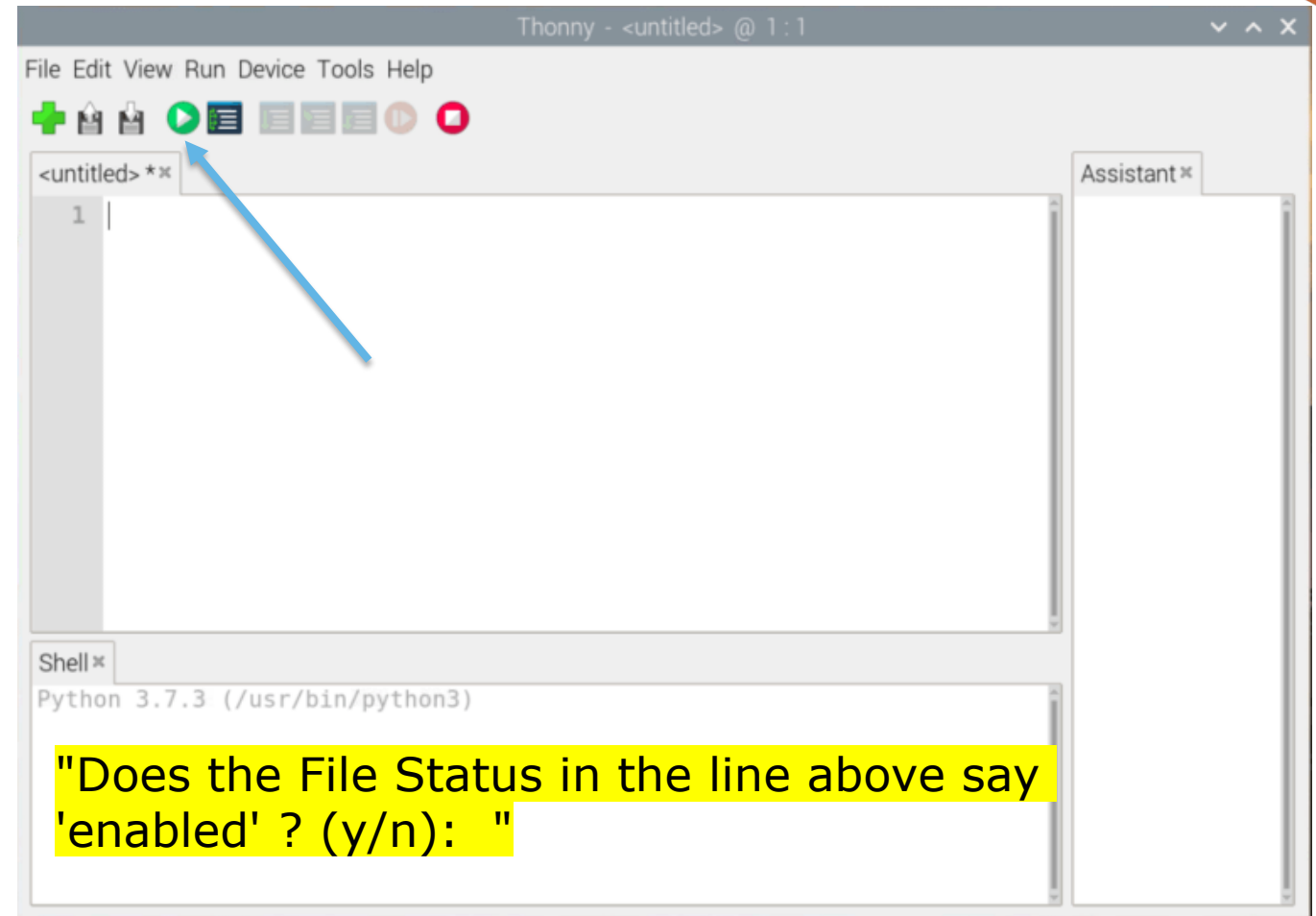
# Run the new code file

In the top-left corner, click the green play button to run the “camFix.py” code file you just created.

Watch the Shell at the bottom of the screen. The program will check your system for the ft-test3.service file status to determine if a fix is necessary:

if Enabled, type “y” and “enter” key to resolve the issue

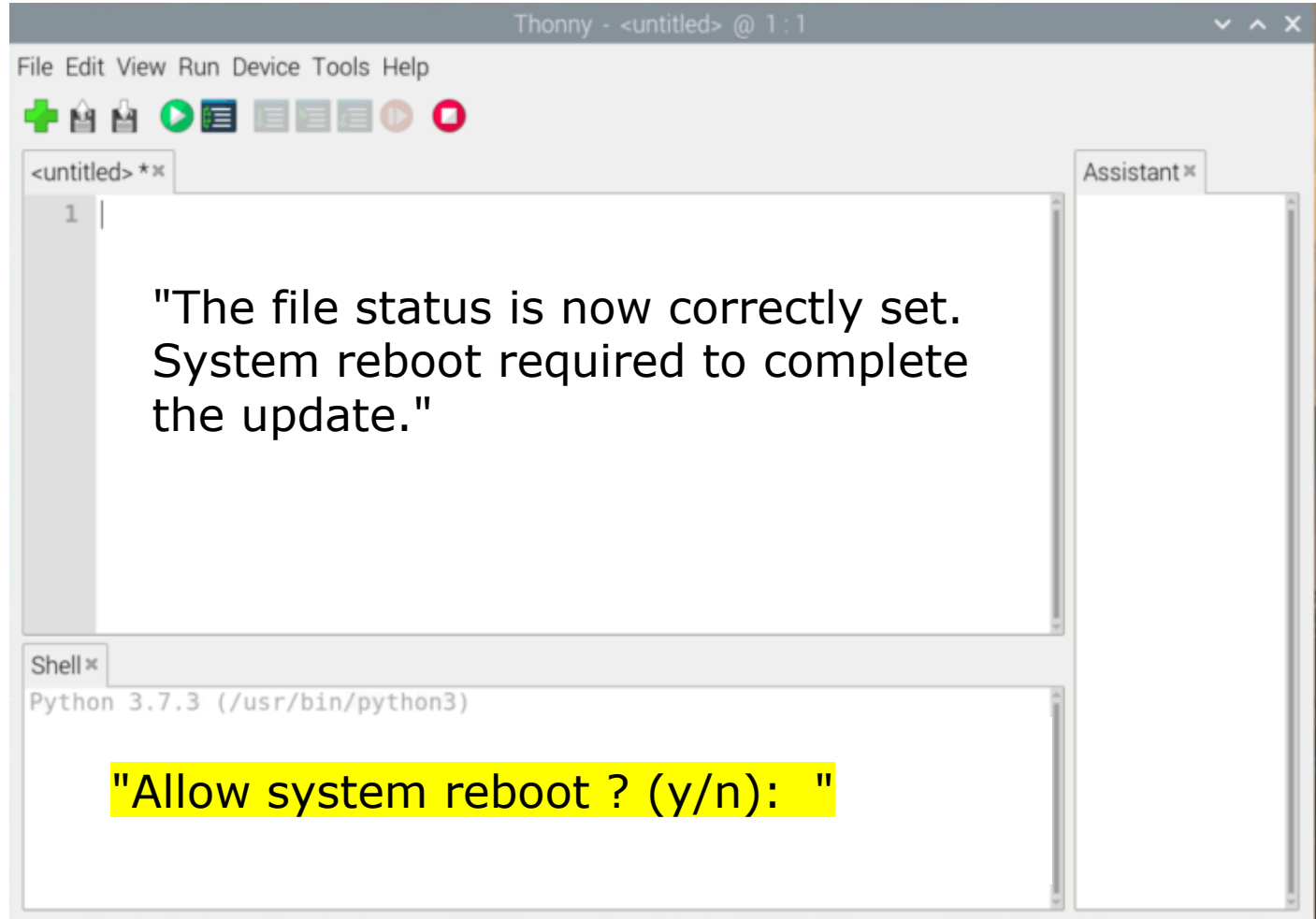
If Disabled, type “n” and “enter” key to exit the code.



# If required, allow pi to reboot itself

If the file is enabled, a system reboot is required to complete the update and resolve the issue.

Watch the Shell at the bottom of the screen. If prompted, type “y” and “enter” key to reboot the system



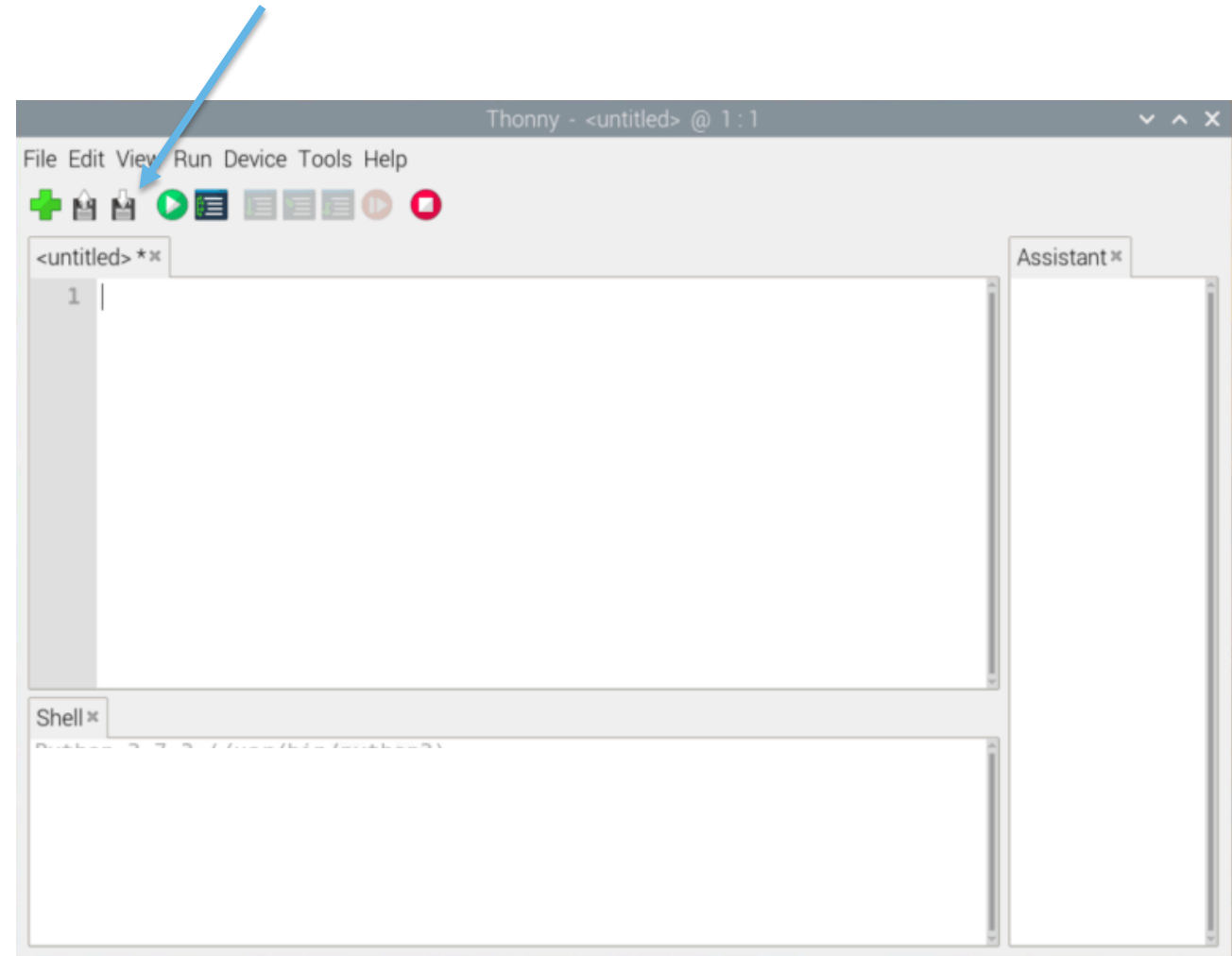
The screenshot shows the Thonny IDE window titled "Thonny - <untitled> @ 1 : 1". The main editor area contains a single line of text: "The file status is now correctly set. System reboot required to complete the update." Below the editor is a "Shell" window titled "Shell\*" showing the prompt "Python 3.7.3 (/usr/bin/python3)". The prompt is followed by the text "Allow system reboot ? (y/n): " which is highlighted in yellow. To the right of the main editor is an "Assistant\*" window.

# After reboot is complete, test the camera

After the reboot, click “load” icon and select project 08: Introduction to Camera

Once the project is loaded, click the green play button to run the project and test the camera

Th





#### **About The Smart Factory @ Wichita**

The Smart Factory network includes Deloitte's several immersive experiences designed to accelerate digital transformation. One of the key locations, The Smart Factory @ Wichita, is designed in collaboration with ecosystem participants to demonstrate firsthand how manufacturers can evolve for the future and up-level their capabilities and processes through technological innovation. The Smart Factory @ Wichita will be a net-zero impact smart building on a smart grid featuring 60,000 square feet of sustainable space. It is the evolution of Deloitte's existing experience at Wichita State University, which features 40+ robots, 26 AR/VR assets, 3D printing, engineering software programs, and more.

#### **About Deloitte**

Deloitte refers to one or more of Deloitte Touche Tohmatsu Limited, a UK private company limited by guarantee ("DTTL"), its network of member firms, and their related entities. DTTL and each of its member firms are legally separate and independent entities. DTTL (also referred to as "Deloitte Global") does not provide services to clients. In the United States, Deloitte refers to one or more of the US member firms of DTTL, their related entities that operate using the "Deloitte" name in the United States, and their respective affiliates. Certain services may not be available to attest clients under the rules and regulations of public accounting. Please see [www.deloitte.com/about](http://www.deloitte.com/about) to learn more about our global network of member firms.