Part I: For someone with Linux and Python 2.7 (but, obviously, not our logic board):

Getting Python 2.7:

Many distributions of Linux already have Python installed. You can check by opening a terminal and typing in the command

```
apt-cache search python | egrep "^python2.[0-9] " –color
```

The *egrep* part filters out other stuff than the directory, which is all you want to see.

We used Python 2.7; earlier versions might work

If you don't have Python installed, you can open a terminal and type:

```
apt-get install python
```

RedHat and some other flavors of Linux need:

```
yum install python
    OR
sudo yum install python
```

You might need to get additional packages such as colorama or py.audio, using apt-get install XXXX. Linux will tell you what you're missing

About the program:

The program adapted to use on a Linux machine is here, elements_linux_KB_no_GPIO.py. The qualifiers *KB_no_GPIO* indicate that our special controls using buttons and input-output pins on a Raspberry Pi computer had to be mimicked using keyboard inputs

Compared to the Linux version of the program running on a Raspberry Pi computer, there are some changes:

Since most Linux machines other than the Raspberry Pi or Arduino lack general-purpose input output (GPIO) pins and no one has a logic board controlled by the GPIO pins such as we made, I stripped out all code using the GPIO pins

Similarly, because anyone else's machine doesn't have the buttons we use to select items (presentations, elements within presentations), we replaced all references to button inputs with keystrokes on the keyboard. E.g., the line

if GPIO.input(10)==1: # pin 10 is normally lo; green button; increment the element number

has to be replaced by a reading from the keyboard, *n=raw_input()* and then tests such as

if n=='g': # pin 22 is normally lo; green button; increment element number

The program as we developed it uses colorama to print color-coded output on the screen. If you don't have it, you can get the package at https://pypi.python.org/pypi/colorama or *apt-get* or *pip* .

Running the program is straightforward, with *python elements...py*. If you copy the program off the DVD, be sure to keep the program and the narrations files in the same directory.

Running the program:

This is straightforward.

Open a command window. I always have a shortcut on my Desktop, but if you don't, right-click on the start button and scroll up to Run; click on it.

In the little dialog box that opens, type in Run and then Enter.

Change to the directory into which you copied the version of elements.py that you want to run – e.g., for the Windows version, it is the file *elements_windows_KB_no_GPIO.py*.

Assuming that you have the PATH variable set so that Python can be run from any directory, such as this one:

Now just type in elements.py and Enter. Follow the prompts that tell you how to run the parts of the presentations you want

In Linux, you will need to specify python, usually:

```
python elements.py
```

With the GPIO pins used on our physical periodic table, we had to become superuser:

```
sudo python elements.py
```

Part II: For someone with Windows and having Python 2.7 or willing to install it:

Getting Python 2.7:

Use the installer on this DVD (python-2.7.12.msi), or go to the official Python site (don't risk malware from other sites), www.python.org/downloads or, just for Windows, www.python.org/downloads/windows

The installation is straightforward. However, it might not put Python's executable application into your Path Variable, which is necessary if you want to run Python from any folder other than the one where Python got installed. Here are two links showing how to do this:

https://www.youtube.com/watch?v=rWNhvq514Qs
https://www.youtube.com/watch?v=Y2q_b4ugPWk

However, these don't apply to all Windows variants, so, here's a better way, for Windows 10 (and there are similar ways to get to Control Panel in other versions):

Find out (using File Explorer), or remember, where you asked for Python to be installed. Mine is C:\Python27, handily

Open the Control Panel – e.g., right-click on the start button (the little 4-panel window at bottom left)

Click on User Accounts

At left, left-click on Change my environment variables

In the dialog box that opens, click on Path to highlight it

Below that, click on Edit

You'll see a more or less lengthy text; you can add at the end (or middle, if you're careful to keep all entries separated by semicolons) a semicolon and then the name of the folder where Python is installed. If my path had not been updated to include Python, I would have typed in at the end simply ;C:\Python27

About the program:

The program adapted to use on Windows is here, elements_windows_KB_no_GPIO.py. The qualifiers *KB_no_GPIO* indicate that our special controls using buttons and input-output pins on a Raspberry Pi computer had to be mimicked using keyboard inputs

Instructions for using Python

If you don't want to install Python, see Part III below. If you want to use Python but don't have it installed yet, you can get it from many places, while the only fully trusted and up-to-date site is https://www.python.org/downloads/, where you can also find instructions for installation. There is also a Python msi installer for Windows.

I think the basic installation has all one needs. If I've overlooked anything and the screen says "module XXX not found" or the like, you can type a line in the Command Prompt window *python –m pip install XXX*

Compared to the Linux version of the program running on a Raspberry Pi computer, there are some changes:

Since a Windows machine lacks general-purpose input output (GPIO) pins and no one has a logic board controlled by the GPIO pins such as we made, I stripped out all code using the GPIO pins

Similarly, because anyone else's machine doesn't have the buttons we use to select items (presentations, elements within presentations), we replaced all references to button inputs with keystrokes on the keyboard. E.g., the line

if GPIO.input(10)==1: # pin 10 is normally lo; green button; increment the element number

has to be replaced by a reading from the keyboard, *n=raw_input()* and then tests such as

if n=='g': # pin 22 is normally lo; green button; increment element number

The program clears the screen frequently so that all relevant text with instructions for the program stay on top of the screen. We did this in order to use a tiny TFT monitor with room for only 10 lines displayed. In Linux, the call to clear the screen is *clear* (invoked in Python with a system call, os.system(clear)); in Windows, this is changed from *clear* to *cls*.

To play the recorded narrations, the program on a Linux machine uses a system call *os.system(aplay somefile.wav)*, where the actual string *somefile.wav* is created with reference to the element being looked at, such as *discovery_of_lithium_narration.wav*. The Windows version has to use *winsound.PlaySound(somefile.wav,0)*. The program has to use a line of code *import winsound*.

The Linux version uses colorama to color-code the presentation within a terminal. In theory, on a Windows machine, one can get colors on a command window by installing ansicon (http://softkube.com/blog/ansi-command-line-colors-under-windows, or https://github.com/mcandre/ansicon-win) and also installing colorama itself: in a command window, enter:

python –m pip install colorama

I used the second option to install ansicon 1.60. Alas, using it by going to the adoxa directory and clicking on ansicon.exe opened a command window but in that window python was unable to run....and I had to uninstall ansicon to get python to work in a standard command window. This was in Windows 10, 64-bit OS. If anyone succeeds in getting colorama to run nicely with Python, please let me know! Thanks.

Running the program is straightforward, with python elements...py. If you copy the program off the DVD, be sure to keep the program and the narrations files in the same directory. Note: Python.exe may or may not be added automatically to your environment variable, which allows you to run Python from any directory. In Windows 10 (and probably most earlier OS versions), go to My Computer, right-click on it to get Properties, then select Advanced, then Environment Variables (you'll need to invoke admin privileges). In the dialog box, go down to System variables; click on Path, then Edit. At the end of the line Variable values, add a semicolon and then *C:\Python27* (or the directory where you installed Python 2.7, if you didn't let the installer use this default directory).