DOCUMENTATION

This program is implementing the interaction for a pushbutton-input and auditory-output interface for setting the day and time on a clock. This user interface can be thought of as the software that would be used for a device such as that shown below, in which there is an auditory display but no visual display, and in which there are five buttons for input, though your systems will use the six "home" keys on the keyboard (space, j, k, l, r and ;).

All testing for this program was done through a terminal on a mac-book pro running Sierra OS so other systems may experience unknown issues. The only dependencies for this program are the 'readchar' library and 'pyaudio'. However before installing pyaudio one may need to install homebrew and use 'brew install portaudio' before installing pyaudio. Another side note is that the user should use pip3 to install pyaudio and readchar as this program was made and tested with python3.

Running program:
After installing dependencies navigate to directory of program and run program with python3, the user will be presented with an audio output of button mappings to interact with software. Once day is selected the prompt will direct on how to select the proper hour and after selecting the hour the program will be terminated.

Notes for developers:
This program is compiled of two python files and 2 directories for audio files: sound.py, set_clock.py, wav_files_provided, my_wav_files. The main class in set_clock.py will set the tone for how the program is run, that is, it calls a series of

functions the main one being "run_menu()". This run_menu has variables in place if one needs to implement minutes, audio files are also provided up to 60 for minutes. If changing button layout is need one must edit the "create_menu_globals()" function in set_clock.py, in which there is globals declared and assigned for buttons in operation. When changing button layout one must also create custom sound files in order to ensure consistent audio instructions. A utility function named "verify_sound_filenames" is also included this will need to be appended in order to test all used sound files. That being said all sound files within have been tested and work, the sound files that were included in "my_wav_files" were created first by recording terminal voice "Alex" and then run through "audacity" audio program to edit and deploy.
The sound.py works by first creating sound objects and then returning them to set_clock.py.