1. We chose to do the clock for our project and decided to do it because we wanted to do some work with outside libraries. Date and time tracking are important features in a number of applications and we decided to make our own clock and date tracker since we may need to apply these concepts later on in future projects and wanted to familiarize ourselves with it.
2. The code is fairly simple, all a user needs to do is run the code and it will work. No imput beyond that is necessary.
3. Technically speaking, it is also very simple. There’s a single while loop that says while true, and there is no false condition so the loop is constantly running all the time constantly updating every tenth of a second. We had it update every second by telling it to sleep for 1 second, but the problem is that it wouldn’t always sleep for the exact same amount of time, so it would sometimes skip a second when displaying the time so we adjusted it to update every tenth of a second. We imported the date time library and the panel library.
4. As stated previously, the only issue we had was that occassionally we would have the clock skip a second because it wasn’t sleeping for exactly one second every time, so we just changed it to update every tenth of a second and display the most recent result of the code.
5. We could add a stopwatch or a timer feature to it by staking the measurement between two times. For example, if we wanted to do a stopwatch, we would add a user input to begin the stopwatch opening another panel and then when they hit enter, the stopwatch will begin counting by taking the current time and subtracting the time the stop watch began, and when they hit enter again, the time stops counting. If we wanted to make a timer, we would have the user select a time and then have the program constantly subtracting the current time from the time selected until the value hits 0 and then the program would do something.