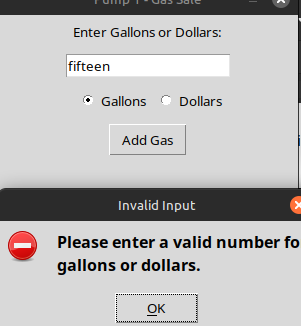
# Data Validation

I had four data sets in mind when testing my program. The first set was incorrect data types, the second set was to test how it interacts with negative numbers, the third was to test how it handles empty input or zeroes, and the fourth was to see how it functions under a typical use case.

# Incorrect Data Types

The program was able to successfully detect strings as an incorrect data type and displayed a pop-up window to notify the user of their error in all the places such an error might occur. Here’s a screenshot showing a string being input where a number is expected, and receiving an error message.



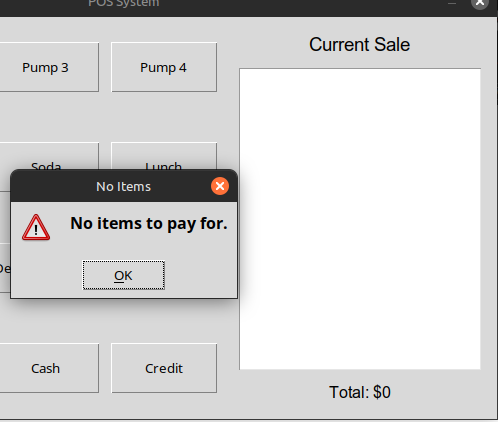
Besides the gas pump screen, this also correctly functions for setting the gas price, setting the price of a custom item, or entering the date for age-verification.

# Negative Numbers

The program is also correctly set up to handle negative numbers, so that you can’t accidentally empty the till. Here’s a custom item with a negative price, showing the error message:

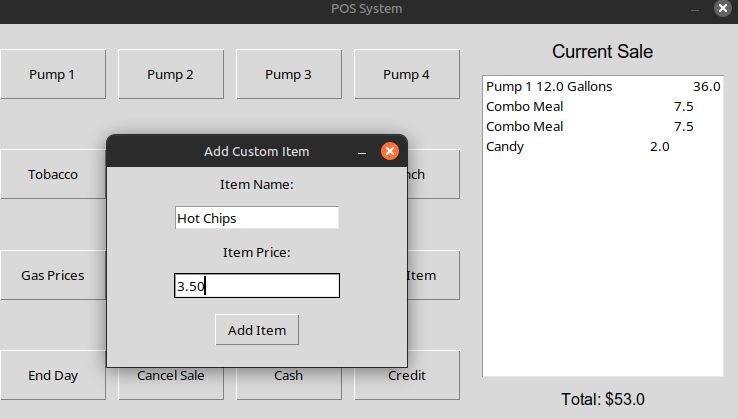
# Empty Fields

All empty fields correctly block entry, with the exception of the ability to create a custom sale item that costs zero dollars. This is intentional behavior that allows for certain items to be “free” under certain circumstances (promotions or sales perhaps.) In addition, selecting any of the payment options while the sales list is empty gives a special error:

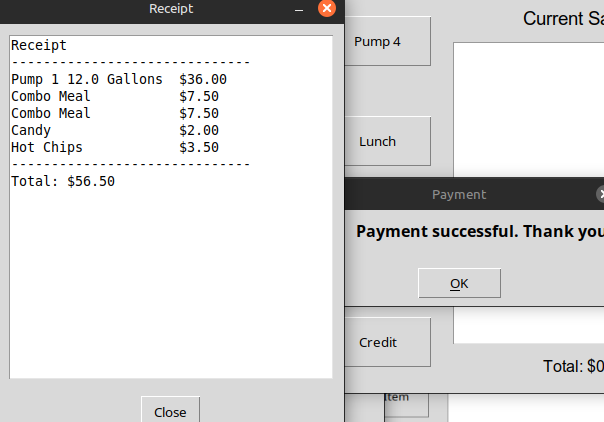


…. and trying to pay with less cash than is required gives a specific error:

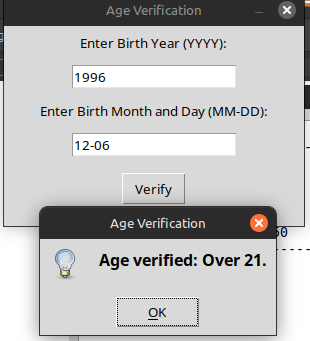
# Typical Use Case Screenshots

Finally, here are some screenshots of each module working correctly, using appropriate and typical data, as a control. Here is adding some typical items to the sales list, including gasoline, lunch items, and a couple custom items:

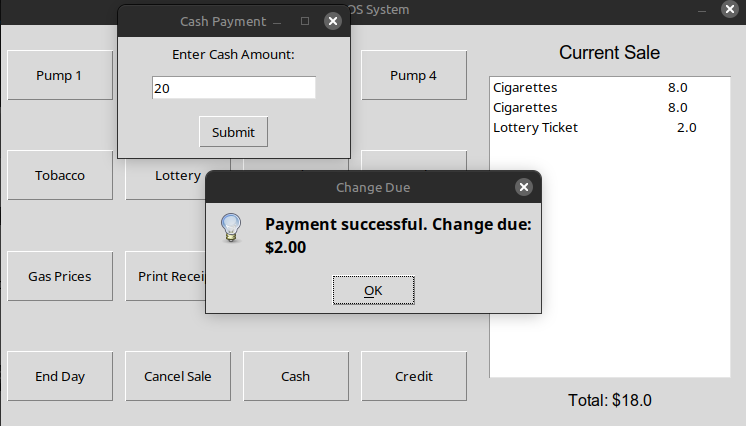
Here is a successful receipt:



Here is successful age verification:



Paying with cash and making change:

And an End of Day Sales Report:

