# Debugging Exercise

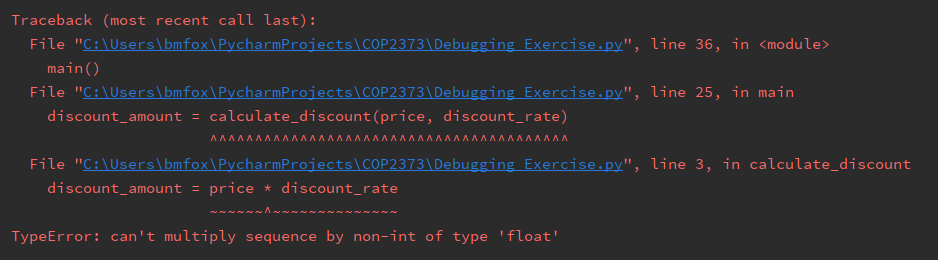
COP 2372, Week 2

05/20/2025

Brayden Fox

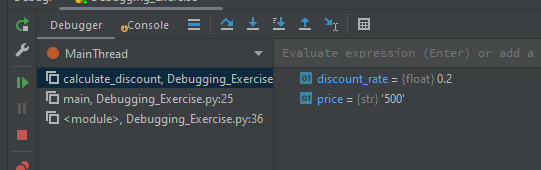
## Identifying the Error

Running the code without changes produces the following error:



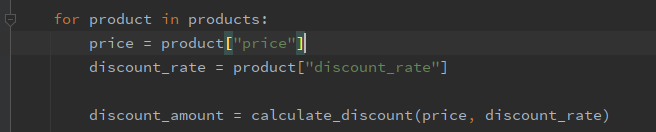
This tells us that on this particular line inside the function `calculate\_discount`, `discount\_amount = price \* discount\_rate`, the interpreter was asked to perform multiplication on two incompatible types, one of which is a sequence.

To determine the kind of sequence, we’ll add a break point at this line and inspect the variables. The first couple iterations are fine. We see an integer `price` being multiplied with a float `discount\_rate`. However, on the third iteration we see the following:



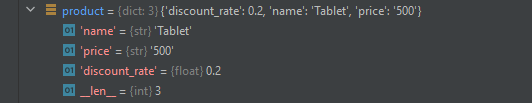
Here, we are attempting to multiply a float with a character sequence (string), which results immediately in the error we’re tracking down.

We can see that `price` is the variable containing the undesired type. If we follow the flow of data back to where `price` is assigned a value, we see the following:



The value of `price` is not mutated between where it was first assigned a value and where it was passed to the function `calculate\_discount`. Therefore, the undesired value of `price` comes from one of the dictionaries, `product`, contained within the list `products`.

We can search the `products` list manually in the code or we can add a break point and step through the code until we see the key `’price’` containing a string value rather than an integer value:



Here, the faulty value is found, and we see that it belongs to the entry for `’Tablet’`. To fix the error, we fix the value:

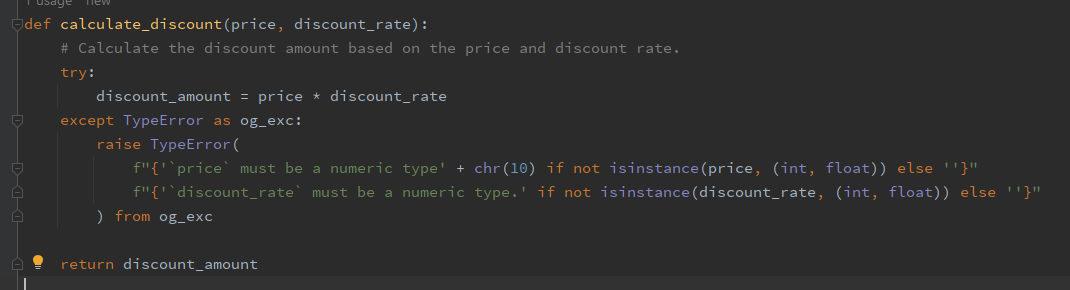




## Improve Error Handling

To catch and handle this error with more meaningful error messaging in the future, we can surround the code block that had previously errored with a try-catch block.

We can take it a step further by determining which one or both variables contains the wrong type by testing and supplementing the default `TypeError` message with a more explicit cause:



*(Note: `chr(10)` is used because, prior to Python 3.12, escape sequences can’t be used inside the expression part of an f-string.)*

## Testing

When testing with the correct data types, the code functions as expected:



When intentionally introducing bad types into the data, we get a cleaner and more precise cause of the error:

