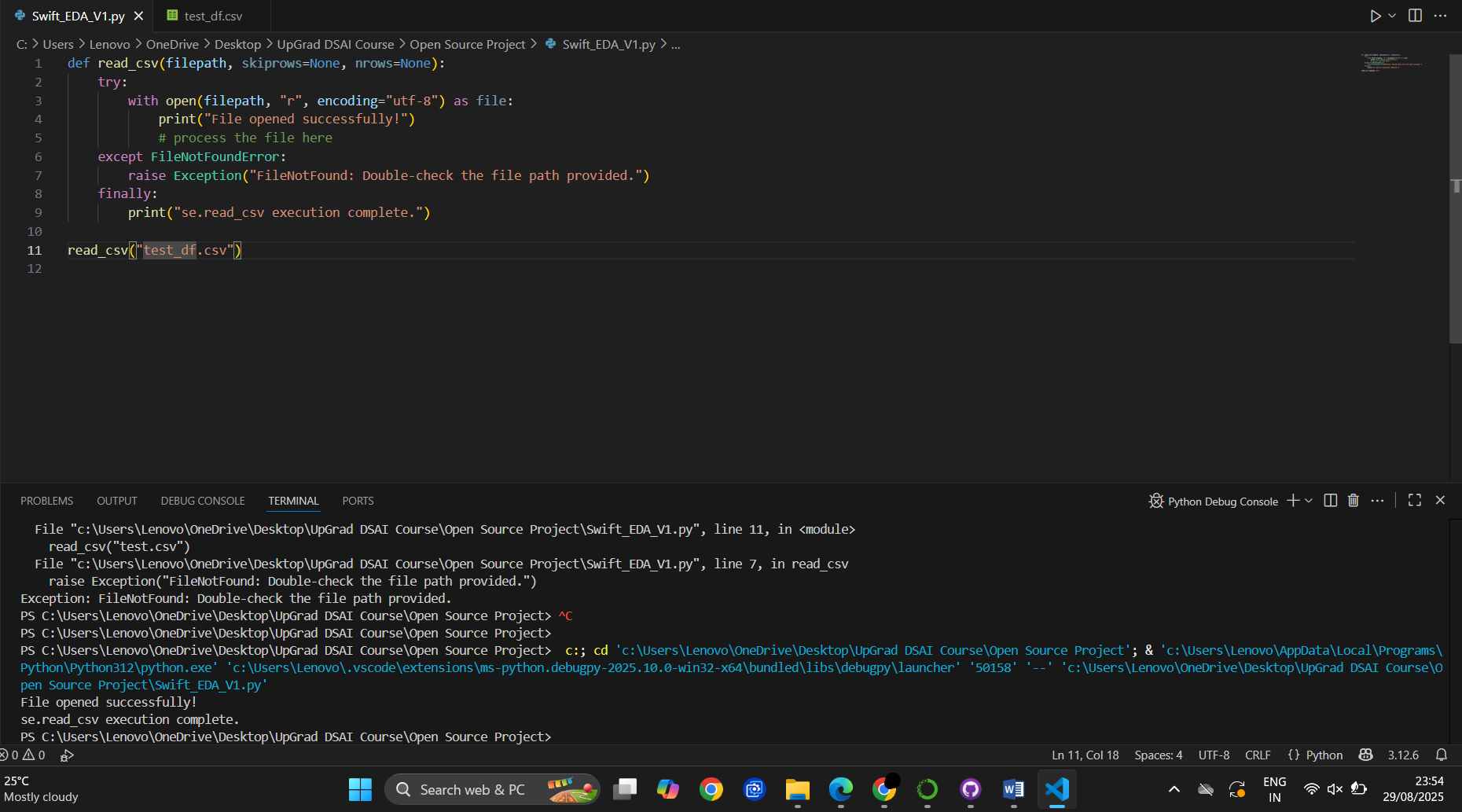
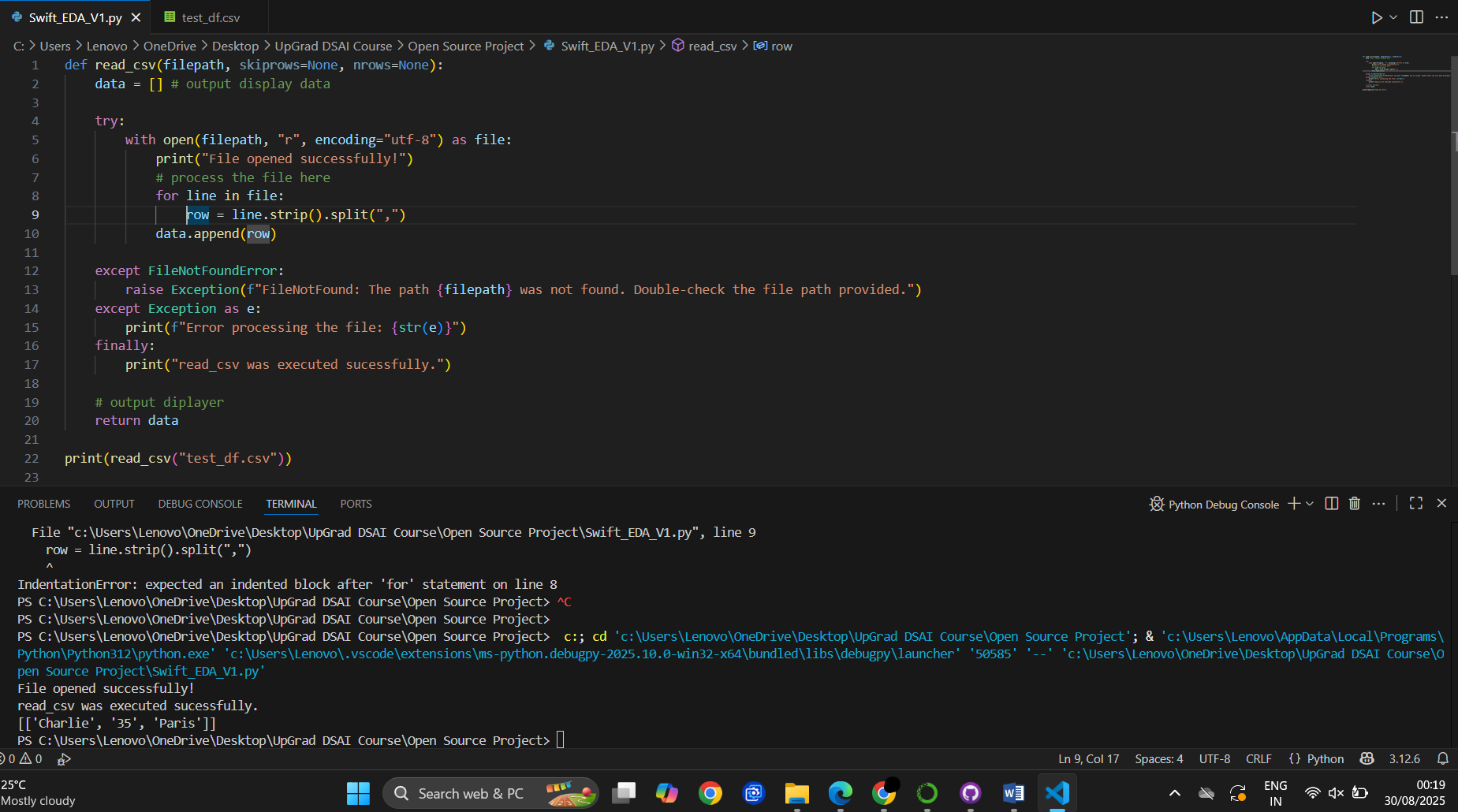


Need to rename error message & add file processing error message

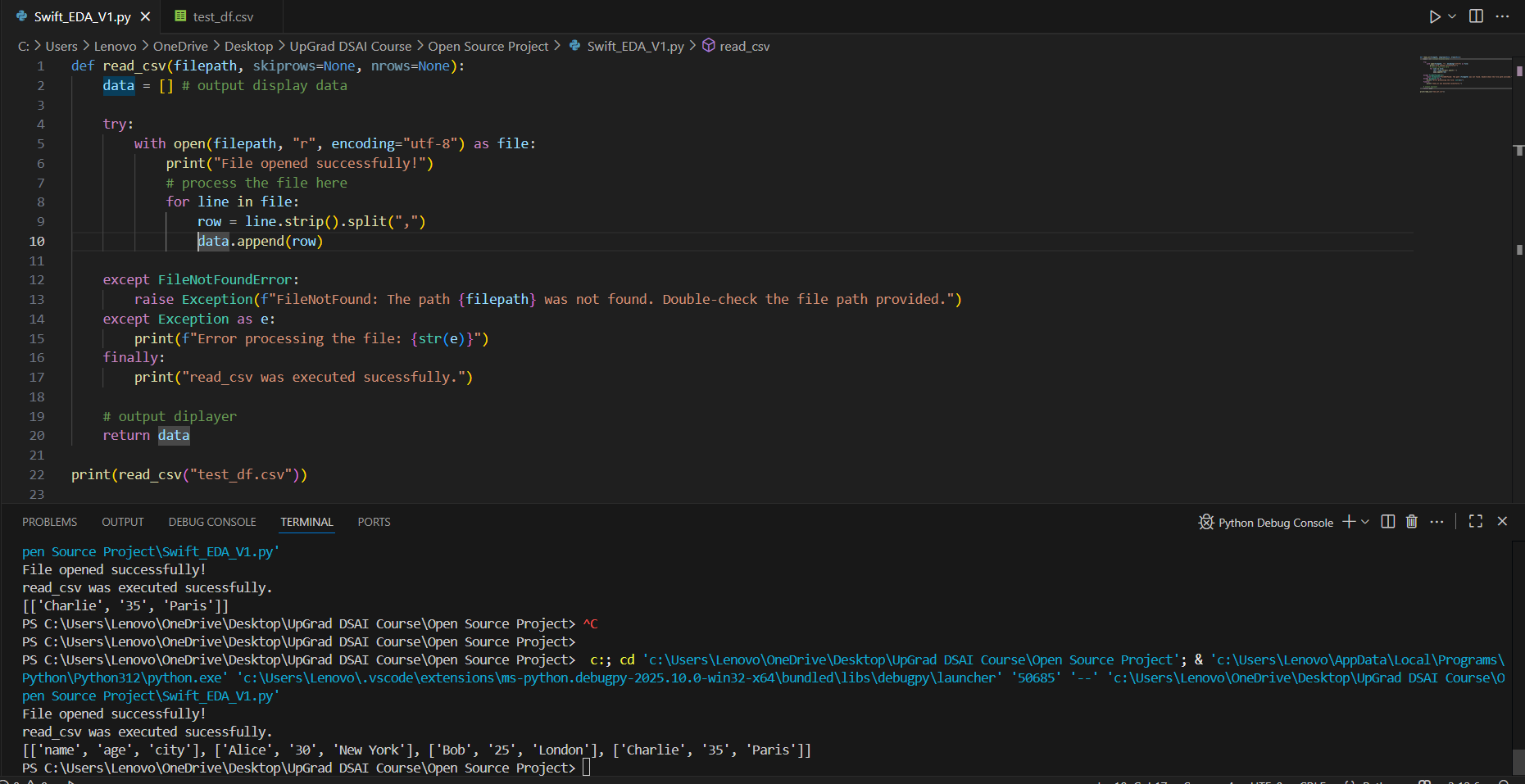


Basic reading works

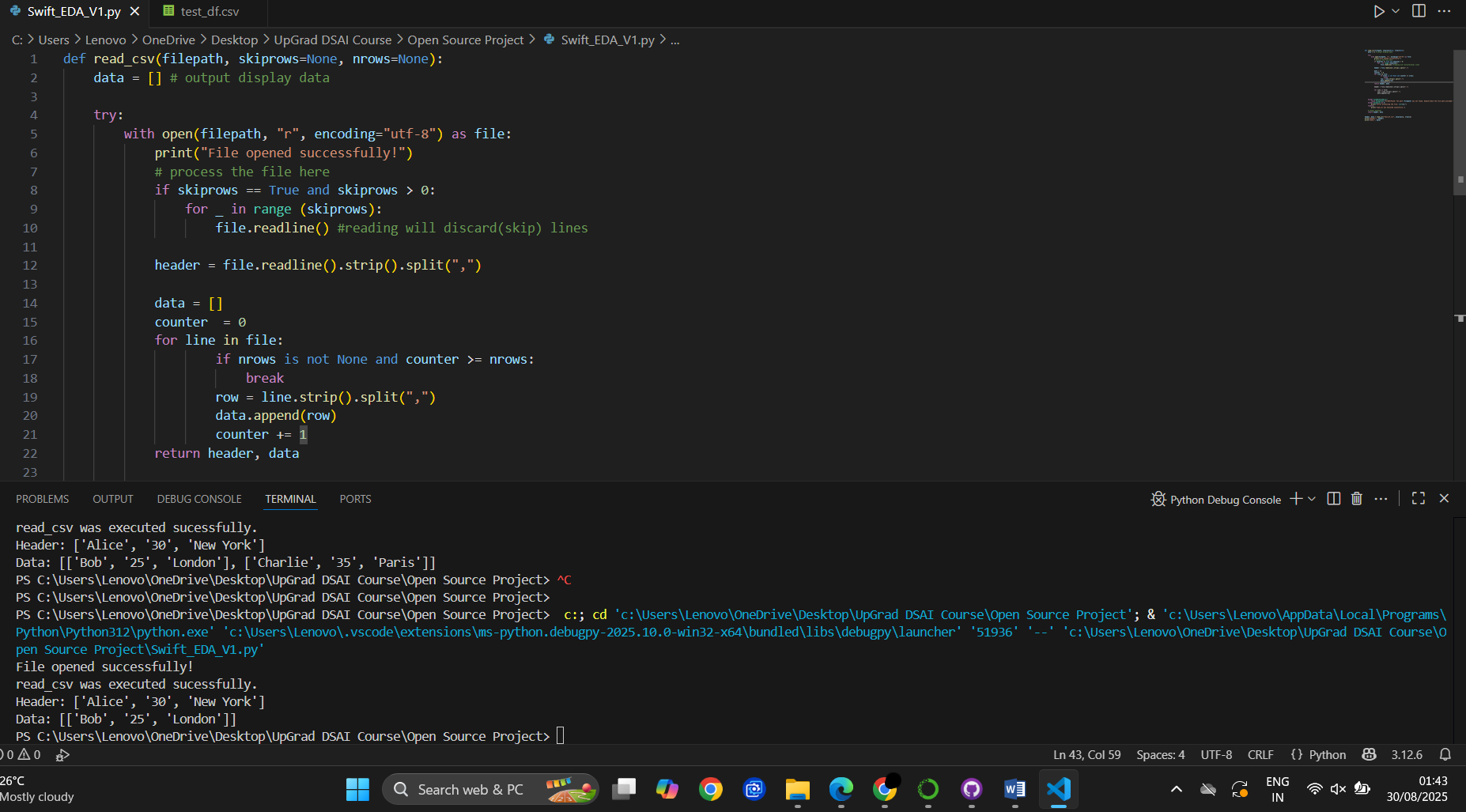


Displays the last line in the csv file as output

Either the data is being overwritten by return data for the latest change (since we didn't give it any structure to follow) or there is a processing error in the read\_csv stage (which seems unlikely)



Fixed: Indentation error in processing line caused overwriting if fixed



Added skiprows & nrows

def read\_csv(filepath, skiprows=None, nrows=None):

    data = [] # output display data

    try:

        with open(filepath, "r", encoding="utf-8") as file:

            print("File opened successfully!")

            # process the file here

            if skiprows == True and skiprows > 0:

                for \_ in range (skiprows):

                    file.readline() #reading will discard(skip) lines

            header = file.readline().strip().split(",")

            data = []

            counter  = 0

            for line in file:

                    if nrows is not None and counter >= nrows:

                        break

                    row = line.strip().split(",")

                    data.append(row)

                    counter += 1

            return header, data

            header = file.readline().strip().split(",")

            for line in file:

                row = line.strip().split(",")

                data.append(row)

    except FileNotFoundError:

        raise Exception(f"FileNotFound: The path {filepath} was not found. Double-check the file path provided.")

    except Exception as e:

        print(f"Error processing the file: {str(e)}")

    finally:

        print("read\_csv was executed sucessfully.")

    # output diplayer

    return header, data

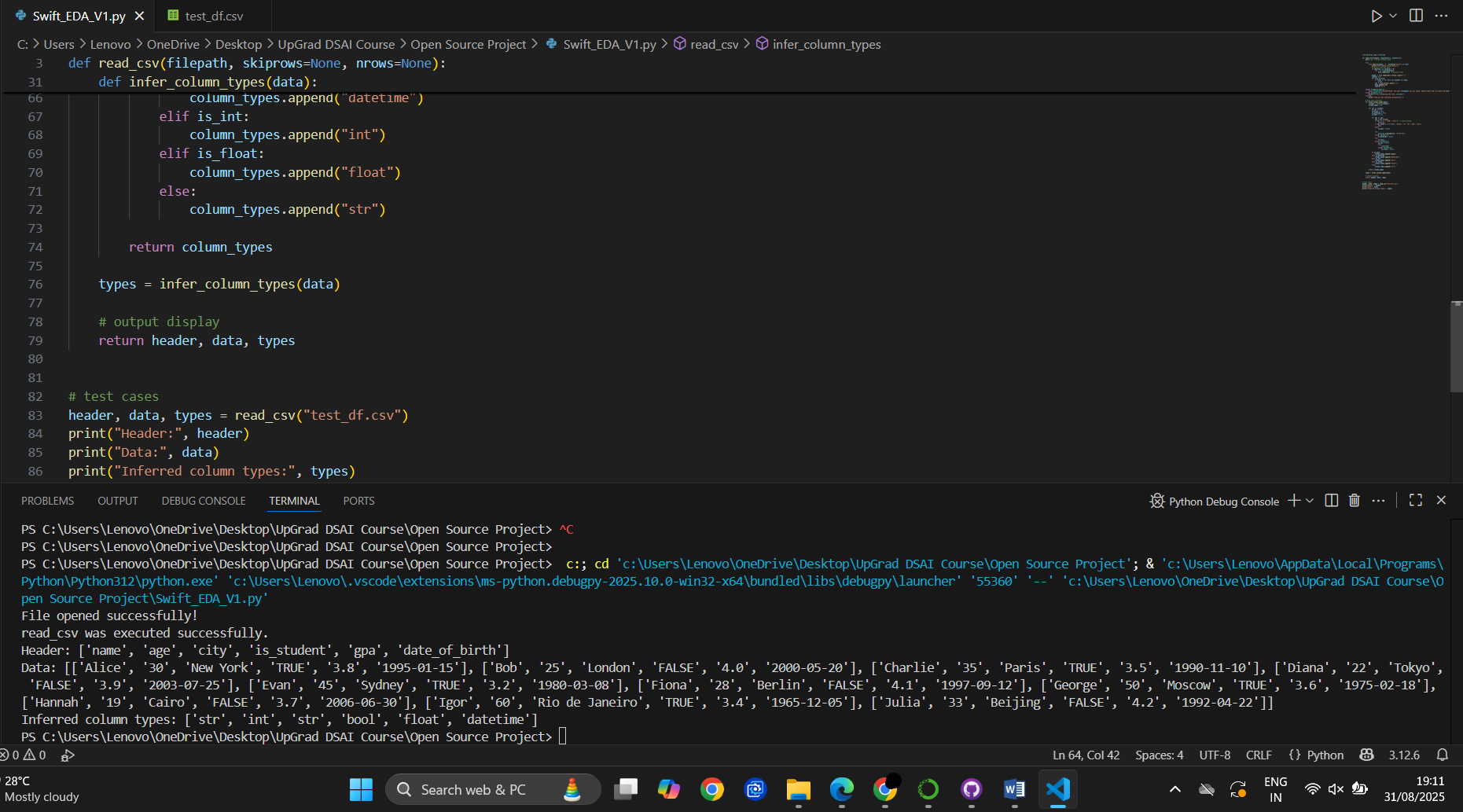
header, data = read\_csv("test\_df.csv")

print("Header:", header)

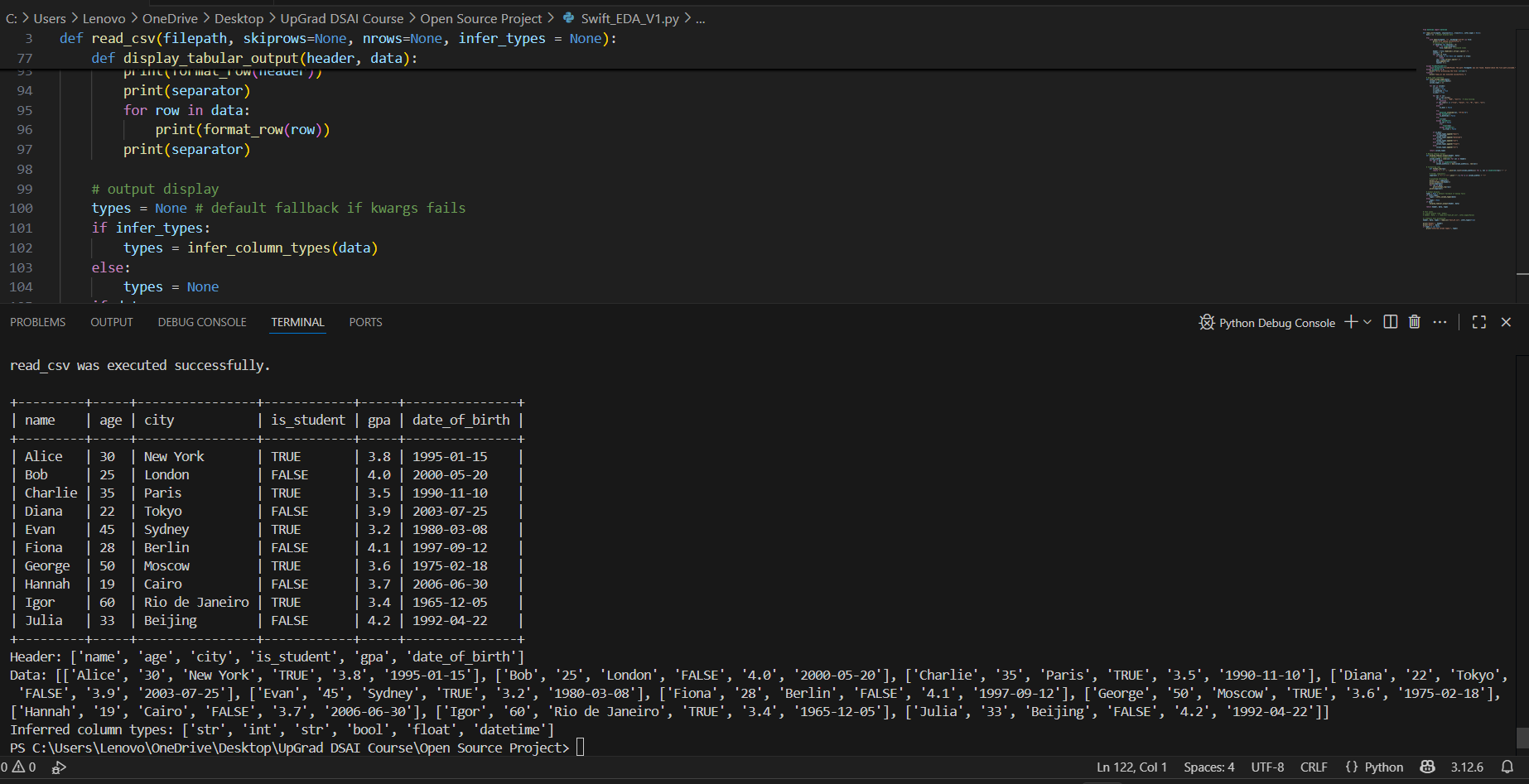
print("Data:", data)

Next to-do list:

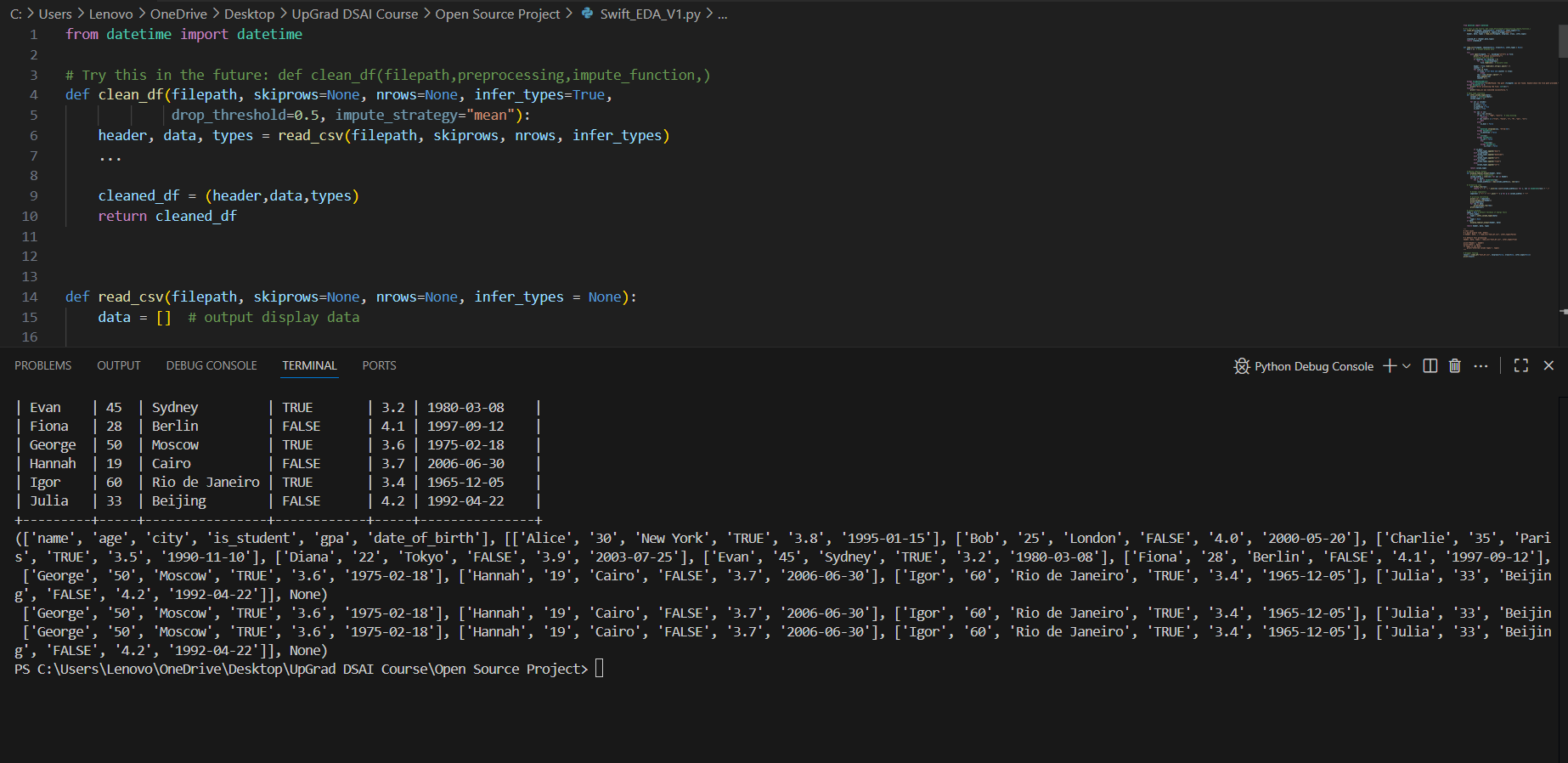
|  | Step |  |  | Feature | Description |
| --- | --- | --- | --- | --- | --- |
|  | 1 |  |  | Data Type Inference | Detect and convert values (int, float, str) |
|  | 2 |  |  | Display as Table | Format output like a table — clean column alignment |
|  | 3 |  |  | Edge Case Handling | Malformed rows, empty lines, column mismatches |
|  | 4 |  |  | Refactor | Clean up logic, remove duplication, prepare for reuse |
|  | 5 |  |  | Optional Enhancements | Delimiter support, header override, column filtering, etc. |



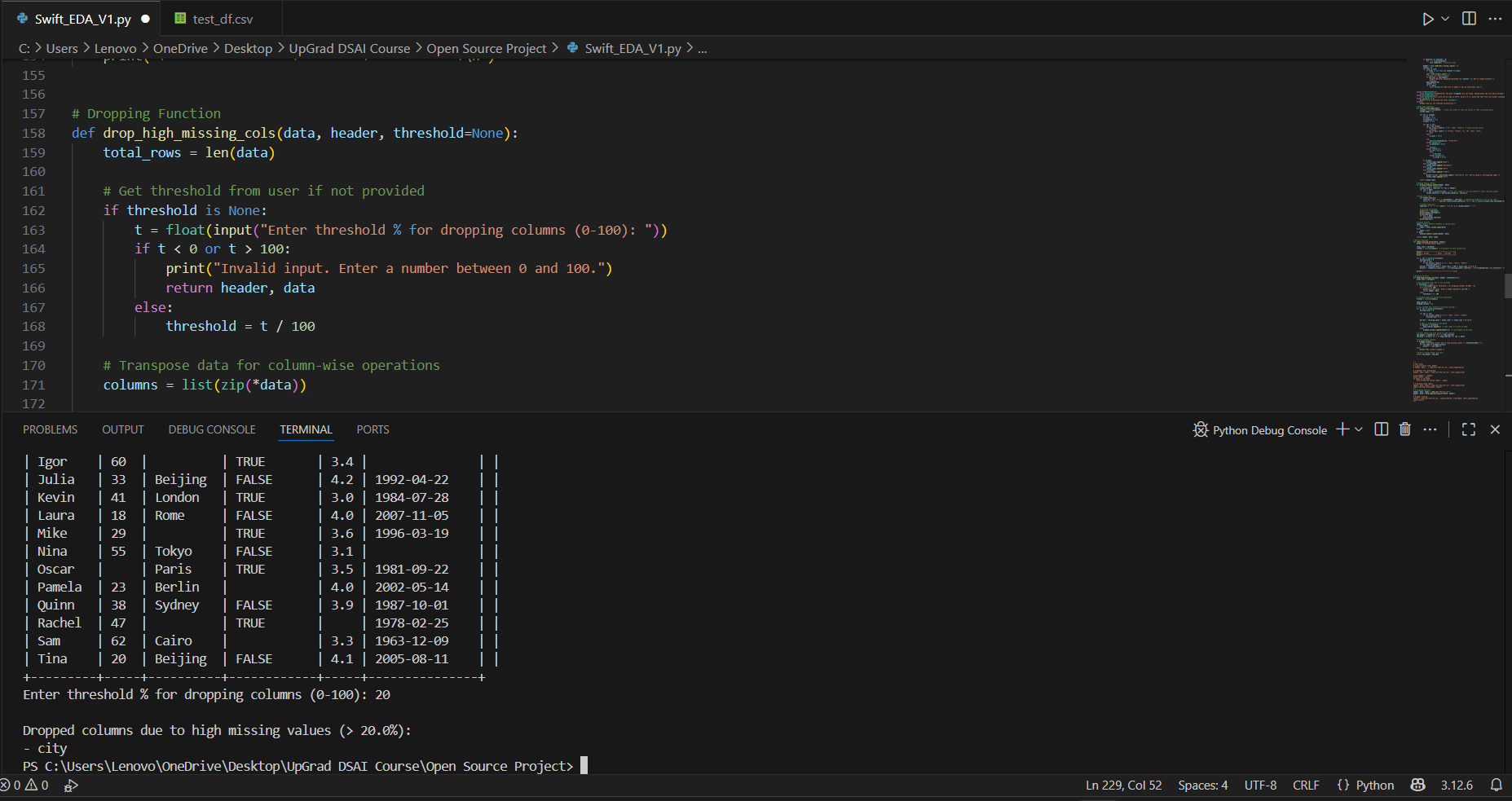
**Added data type parsing NOTE: add type casting in future versions**

****

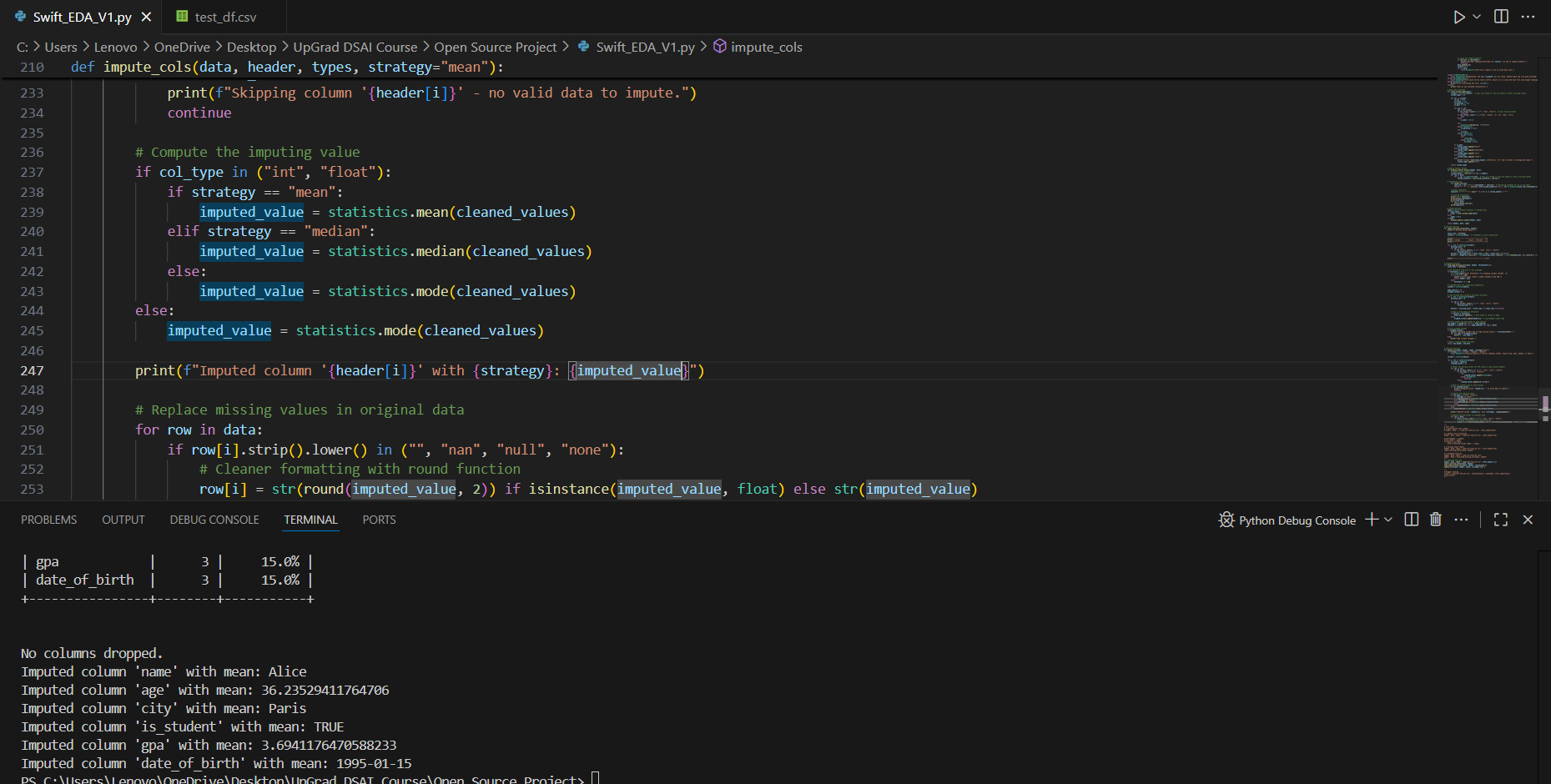
Tabular display & Data type inference works perfectly



|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  | Missing Value Report Generator works fine |  |  |



Drop missing values function works fine  
**NOTE: Values are dropped from the memory not the csv file since they’re stored as variables. Need to build write\_csv function using os & file reading**

****

**impute\_cols is filling in with strings.**

**To keep internal data in native types (int/float) for analysis instead of CSV-style strings, we’ll need to change how data is stored and handled — which affects read\_csv(), display\_tabular\_output() and more**

| **Feature** | **Description** |
| --- | --- |
| ✅ **Per-column strategy** | Let users specify different strategies per column (optional for now). |
| 📋 **Log what was imputed** | Print a summary: “Imputed column ‘age’ with mean: 32.4”. |
| 🔎 **Detect outliers** | Replace missing values only if the computed value isn’t an outlier (future phase). |
| 💾 **Preserve types** | After imputation, convert back to int, float, bool (not all strings). |

|  |  |
| --- | --- |
| Features | Complete |
| Custom read\_csv() function | Complete |
| skiprows, nrows arguements | Complete |
| Header parsing | Complete |
| Datatype inference | Complete |
| Tabular ouput format | Complete |
| Infer\_types for controlled d-type | Complete |
| Wrapper function | Complete & Testable |

**Phase 1 Completed Successfully**

**Phase 1 Edge Cases:**  
Line 43: zip will break if rows are uneven, empty & have missing values

Line 92: enumerate will break if rows are uneven & have missing values

Both are row & col length mismatch & can be solved by introducing warnings & validation checks (quick fix for missing data exploration)

Overall code will fail if data structure is uneven (ex: one has str, int, float, bool & other has str, float, bool, str, that won’t be logged)

Future problem: Unicode errors (we only account for UTF-8) & incorrect formats used (xls provided while read\_csv is used):

Fix by introducing a validation check & reuse whole file reading block for xls & json formats

Allow flexibility in date format (can be postponed)

Case sensitivity in NaN & null: Fix by standardizing list of NaNs (NaN, nan, NULL .etc)

Account for unstructured data

**To keep internal data in native types (int/float) for analysis instead of CSV-style strings, we’ll need to change how data is stored and handled — which affects read\_csv(), display\_tabular\_output() and more**

**Phase 1 Patch Notes:**

Fixed row length mismatch with header length in processing section

Added UnicodeDecodeError (Temporary fix only)

Added csv file format check

Added check for empty and invalid CSV files after appending to ensure downstream processing

Added strip().lower() function chain for NaN & null variables

Amended formatting rows for rows that are too small

Typecasting & print statements for Impute function were rectified

display\_tabular\_output function casts all values as strings for neat tabular display

**Real World Test Case Report:**

**Test Results on Titanic dataset:**  
Pros:

* Finished processing in less than a second
* No bugs encountered

Cons:

* Output eats up a lot of terminal & is hard to interpret, need a function like .head()

Future possibility after finishing all Phases: Train a ML model to make sensible imputes

**Phase 2**

Add multi format support for JSON & Excel reading (refractor CSV reader & update wrapper to accept kwargs) Build a failsafe for wrong format input

Add limiter for concise output (df.head() in pd)

Create/enhance summary stats function to show min, max, mean, median, mode, percentiles & IQR (better to create it as a helper function instead of wrapper arg function)

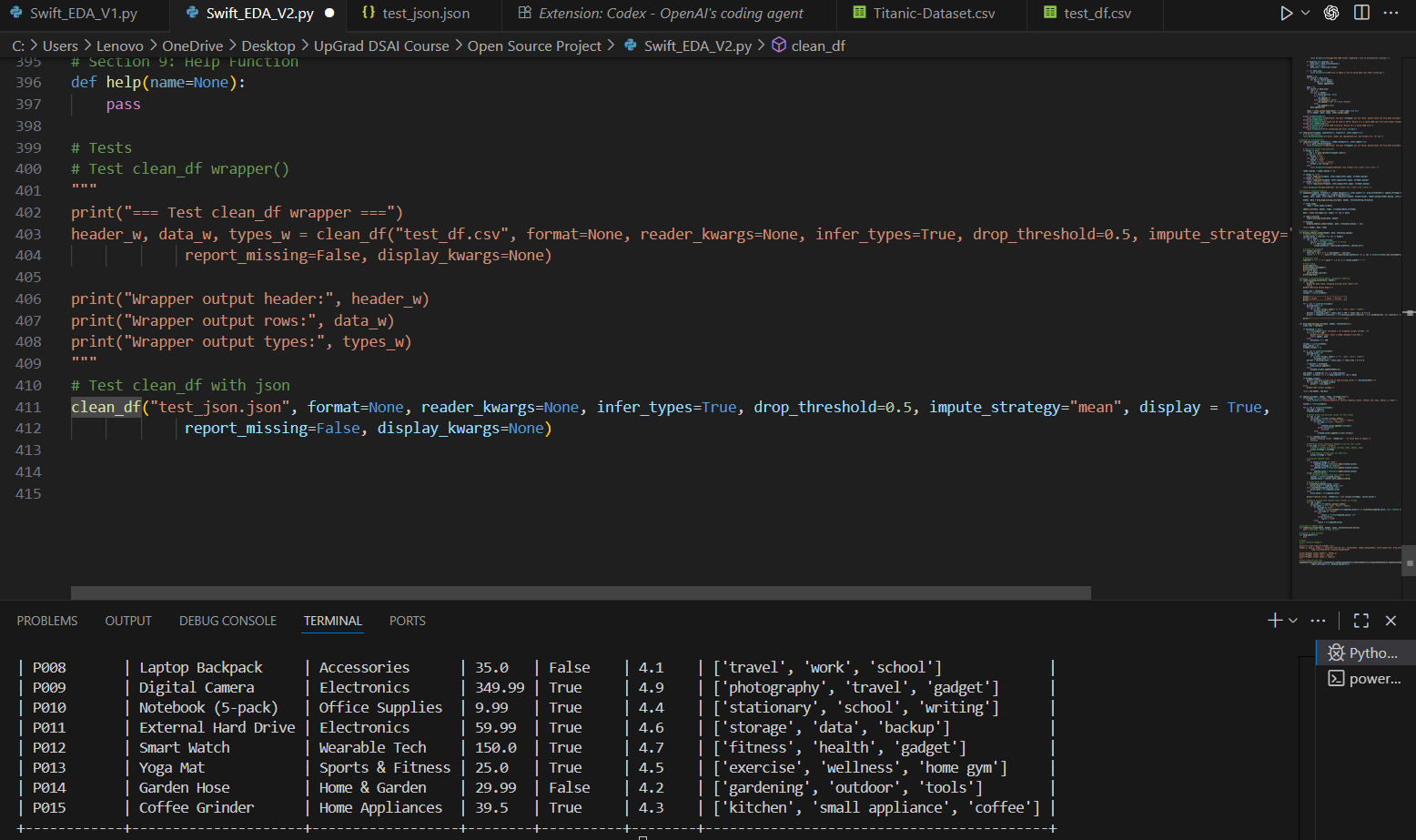
Create help() for every function after implementing major Phase 2 updates

Low priority:

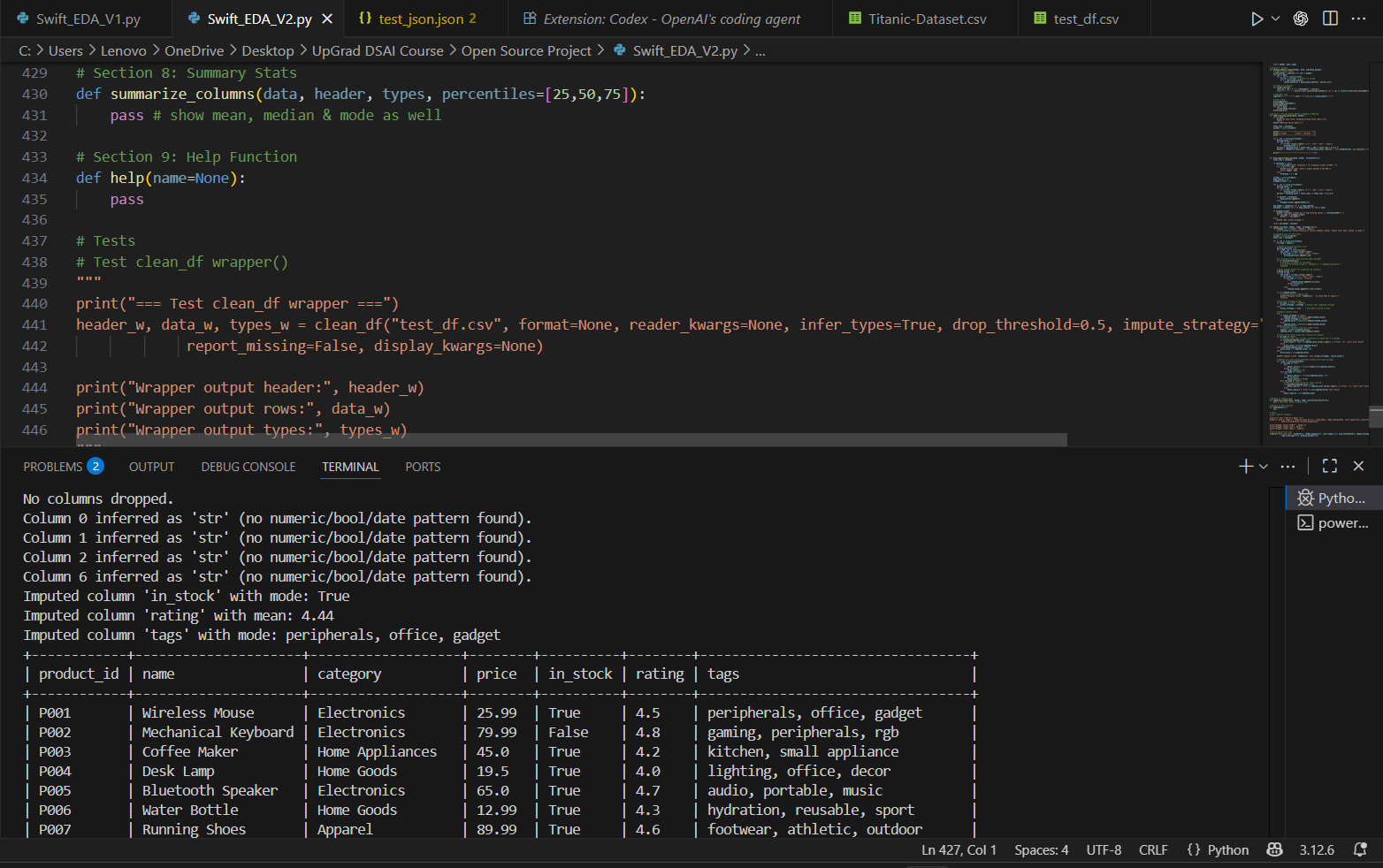
Type-casting (can be added in V3 along with visualizations)

Consolidate all print statements into a centralized logging system

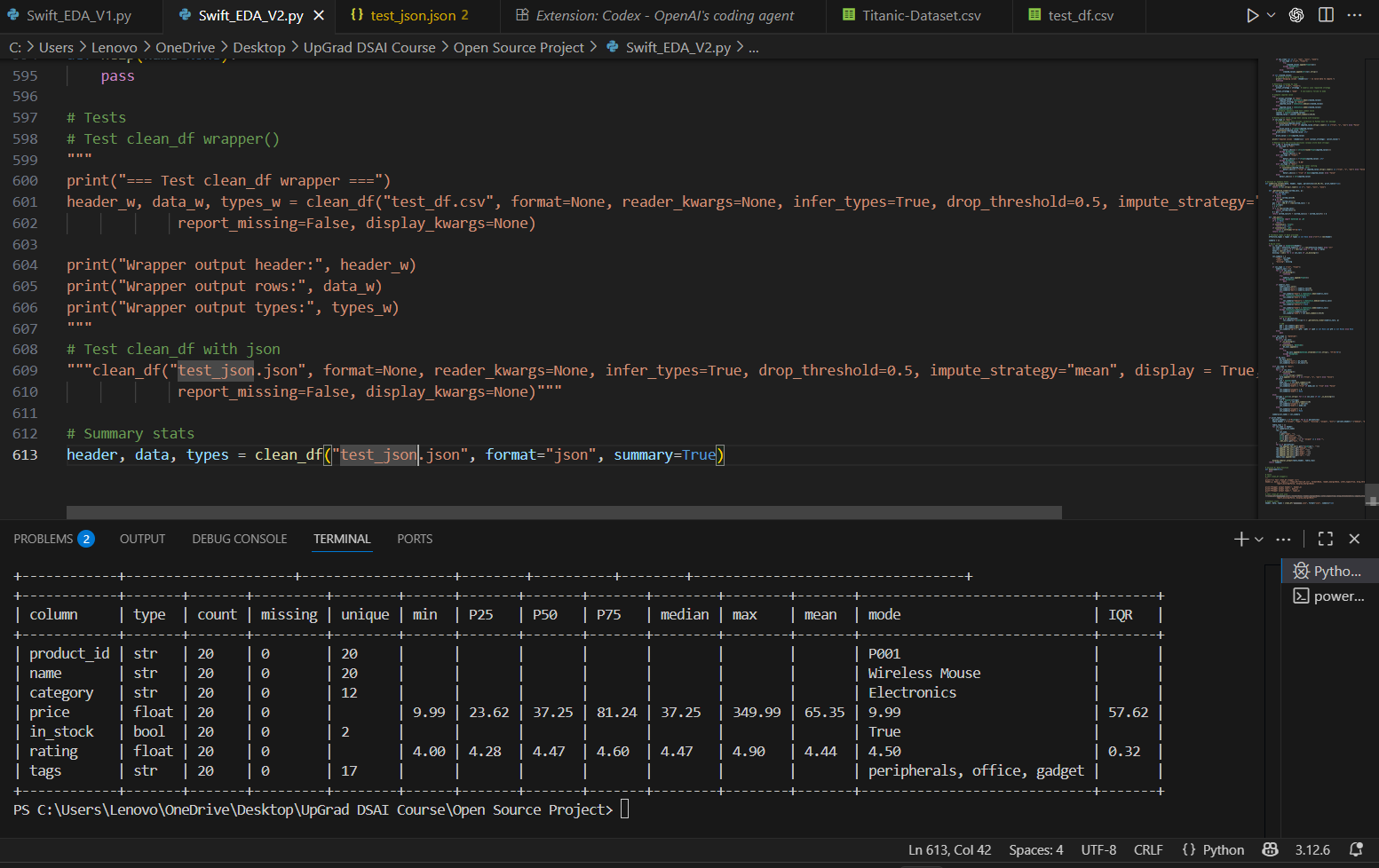
Test for edge cases

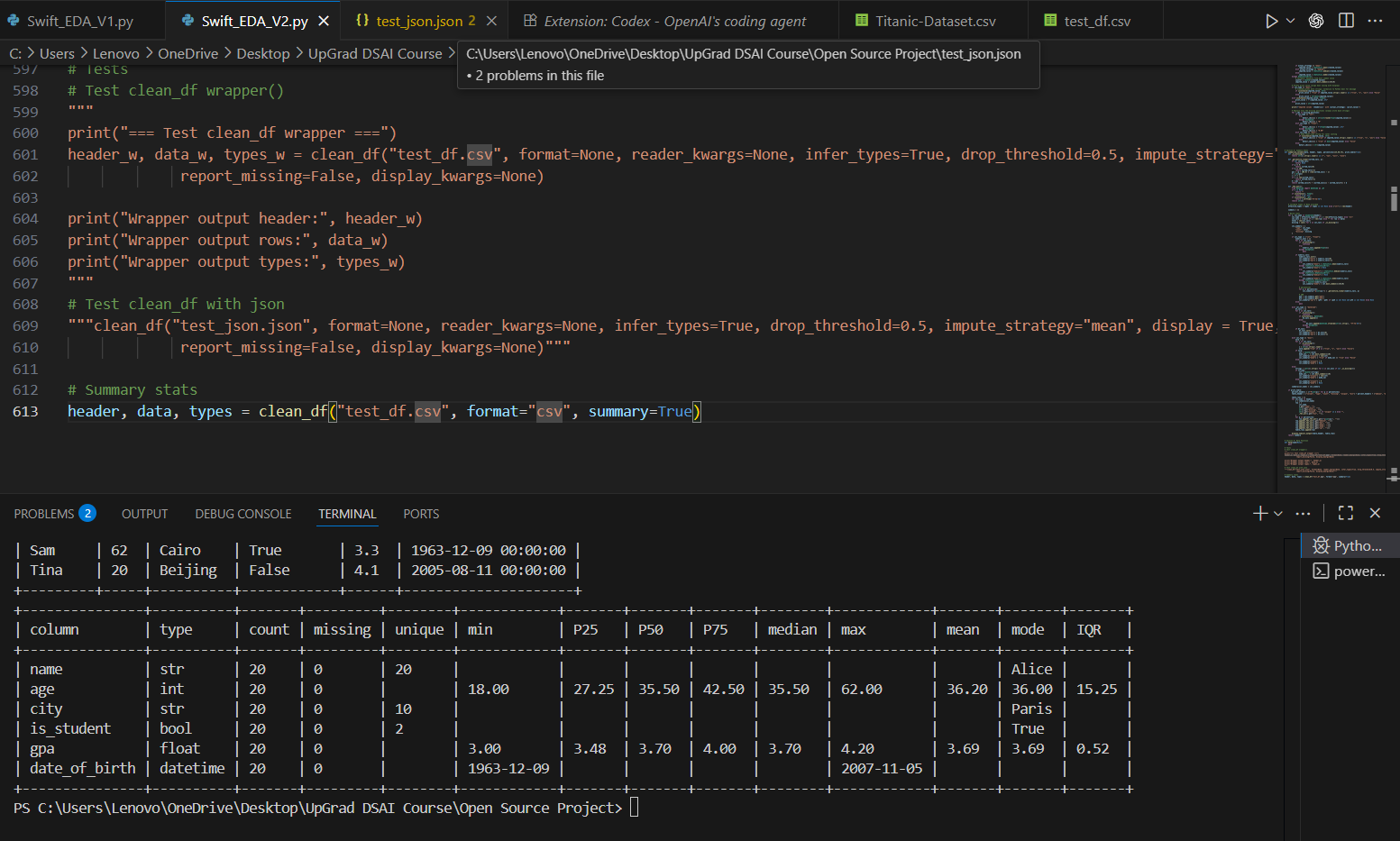


JSON file reads well but output is misleading



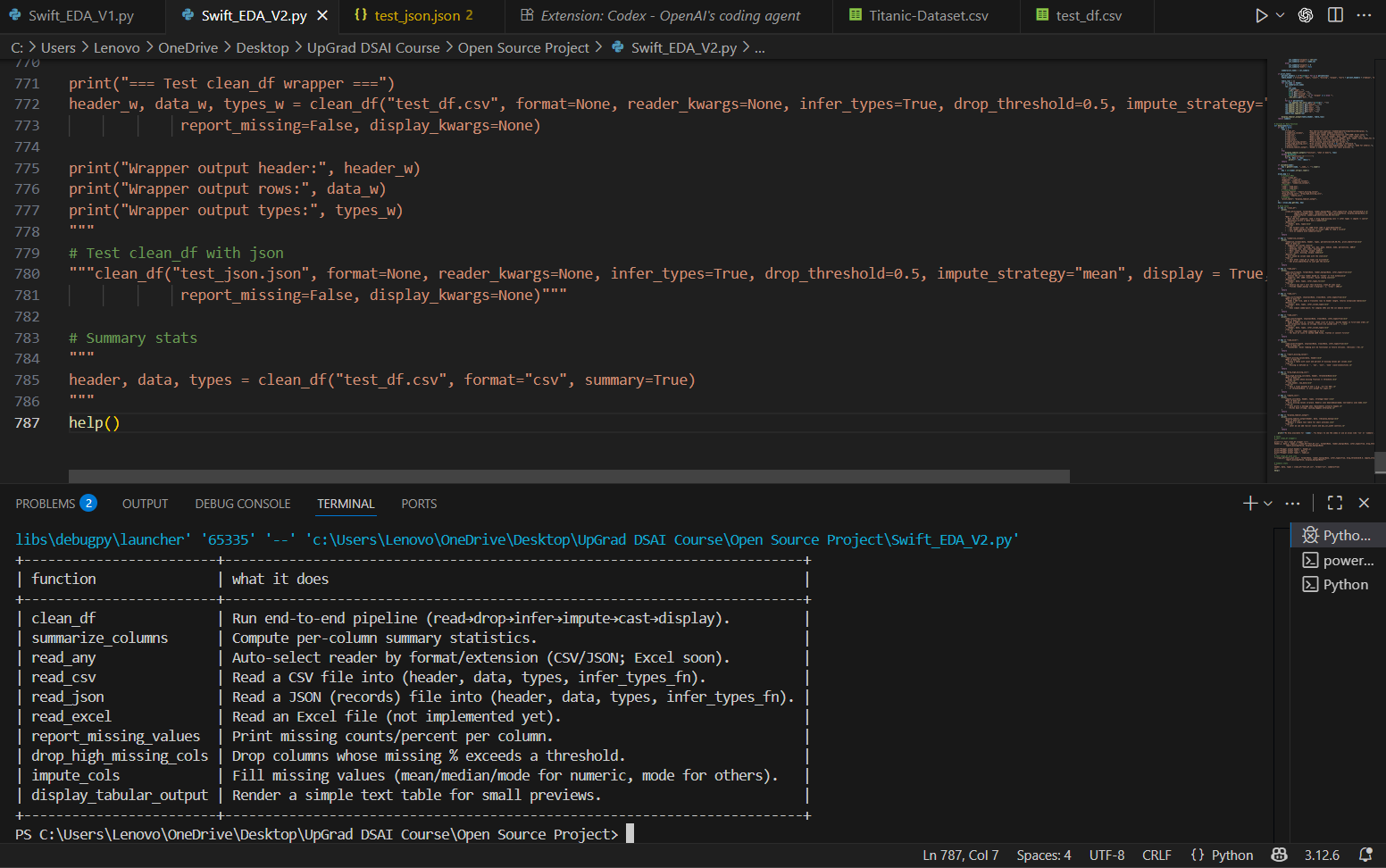
Fixed by making print statements clearer





summarize\_columns replicates describe function perfectly for both csv & json

NOTE: always call it after calling clean\_df



help() works fine

**V2 Patch Notes:**

* Refractored file processing by separating functions.
* Introduced json processing & created a centralized input file processor read\_any to check file extension & streamline automation to reduce file type & user errors
* Introduced percentiles & IQR in column summary along with mean, median & mode
* Tabulated summary statistics
* help() is now functional
* Need to add max\_col\_width control to prevent tabular output flooding the terminal
* Removed read\_excel function due to dependencies on openpyxl

Phase 3 notes:

Add a central log function for verbosity removing inconsistent print statements