Unit 4: Data Cleaning and Transformation

Overview

In this unit, I explored what it takes to prepare raw data for meaningful analysis. We broke down each stage of the **data management pipeline**, understanding how data flows from initial collection to the final point of analysis and visualisation. The focus was on how to clean, structure, and transform data in a way that ensures reliability and usability across systems.

Key Learning Areas

1. **Data Cleaning Techniques**: I learned to handle missing values, outliers, and formatting issues by identifying bad data, matching inconsistent strings, and spotting structural anomalies (Kazil & Jarmul, 2016).
2. **The Data Management Pipeline (EMC, 2015):** Using the EMC (2015) model, I mapped out the full process—capturing, cleaning, integrating, designing databases, analysing, and presenting data—while noting how early mistakes can affect final insights.
3. **Automating Data Processes**: We discussed how to automate workflows using tools like Pandas and NumPy to reduce manual steps and improve efficiency.

Additional Concepts Covered

1. **Data models vs Data Architecture**:

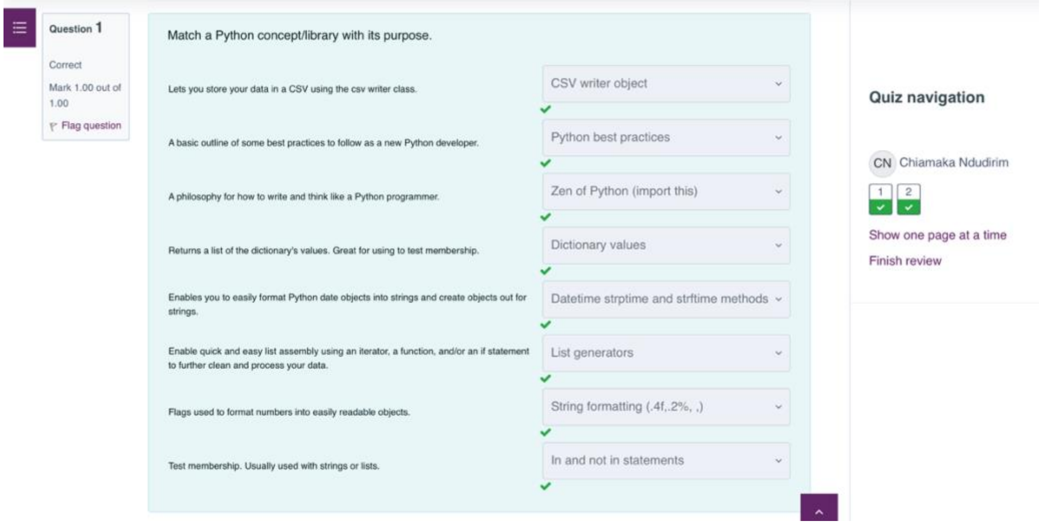
I learned to distinguish between data models (which define relationships between data elements) and data architecture (which focuses on how data is captured, organised, and structured across a system).

2. **Python Tools in Practice**:

As part of our formative activity, I practiced matching key Python concepts and libraries—such as Pandas and NumPy—to real data tasks like cleaning, transformation, and validation.

Formative Activity

In the **formative activity**, I matched Python libraries to data handling tasks, reinforcing practical applications of cleaning and transformation techniques.



Question 1

Correct

Mark 1.00 out of 1.00

Flag question

Match a Python concept/library with its purpose.

Lets you store your data in a CSV using the csv writer class.
CSV writer object ✓

A basic outline of some best practices to follow as a new Python developer.
Python best practices ✓

A philosophy for how to write and think like a Python programmer.
Zen of Python (import this) ✓

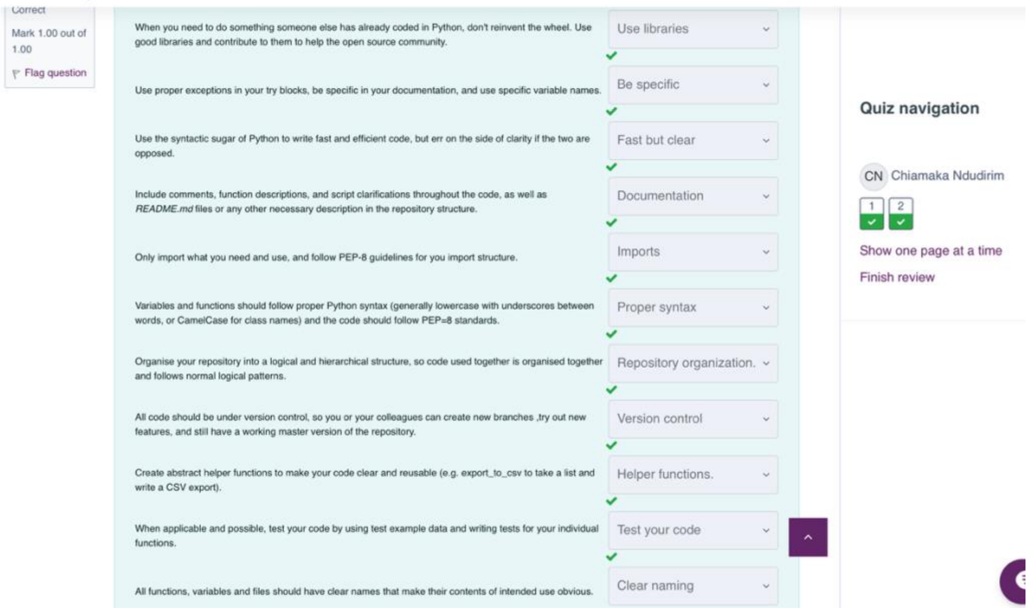
Returns a list of the dictionary's values. Great for using to test membership.
Dictionary values ✓

Enables you to easily format Python date objects into strings and create objects out for strings.
Datetime strptime and strftime methods ✓

Enable quick and easy list assembly using an iterator, a function, and/or an if statement to further clean and process your data.
List generators ✓

Flags used to format numbers into easily readable objects.
String formatting (.4f,.2%, ,) ✓

Test membership. Usually used with strings or lists.
In and not in statements ✓



Correct

Mark 1.00 out of 1.00

Flag question

When you need to do something someone else has already coded in Python, don't reinvent the wheel. Use good libraries and contribute to them to help the open source community.
Use libraries ✓

Use proper exceptions in your try blocks, be specific in your documentation, and use specific variable names.
Be specific ✓

Use the syntactic sugar of Python to write fast and efficient code, but err on the side of clarity if the two are opposed.
Fast but clear ✓

Include comments, function descriptions, and script clarifications throughout the code, as well as *README.md* files or any other necessary description in the repository structure.
Documentation ✓

Only import what you need and use, and follow PEP-8 guidelines for you import structure.
Imports ✓

Variables and functions should follow proper Python syntax (generally lowercase with underscores between words, or CamelCase for class names) and the code should follow PEP=8 standards.
Proper syntax ✓

Organise your repository into a logical and hierarchical structure, so code used together is organised together and follows normal logical patterns.
Repository organization. ✓

All code should be under version control, so you or your colleagues can create new branches ,try out new features, and still have a working master version of the repository.
Version control ✓

Create abstract helper functions to make your code clear and reusable (e.g. export_to_csv to take a list and write a CSV export).
Helper functions. ✓

When applicable and possible, test your code by using test example data and writing tests for your individual functions.
Test your code ✓

All functions, variables and files should have clear names that make their contents of intended use obvious.
Clear naming

Personal Reflection

This unit helped me see how much work goes into preparing data before analysis begins. I now understand how crucial cleaning and structure are to producing meaningful insights and feel more equipped to manage real-world datasets confidently.

References

EMC Education Services (2015) *Data Science and Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data*. Indianapolis: Wiley.

Kazil, J. and Jarmul, K. (2016) *Data Wrangling with Python: Tips and Tools to Make Your Life Easier*. 1st edn. Beijing: O'Reilly Media.