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**:

   We looked at how to deal with inconsistencies, outliers, missing values, and formatting issues that can distort insights. As Kazil & Jarmul (2016) suggest, this includes:

   - Identifying bad data and outliers
   - Vetting the reliability of sources
   - Matching inconsistent strings
   - Spotting patterns or anomalies that affect structure

2. **The Full Data Management Pipeline (EMC, 2015):**

I learned the step-by-step process involved in managing data effectively:

- Capturing raw data
- Data cleaning
- Data integration
- Database design
- Data analysis
- Data presentation and visualisation

Each step feeds into the next, and errors early in the pipeline can carry through and affect final outcomes, making the cleaning and integration stages especially critical.

3. **Automating Data Processes**:

We also covered what it takes to automate data workflows—using tools, libraries, and consistent structure to reduce manual handling and improve efficiency.

Additional Concepts Covered

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

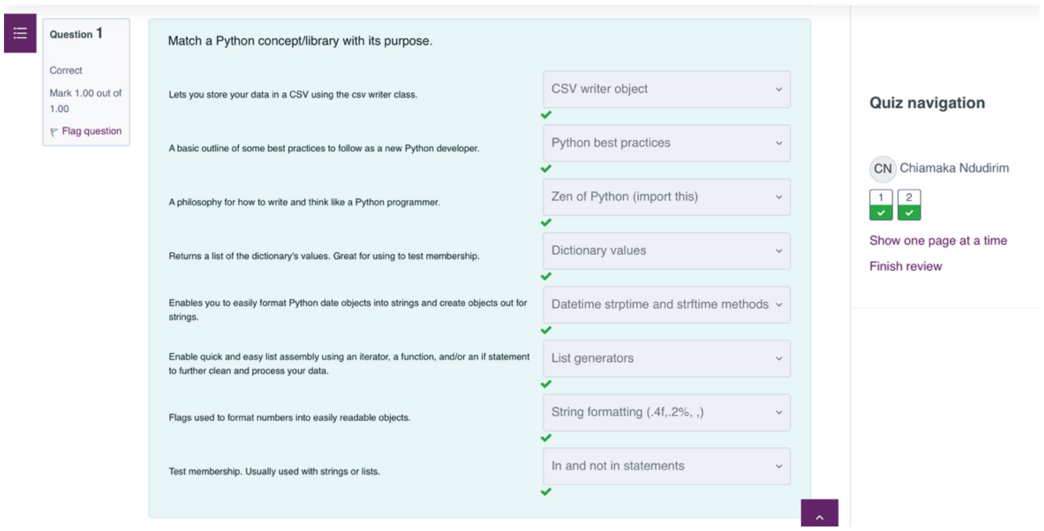
I learned to distinguish between data models (which define relationships between data elements) and data architecture (which focuses on how data is captured, organised, and structured across a system).

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

We had a formative activity to complete a Data Management pipeline test. Instructions were to match python concepts or libraries to their purposes and to match python practices with the suitable description, as seen below.



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 ✓

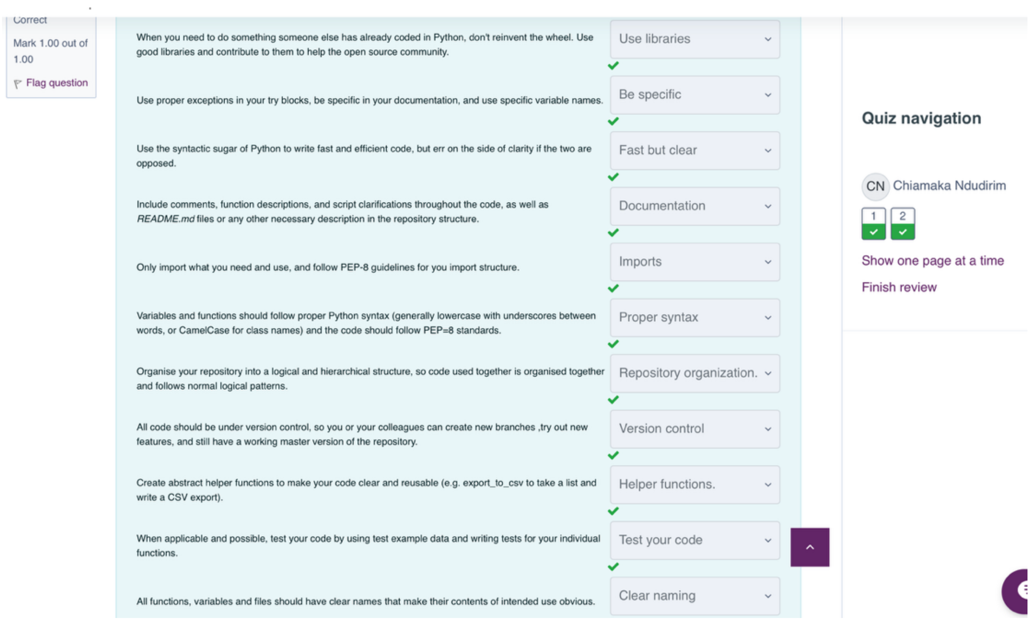
Quiz navigation

CN Chiamaka Ndudirim

1 ✓ 2 ✓

Show one page at a time

Finish review


Correct

Mark 1.00 out of 1.00

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

Quiz navigation

CN Chiamaka Ndudirim

Personal Reflection

This unit helped me connect the dots between raw data and final analysis. It became clear that the way data is prepared behind the scenes is just as important as the analysis itself. With a better understanding of the data pipeline and how to apply structured cleaning techniques, I feel more confident in managing real-world data problems—especially when scaling or automating processes.

Core Readings

- McKinney, W. (2022) Python for Data Analysis: Data Wrangling with Pandas, NumPy, and Jupyter. 3rd edn. Sebastopol, California: O'Reilly.
    - Chapter 7
- Huxley et al. (2020) Data Cleaning. Sage Foundation.