Title: Data Structures

You will now become familiar with data structures, which are approaches to organise, store and manage electronic data. To do this, visit python.org (n.d.) and any other relevant sources of information that you may find useful to support this task.

- Select at least two different data structures to hold the data associated with the list of functional and non-functional requirements that you defined for Task 1.
- Justify your data structure choices.
- Select at least one academic paper, which might be similar to the work of Abeykoon et al. (2020).
- Use your sourced information to support your data structure choices.

I have used lists and dictionaries to store the data related to the functional and non-functional requirements list for Task 1, 'e-Portfolio Activity: Data Structures Reflection'. As I mentioned, these data structures are theoretical concepts essential to the ZenDesk system I use daily. The practical application of lists and dictionaries in a real-world system underscores their importance and relevance, as highlighted by Loshin (2021).

According to Olumide (2023), Python's built-in data type lists store an ordered collection of mutable elements and can contain multiple data types. They include methods such as append, insert, remove, pop, sort, reverse, count, index, and extend. Lists are created by separating elements with commas and enclosing them in square brackets. They help keep several connected data sets in a single list. However, making a list of lists can be challenging due to incorrect initialisation. One way to initialise a nested list is by using list comprehension and range().

On the other hand, Python's built-in dictionary is a valuable data type that stores key-value pairs. The keys can be of any immutable kind and must be unique. A dictionary can perform various operations, such as storing and retrieving values and deleting keys. If you want a list of all the keys used in the dictionary, you can use the list(d) method (Python, N.D.).

As far as I can understand, the dictionary is a superior tool for organising functional and non-functional requirements into key-value pairs (Ross, 2023). Its flexibility allows users to define new key-value combinations, adding additional requirements quickly. Moreover, dictionary values can accommodate different data types. While the list of lists has its merits, its adaptability is limited, and it requires restructuring the main structure to add new categories or requirement types. Hence, I maintain my decision to use a dictionary as the primary data structure for this task.

I came across an article on Google Scholar that aligns with the research conducted by Abeykoon et al. (2020). The article by Rashid et al. (2020) discusses the importance of data dictionaries as a centralised repository of descriptive information about data sets. It highlights their role in aiding data providers in tasks such as conversion, validation, and storage and in reducing ambiguity when interpreting data set content. The article also points out that the lack of a common metadata standard for existing data dictionaries makes automating tasks involving data combinations challenging. It suggests that emerging Semantic Web technologies can enhance the functionality of data dictionaries.

References:

Bader, D. (2020) *Common Python Data Structures (Guide) – Real Python. [online] realpython.com.* Available from*:* https://realpython.com/python-data-structures/ [Accessed 8 May 2024].

Loshin, D. (2021) *What are Data Structures? - Definition from WhatIs.com*. [online] SearchDataManagement. Available from: https://www.techtarget.com/searchdatamanagement/definition/data-structure [Accessed 8 May 2024].

Olumide, S. (2023) *List Within a List in Python – How to Initialize a Nested List*. [online] Available from: https://www.freecodecamp.org/news/list-within-a-list-in-python-initialize-a-nested-list/ [Accessed 8 May 2024].

Python (N.D.) *5. Data Structures — Python 3.9.0 documentation*. [online] Available from: https://docs.python.org/3/tutorial/datastructures.html#dictionaries [Accessed 6 May 2024].

Rajagopal, A. (2023) *Demystifying Dictionaries in Python: Unleashing Key-Value Pairs for Efficient Data Management*. [online] Medium. Available from: https://aniruddh123.medium.com/demystifying-dictionaries-in-python-unleashing-key-value-pairs-for-efficient-data-management-d0b994ea3cd [Accessed 8 May 2024].

Rashid, S.M., McCusker, J.P., Pinheiro, P., Bax, M.P., Santos, H.O., Stingone, J.A., Das, A.K. and McGuinness, D.L. (2020). The Semantic Data Dictionary – An Approach for Describing and Annotating Data. *Data Intelligence*, 2(4), pp.443–486. Available from: https://doi.org/10.1162/dint_a_00058.

Ross, J. (2023) *Chapter 6 Dictionaries* | *Introduction to Programming*. Available from: https://infx511.github.io/dictionaries.html [Accessed 28 May 2024].