Data Structures and how they support Object-Orientated Development

Data structures are a common part of programming. As software is developed and also built in a variety of languages it is likely some form of basic data structure will be used. From simple objects like lists and arrays to collections or dictionaries. With most modern languages you also then have the benefit of the built-in data methods such as count, add, and remove just as examples.

By having the ability to create immutable or mutable data structures and perform programmatical actions on them they support a wide range of activities. Code becomes easier to test as it can be documented as to what the data structure is. Exceptions and error handling can be applied to the objects or the items inside an object. They provide simple means of interrogation and manipulation as well.

As well as testing data structures to become searchable. Linear and random searches are probably the simplest searches to perform but are the most inefficient. More efficient searches such as binary or hash-based are available but often require data to be sorted and have an index.

The selection and use of data structures will often depend on the design of a solution or program and may be proposed upfront but could be added as part of the development to answer certain problems so decisions need to be made on how they fit within the design or possibly impact initial design or re-usable assets like requirements, design or even testing.

Dictionaries:

Can be referred to as a lookup table a dictionary allows multiple records to be added with a key and value pair. Mutable with the ability to run methods against the object and its contents.

Lists:

An ordered collection of items using an index value for each item or position in the list. Index[0] been the first and index[x] been the last. Also mutable with the ability to run methods against the object and its contents.

Tuples:

An immutable data structure that cannot be modified. Useful when you have a set of static values.