Data Strucutres Compared

The following table provides a comparison of the four main data structures:

Feature	List	Tuple	Set	String
Mutability	Mutable (can be modified)	Immutable (cannot be modified)	Mutable (can be modified)	Immutable (cannot be modified)
Ordering	Ordered (maintains insertion order)	Ordered (maintains insertion order)	Unordered (no guaranteed order)	Ordered (maintains character sequence)
Indexing	Supports indexing `list[0]`	Supports indexing `tuple[0]`	No indexing support	Supports indexing `string[0]`
Duplicates	Allows duplicate values	Allows duplicate values	No duplicates allowed	Allows duplicate characters
Syntax	`[1, 2, 3]` or `list()`	`(1, 2, 3)` or `tuple()`	`{1, 2, 3}` or `set()`	`"hello"` or `str()`
Primary Use Cases	General-purpose data storage, dynamic collections	Fixed collections, coordinates, function returns	Mathematical sets, removing duplicates, membership testing	Text data, character sequences
Performance	Good for most operations	Faster than lists for iteration	Very fast membership testing	Optimised for text operations
Memory Usage	Higher overhead due to mutability	Lower overhead, more memory efficient	Moderate overhead, hash table structure	Very memory efficient for text
Common Methods	`append()`, `remove()`, `pop()`, `sort()`	`count()`, `index()`	`add()`, `remove()`, `union()`, `intersection()`	`split()`, `join()`, `replace()`, `strip()`
Length Function	`len(my_list)`	`len(my_tuple)`	`len(my_set)`	`len(my_string)`