CheetSheet - Python DataStructures

This Python Data Structure Cheat Sheet is a concise guide to the most commonly used data structures, covering their key features, use cases, methods, performance characteristics, and tips. Mastering this single sheet can save significant time and make you a Python data structure guru.

Author - Vishal Mathur (Github - vmathur1)

Data Structure	Mutable	Ordered	Allows Duplicates	Unique Elements	Unique Keys	Homogeneous	Access Method
List	✓	✓	✓				Sequential
Tuple		✓	✓				Sequential
Set	✓			✓			
Dictionary	✓				✓		Associative key
String		✓	✓			✓ (characters only)	Sequential
Array (NumPy)		✓	✓			✓ (same data type)	Sequential

Data Structure	Description	Unique Features	When use?	When not?	Main Methods	Performance (Read/Insert)
List []	Leverage slicing for efficient subsetting of lists (irst_three = numbers[:3]). Consider converting lists to tuples if mutability is not required.	Mutable Ordered Allows duplicates Sequential access heterogeneous	General purpose storage, dynamic data sets	Inefficient for frequent insertions/deletions in the middle AND memory intensive for large data	append, insert, remove, pop, extend, clear, sort, reverse, copy	Fast for random access (O(1)), slower for insertion/ deletion (O (n))
Tuple ()	Use tuples when data integrity is crucial and accidental modification is undesired. Use named tuples (from collections module) for self-documenting tuples with named fields.	Immutable Ordered Allows duplicates Sequential access heterogeneous	Grouping related data, function return with multiple values, dictionary keys	Not suitable for situations requiring modifications AND Less flexible than lists	count, index, in	Faster than lists for lookups (O(1)), cannot be modified
Set {}	Remember, sets are mutable but element in it cannot contain mutable elements like lists or dictionaries. Confusing much? Utilize methods like union, intersection, and difference for set operations.	Mutable Unordered Unique elements No sequential access heterogeneous	Checking for membership, removing duplicates	No ordering AND cannot access elements by index	add, remove, discard, pop, update, intersection, union, difference	Fast for membership testing (O(1)), slower for access by index
Dictionary {}	Use get method with a default value to avoid KeyError if the key is not found. Consider using OrderedDict (from collections module) for maintaining insertion order.	Mutable Unordered Unique keys Associative access key heterogeneous	Associative lookups, mapping unique keys to values	Not suitable for ordered data AND has overhead for key-value lookups compared to lists/arrays for numeric data	get, setdefault, update, pop, items, keys, values, clear	Fast for lookups by key (O(1)), slower for access by index (not supported)
String ()	Remember, strings are immutable, so any modification results in a new string object. Leverage built-in string methods like split, join, and find for text manipulation.	Immutable Ordered Allows duplicates Sequential access Homogeneous chr	Representing text, formatting output	Cannot modify existing strings (creates new objects for changes) AND memory intensive for large text data	count, find, index, split, join, upper, lower, strip	Fast for character access (O(1)), not for numerical compute
Array (NumPy)	Leverage vectorized operations for efficient numerical computations. Utilize slicing for efficient subsetting of arrays. Explore broadcasting capabilities of NumPy arrays for simplified calculations.	Immutable Ordered Duplicates Sequential access Optimized num compute	High-performance numerical computations, linear algebra operations	Fixed size (cannot resize after creation) AND less intuitive syntax compared to built-in Python lists	reshape, transpose, ones, zeros, arange, linspace, sqrt, sin, cos, exp, mean, std, sum	Very fast for element-wise operations (O(1))