Python Data Structures Research

List and its Methods

A list in Python is an ordered collection of items that can be changed (mutable). Lists can contain elements of different types such as integers, strings, and even other lists. Lists are defined using square brackets [].

Common list methods include:

- append(x): Adds an element x to the end of the list.
- insert(i, x): Inserts an element x at a specific index i.
- remove(x): Removes the first occurrence of element x.
- pop(i): Removes and returns the element at index i (default is last).
- sort(): Sorts the list in ascending order.
- reverse(): Reverses the order of elements in the list.
- index(x): Returns the index of the first element equal to x.
- count(x): Returns the number of times x appears in the list.
- clear(): Removes all items from the list.

Lists are very flexible and commonly used in data processing and storage.

Sets and its Methods

A set is an unordered collection of unique elements. Sets are useful when you want to store items without duplicates. Sets are defined using curly braces {} or the set() function.

Common set methods include:

- add(x): Adds element x to the set.
- remove(x): Removes element x from the set (raises error if not found).
- discard(x): Removes element x without error if not found.
- pop(): Removes and returns a random element.
- clear(): Removes all elements from the set.
- union(other_set): Returns a new set with elements from both sets.
- intersection(other_set): Returns elements common to both sets.
- difference(other_set): Returns elements in the first set but not in the other.
- issubset(other): Checks if the set is a subset of another set.

Sets are commonly used in situations where uniqueness is important.

Dictionary and its Methods

A dictionary in Python stores data in key-value pairs. It is unordered and mutable. Dictionaries are created using curly braces with key:value pairs.

Example: {'name': 'Alice', 'age': 25}

Common dictionary methods include:
- get(key): Returns the value for the given key.
- keys(): Returns a view of all keys.
- values(): Returns a view of all values.
- items(): Returns a view of all key-value pairs.
- update(dict): Updates the dictionary with another dictionary.
- pop(key): Removes the item with the given key.
- popitem(): Removes the last inserted key-value pair.
- clear(): Removes all items from the dictionary.

Dictionaries are ideal for representing structured data like records or settings.

Tuple

A tuple is similar to a list, but it is immutable, meaning its elements cannot be changed after creation. Tuples are defined using parentheses ().

Example: my_tuple = (1, 2, 3)

Key characteristics:
- Can contain elements of different types.
- Support indexing and slicing.
- Faster than lists due to immutability.

- Commonly used for fixed collections like coordinates or return values.

Tuples support methods like:

- count(x): Returns the number of times x appears.
- index(x): Returns the index of the first occurrence of x.

Tuples are used when you want to protect data from modification.

Slicing

Slicing allows you to extract parts of sequences like lists, strings, or tuples. It uses the syntax: sequence[start:stop:step]

- start: Index to begin the slice (inclusive).
- stop: Index to end the slice (exclusive).
- step: Interval between elements (default is 1).

Example:

- my_list[1:4] returns elements at index 1, 2, and 3.
- my_list[:3] returns the first three elements.
- my_list[::2] returns every second element.

Slicing is a powerful way to access sub-parts of data without writing loops.