1. String Methods

Strings are immutable in Python, but you can use a variety of built-in methods to manipulate them.

Method	Description	Example
len(s)	Returns the length of the string.	len("hello") → 5
s.lower()	Converts all characters to lowercase.	"HELLO".lower() → "hello"
s.upper()	Converts all characters to uppercase.	"hello".upper() → "HELLO"
s.strip()	Removes leading and trailing whitespace.	" hello ".strip() → "hello"
s.replace(old, new)	Replaces all occurrences of old with new.	"hello world".replace("world", "Python") → "hello Python"
s.split(delim)	Splits the string into a list of substrings.	"a,b,c".split(",") → ["a", "b", "c"]
s.join(iterable)	Joins elements of an iterable with the string as separator.	"-".join(["a", "b", "c"]) → "a-b-c"
s.find(sub)	Returns the index of the first occurrence of sub (returns -1 if not found).	"hello".find("e") → 1
s.count(sub)	Returns the number of occurrences of sub in the string.	"hello".count("l") → 2
s.startswith(prefix)	Checks if the string starts with the specified prefix.	"hello".startswith("he") → True
s.endswith(suffix)	Checks if the string ends with the specified suffix.	"hello".endswith("lo") → True

---

2. List Methods

Lists are mutable, and they support a variety of methods for modifying and interacting with their elements.

Method	Description	Example
len(lst)	Returns the number of elements in the list.	len([1, 2, 3]) → 3
lst.append(x)	Adds an element x to the end of the list.	[1, 2].append(3) → [1, 2, 3]
lst.extend(iterable)	Adds all elements of the iterable to the list.	[1, 2].extend([3, 4]) → [1, 2, 3, 4]

Method	Description	Example
lst.insert(i, x)	Inserts element x at index i.	[1, 2].insert(1, 3) → [1, 3, 2]
lst.remove(x)	Removes the first occurrence of x from the list.	[1, 2, 2].remove(2) → [1, 2]
lst.pop(i)	Removes and returns the element at index i. If no index is specified, removes and returns the last item.	[1, 2].pop() → 2
lst.index(x)	Returns the index of the first occurrence of x in the list.	[1, 2, 3].index(2) → 1
lst.count(x)	Returns the number of occurrences of x in the list.	[1, 2, 2].count(2) → 2
lst.sort()	Sorts the elements of the list in place.	[3, 1, 2].sort() → [1, 2, 3]
lst.reverse()	Reverses the elements of the list in place.	[1, 2, 3].reverse() → [3, 2, 1]
lst.copy()	Returns a shallow copy of the list.	[1, 2].copy() → [1, 2]

---

3. Tuple Methods

Tuples are immutable, so their methods are limited to non-modifying operations.

Method	Description	Example
len(t)	Returns the number of elements in the tuple.	len((1, 2, 3)) → 3
t.count(x)	Returns the number of occurrences of x in the tuple.	(1, 2, 2).count(2) → 2
t.index(x)	Returns the index of the first occurrence of x in the tuple.	(1, 2, 3).index(2) → 1

---

4. Set Methods

Sets are unordered collections of unique elements, and they support methods for performing set operations.

Method	Description	Example
len(s)	Returns the number of elements in the set.	len({1, 2, 3}) → 3
s.add(x)	Adds element x to the set.	{1, 2}.add(3) → {1, 2, 3}
s.remove(x)	Removes element x from the set. Raises KeyError if x is not present.	{1, 2}.remove(1) → {2}

Method	Description	Example
s.discard(x)	Removes element x from the set. Does not raise an error if x is not found.	{1, 2}.discard(1) → {2}
s.pop()	Removes and returns an arbitrary element from the set.	{1, 2}.pop() → 1 or 2 (arbitrary)
s.clear()	Removes all elements from the set.	{1, 2}.clear() → set()
s.union(t)	Returns a set containing all elements from s and t.	{1, 2}.union({2, 3}) → {1, 2, 3}
s.intersection(t)	Returns a set containing elements common to both s and t.	{1, 2}.intersection({2, 3}) → {2}
s.difference(t)	Returns a set containing elements in s but not in t.	{1, 2}.difference({2, 3}) → {1}
s.symmetric_difference(t)	Returns a set containing elements in either s or t, but not both.	{1, 2}.symmetric_difference({2, 3}) → {1, 3}

---

5. Dictionary Methods

Dictionaries are mutable and store key-value pairs. Here are some commonly used methods.

Method	Description	Example
len(d)	Returns the number of key-value pairs in the dictionary.	len({'a': 1, 'b': 2}) → 2
d.get(key)	Returns the value for key, or None if key is not found.	{'a': 1}.get('a') → 1
d.keys()	Returns a view object containing all the keys.	{'a': 1}.keys() → dict_keys(['a'])
d.values()	Returns a view object containing all the values.	{'a': 1}.values() → dict_values([1])
d.items()	Returns a view object containing key-value pairs.	{'a': 1}.items() → dict_items([('a', 1)])
d.update(other)	Updates the dictionary with key-value pairs from another dictionary or iterable.	{'a': 1}.update({'b': 2}) → {'a': 1, 'b': 2}
d.pop(key)	Removes and returns the value associated with key.	{'a': 1}.pop('a') → 1

Method	Description	Example
d.popitem()	Removes and returns an arbitrary (key, value) pair.	{'a': 1}.popitem() → ('a', 1)
d.clear()	Removes all key-value pairs from the dictionary.	{'a': 1}.clear() → {}
d.setdefault(key, default)	Returns the value for key, and if key doesn't exist, inserts it with default value.	{'a': 1}.setdefault('b', 2) → 2