Options	String	List	Set	Dictionary

Create	**# Single quotes**
	string1 = 'Hello, world!'
	
	# Double quotes
	string2 = "Python”
	
	# Triple quotes for multiline strings
	string3 = """This is a
	            multiline string."""
		# Empty list
		my_list = []
		
		# List with elements
		my_list = [1, 2, 3, 4]
		
		# Mixed data types
		my_list = [1, "hello", 3.14]
			#Empty Set
			my_set = set() # Note: {} creates an empty dictionary, not a set.
			
			#Set with Elements
			my_set = {1, 2, 3}
			
			#From an Iterable
			my_set = set([1, 2, 2, 3]) # {1, 2, 3}
				#Dictionary with Key-Value Pairs
				my_dict = {"name": "Alice", "age": 25, "city": "New York"}
				
				#Using the dict() Constructor
				my_dict = dict(name="Alice", age=25, city="New York")
				
				#From a List of Tuples
				my_dict = dict([("name", "Alice"), ("age", 25)])

Read/Access	**# Accessing characters**
	first_char = string1[0]
	last_char = string1[-1]
	
	# Slicing
	substring = string1[0:5]
		#Indexing: Access elements by their position
		my_list = [10, 20, 30, 40]
		print(my_list[0])
		# 10 (first element)
		print(my_list[-1])
		# 40 (last element)
		
		#Slicing: Extract portions of the list.
		print(my_list[1:3])
		# [20, 30]
		print(my_list[:2])
		# [10, 20]
		
		#Iterating Through a List
		for item in my_list:
		   print(item)
			# for loop
			print("\nElements of set: ")
			for i in set1:
			   print(i, end=" ")
			
			# Checking the element using in keyword
			print("\n")
			print("Geeks" in set1)
			
			# using list() method
			s = set([1, 2, 3])
			list(s)[0]
				#Using Keys
				print(my_dict["name"])
				
				#Using get() (to avoid KeyError if the key doesn't exist)
				print(my_dict.get("age")) # 25
				print(my_dict.get("height", "Not found")) # Not found
				
				#Checking Membership Keys Only
				print("name" in my_dict) # True
				print("salary" not in my_dict) # True
				
				#Dictionary Methods
				keys(): Returns a view of all keys.
				print(my_dict.keys()) # dict_keys(['name', 'age', 'city'])
				
				values(): Returns a view of all values.
				print(my_dict.values()) # dict_values(['Alice', 25, 'New York'])
				
				items(): Returns a view of all key-value pairs.
				print(my_dict.items()) # dict_items([('name', 'Alice'), ('age', 25), ('city', 'New York')])

Update	**# Changing case**
	upper_case = string1.upper()
	lower_case = string1.lower()
	
	# Splitting and joining
	words = string2.split()
	
	# Splits into a list of words
	joined = " ".join(words) # Joins list into a string
	
	# Stripping whitespace
	trimmed = " Hello ".strip()
	
	# Replace
	replaced = string1.replace("world", "Python")
	
	# Finding substrings
	index = string1.find("world") # Returns -1 if not found
	
	# Checking content
	is_alpha = "abc".isalpha()
	
	is_digit = "123".isdigit()
		#append(): Adds an element to the end.
		my_list.append(50)
		
		#extend(): Adds elements from another list.
		my_list.extend([60, 70])
		
		#insert(): Inserts an element at a specific index
		my_list.insert(2, 25) # Inserts 25 at index 2
		
		#Modifying Elements
		my_list[0] = 15 # Change first element to 15
			add(): Adds a single element.
			my_set.add(4)
			
			update(): Adds multiple elements (from an iterable).
			my_set.update([5, 6])
				# Add a new key-value pair
				my_dict["height"] = 170
				
				# Update an existing key
				my_dict["age"] = 26
				
				#update(): Updates the dictionary with key-value pairs from another dictionary or iterable.
				my_dict.update({"age": 30, "city": "San Francisco"})

Delete	**1. Delete a Character by Index** **You can create a new string without the character at a specific index using slicing.** s = "hello" index_to_delete = 1 new_s = s[:index_to_delete] + s[index_to_delete + 1:] print(new_s) # Output: hllo **2. Remove a Specific Character** **Use the replace() method to remove all occurrences of a specific character**.	**#remove(): Removes the first occurrence of a value.** my_list.remove(20) **#pop(): Removes and returns an element by index.** my_list.pop() # Removes last element my_list.pop(1) # Removes element at index 1	**remove(): Removes a specific element (raises an error if not found).** my_set.remove(2) **discard(): Removes a specific element (does not raise an error if not found).** my_set.discard(10) **pop(): Removes and returns an arbitrary element**. my_set.pop() **clear(): Removes all elements.**	**#pop(): Removes a key and returns its value**. age = my_dict.pop("age") # Removes "age" **#popitem(): Removes and returns the last inserted key-value pair (arbitrary before Python 3.7).** last_item = my_dict.popitem() **#del: Deletes a key-value pair**. del my_dict["city"] **#clear(): Removes all elements**. my_dict.clear()

	s = "hello" new_s = s.replace("l", "") print(new_s) # Output: heo **3. Remove Characters by Condition** **Use list comprehensions or the filter() function to remove characters conditionally.** s = "hello123" new_s = ''.join([char for char in s if not char.isdigit()]) print(new_s) # Output: hello **4. Delete a Substring** **You can use replace() to remove a specific substring.** s = "hello world" new_s = s.replace("world", "") print(new_s.strip()) # Output: hello **5. Delete the Entire String** **If you want to delete a string completely, you can set the variable to None or an empty string**. s = "hello" s = None # or s = "" print(s) # Output: None or "" Strings are immutable in Python, so methods that modify a string return a new string. Use r"raw strings" to avoid escaping backslashes in	**#clear(): Removes all elements.** my_list.clear()	my_set.clear()	