**1. Creating a list and accessing elements:**

* A list is a collection of items stored inside square brackets [].
* Example:

fruits = ["apple", "banana", "cherry", "date"]

* Access elements by index:

print(fruits[0]) # apple

print(fruits[2]) # cherry

**2. Indexing in lists:**

* **Positive indexing:** counts from the beginning (0, 1, 2…).
  + Example: fruits[1] → "banana"
* **Negative indexing:** counts from the end (-1, -2, -3…).
  + Example: fruits[-1] → "date"
* Indexing helps access **individual elements** directly.

**3. Slicing a list:**

* Slicing is used to access a **range of elements**.
* Syntax: list[start:end] → includes start, excludes end.
  + Example: fruits[1:3] → ["banana", "cherry"]
* Omitting start → starts from beginning: fruits[:2] → ["apple", "banana"]
* Omitting end → goes to last element: fruits[2:] → ["cherry", "date"]
* Using step: list[start:end:step] → selects every step element:
  + Example: fruits[0:4:2] → ["apple", "cherry"]

**4. Common list operations:**

* **Concatenation:** joining two lists using +.

list1 = [1, 2]

list2 = [3, 4]

result = list1 + list2 # [1, 2, 3, 4]

* **Repetition:** repeating a list using \*.

list1 = [1, 2]

result = list1 \* 2 # [1, 2, 1, 2]

* **Membership:** check if an element exists in a list using in.

fruits = ["apple", "banana"]

print("apple" in fruits) # True

print("cherry" in fruits) # False

**5. List methods:**

* **append():** adds an item at the end of the list.

fruits = ["apple", "banana"]

fruits.append("cherry") # ["apple", "banana", "cherry"]

* **insert():** inserts an item at a specific index.

fruits.insert(1, "orange") # ["apple", "orange", "banana", "cherry"]

* **remove():** removes the first occurrence of a value.

fruits.remove("banana") # ["apple", "orange", "cherry"]

* **pop():** removes and returns an element by index (default last).

fruits.pop() # removes "cherry"

fruits.pop(0) # removes "apple"

**6. Iterating over a list using loops:**

* **For loop:** used to go through each element of the list.

fruits = ["apple", "banana", "cherry"]

for fruit in fruits:

print(fruit)

* **While loop:** can also iterate using an index.

i = 0

while i < len(fruits):

print(fruits[i])

i += 1

**7. Sorting and reversing a list:**

* **sort():** sorts the list **in place** (changes original list).

numbers = [3, 1, 4, 2]

numbers.sort() # [1, 2, 3, 4]

* **sorted():** returns a **new sorted list** without changing original.

numbers = [3, 1, 4, 2]

new\_numbers = sorted(numbers) # [1, 2, 3, 4]

print(numbers) # [3, 1, 4, 2]

* **reverse():** reverses the list **in place**.

numbers = [1, 2, 3, 4]

numbers.reverse() # [4, 3, 2, 1]

**8. Basic list manipulations:**

* **Addition:**
  + Using append() to add one element: fruits.append("date")
  + Using extend() to add multiple elements: fruits.extend(["mango", "kiwi"])
  + Using insert() to add at a specific position: fruits.insert(1, "orange")
* **Deletion:**
  + Using remove() to delete by value: fruits.remove("banana")
  + Using pop() to delete by index: fruits.pop(0)
  + Using del to delete element or slice: del fruits[1:3]
* **Updating:**
  + Change element by index: fruits[0] = "apple\_new"
* **Slicing:**
  + Access a range of elements: fruits[1:3]
  + Modify a range: fruits[1:3] = ["kiwi", "mango"]

**9. Introduction to tuples and immutability:**

* A tuple is a **collection of items**, similar to a list, but **cannot be changed** after creation.
* This property is called **immutability**.
* Tuples are written with **parentheses ()**.
* Example:

my\_tuple = (1, 2, 3, 4)

**10. Creating and accessing elements in a tuple:**

* **Creating:**

t1 = (10, 20, 30)

t2 = ("apple", "banana")

* **Accessing elements using indexing:**
  + Positive index: starts from 0
* print(t1[1]) # 20
  + Negative index: starts from -1
* print(t2[-1]) # banana
* **Slicing tuples:**

print(t1[0:2]) # (10, 20)

**11. Basic operations with tuples:**

* **Concatenation:** join two tuples using +

t1 = (1, 2)

t2 = (3, 4)

t3 = t1 + t2 # (1, 2, 3, 4)

* **Repetition:** repeat a tuple using \*

t1 = (1, 2)

t2 = t1 \* 2 # (1, 2, 1, 2)

* **Membership:** check if an element exists using in

fruits = ("apple", "banana")

print("apple" in fruits) # True

print("cherry" in fruits) # False

**12. Accessing tuple elements using indexing:**

* **Positive indexing:** counts from 0 for the first element.

my\_tuple = (10, 20, 30, 40)

print(my\_tuple[0]) # 10

print(my\_tuple[2]) # 30

* **Negative indexing:** counts from -1 for the last element.

print(my\_tuple[-1]) # 40

print(my\_tuple[-3]) # 20

* Indexing helps **access individual elements directly**.

**13. Slicing a tuple (accessing ranges of elements):**

* Syntax: tuple[start:end] → includes start, excludes end.

my\_tuple = (10, 20, 30, 40, 50)

print(my\_tuple[1:4]) # (20, 30, 40)

* Omitting start → starts from beginning:

print(my\_tuple[:3]) # (10, 20, 30)

* Omitting end → goes to last element:

print(my\_tuple[2:]) # (30, 40, 50)

* Using step → select elements at intervals:

print(my\_tuple[0:5:2]) # (10, 30, 50)

**14. Introduction to dictionaries:**

* A dictionary is a collection of **key-value pairs**.
* Each key is unique and used to access its value.
* Dictionaries are written using {}.
* Example:

person = {"name": "Utsav", "age": 20, "city": "Ahmedabad"}

**15. Accessing dictionary elements:**

Access a value using its key:

print(person["name"]) # Utsav

**Adding elements to a dictionary:**

Add a new key-value pair:

person["email"] = "utsav@example.com"

**Updating elements in a dictionary:**

Change the value of an existing key:

person["age"] = 21

**Deleting elements from a dictionary:**

Remove a key-value pair using del:

del person["city"]

Or using pop():

person.pop("email")

**16. Dictionary methods:**

* keys() → returns all keys:

print(person.keys()) # dict\_keys(['name', 'age'])

* values() → returns all values:

print(person.values()) # dict\_values(['Utsav', 21]

* items() → returns key-value pairs as tuples:

print(person.items()) # dict\_items([('name', 'Utsav'), ('age', 21)])

**17. Iterating over a dictionary using loops:**

* **Using for loop on keys:**

person = {"name": "Utsav", "age": 20}

for key in person:

print(key, ":", person[key])

* **Using items() to get key and value together:**

for key, value in person.items():

print(key, "=", value)

**18. Merging two lists into a dictionary:**

* **Using a loop:**

keys = ["name", "age", "city"]

values = ["Utsav", 20, "Ahmedabad"]

my\_dict = {}

for i in range(len(keys)):

my\_dict[keys[i]] = values[i]

print(my\_dict)

# Output: {'name': 'Utsav', 'age': 20, 'city': 'Ahmedabad'}

**19. Counting occurrences of characters in a string using dictionaries:**

* Step 1: create an empty dictionary.
* Step 2: loop through each character in the string.
* Step 3: update count in the dictionary.

text = "hello"

count = {}

for char in text:

if char in count:

count[char] += 1

else:

count[char] = 1

print(count)

# Output: {'h': 1, 'e': 1, 'l': 2, 'o': 1}

**20. Defining functions in Python:**

* A function is a **block of code** that performs a specific task.
* Defined using the def keyword.
* Example:

def greet():

print("Hello!")

* Call it by its name: greet() → Output: Hello!

**21. Different types of functions:**

* **a) Without parameters and without return value:**

def greet():

print("Hello!")

* **b) With parameters and without return value:**

def greet(name):

print("Hello", name)

greet("Utsav") # Output: Hello Utsav

* **c) Without parameters but with return value:**

def get\_greeting():

return "Hello!"

message = get\_greeting()

print(message) # Hello!

* **d) With parameters and with return value:**

def add(a, b):

return a + b

result = add(5, 3)

print(result) # 8

**22. Anonymous functions (lambda functions):**

* Lambda functions are **short, one-line functions** without a name.
* Syntax: lambda arguments: expression
* Example:

square = lambda x: x\*\*2

print(square(5)) # 25

* Can also use multiple arguments:

add = lambda a, b: a + b

print(add(2, 3)) # 5