Name :- Gajera pal

**Accessing List**

**1.Creating and Accessing Elements in a List**:

**ANS**

* A list is created using square brackets [].
* You access elements by referencing their index. For example:
* my\_list = [10, 20, 30, 40]
* print(my\_list[0])

**2.Indexing in Lists**:

**ANS**

* **Positive Indexing**: Starts from 0 for the first element.
* my\_list[1]
* **Negative Indexing**: Starts from -1 for the last element.
* my\_list[-1]

**3.Slicing a List**:

**ANS**

* To access a range of elements, use slicing: list[start:end] (includes start, but excludes end).
* my\_list[1:3]
* You can also omit start or end:
  + my\_list[:3] gets elements from the beginning up to (but not including) index 3.
  + my\_list[2:] gets elements from index 2 to the end.

**List Operations**

1.**Common List Operations:**

**ANS**

**Concatenation:**

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

**Repetition**:

* my\_list = [1, 2]
* result = my\_list \* 2 # Outputs: [1, 2, 1, 2]

**Membership**:

* 3 in my\_list # Outputs: True if 3 is in my\_list
* 5 not in my\_list # Outputs: True if 5 is not in my\_list

**2.List Methods:**

**ANS**

**append()**:

my\_list.append(5)

**insert()**:

my\_list.insert(1, 6)

**remove()**:

my\_list.remove(6)

**pop()**:

popped = my\_list.pop(1)

**Working with Lists**

**1.Iterating Over a List:**

**ANS**

my\_list = [1, 2, 3]

for item in my\_list:

print(item)

**2.Sorting and Reversing a List:**

**ANS**

**sort()**:

my\_list.sort()

**sorted()**:

sorted\_list = sorted(my\_list)

**reverse()**:

my\_list.reverse()

**3.Basic List Manipulations:**

**ANS**

**Addition**:

my\_list.append(4)

**Deletion**:

del my\_list[1]

**Updating**:

my\_list[0] = 10

**Slicing**:

sub\_list = my\_list[1:3]

**Tuple**

**1.Introduction to Tuples and Immutability:**

**ANS**

* A **tuple** is a collection of ordered elements, similar to a list, but **immutable** (cannot be modified after creation).
* my\_tuple = (1, 2, 3)

**2.Creating and Accessing Elements in a Tuple:**

**ANS**

* Tuples are created using parentheses ().
* my\_tuple = (10, 20, 30)
* Access elements via indexing (positive or negative).
* print(my\_tuple[0]) # 10
* print(my\_tuple[-1]) # 30

**3.Basic Operations with Tuples:**

**ANS**

**Concatenation**:

tuple1 = (1, 2)

tuple2 = (3, 4)

result = tuple1 + tuple2 # (1, 2, 3, 4)

**Repetition**:

my\_tuple = (1, 2)

result = my\_tuple \* 2 # (1, 2, 1, 2)

**Membership**:

3 in my\_tuple # True if 3 is in my\_tuple

5 not in my\_tuple # True if 5 is not in my\_tuple

**Accessing Tuples**

**1.Accessing tuple elements using positive and negative indexing.**

**ANS**

* Positive indexing: tuple[1] (accesses the second element).
* Negative indexing: tuple[-1] (accesses the last element).

**2. Slicing a tuple to access ranges of elements?**

**ANS**

Slicing a tuple: tuple[start:stop]

* Accesses elements from start index to stop-1 index.  
  Example: tuple[1:4] gives elements at indices 1, 2, and 3.

**Dictionaries**

**1. Introduction to dictionaries: key-value pairs.**

**ANS**

A dictionary is a collection of key-value pairs, where each key is unique, and it's associated with a value.  
Example: {'name': 'Alice', 'age': 25}

**2.** **Accessing, adding, updating, and deleting dictionary elements.**

**ANS**

* **Accessing**: Use the key to get the value.  
  Example: dict['key']
* **Adding**: Assign a value to a new key.  
  Example: dict['new\_key'] = 'value'
* **Updating**: Change the value of an existing key.  
  Example: dict['key'] = 'new\_value'
* **Deleting**: Remove a key-value pair using del.  
  Example: del dict['key']

**3. Dictionary methods like keys(), values(), and items().**

**ANS**

* **keys()**: Returns a list of all keys.  
  Example: dict.keys()
* **values()**: Returns a list of all values.  
  Example: dict.values()
* **items()**: Returns a list of all key-value pairs as tuples.  
  Example: dict.items()

**Working with Dictionaries**

**1. Iterating over a dictionary using loops**

**ANS**

**Iterating over keys**:

* for key in dict:
* print(key)

**Iterating over values**:

* for value in dict.values():
* print(value)

**Iterating over key-value pairs**:

* for key, value in dict.items():
* print(key, value)

**2. Merging two lists into a dictionary using loops or zip()**

**ANS**

Using zip():

keys = ['a', 'b', 'c']

values = [1, 2, 3]

dict = dict(zip(keys, values))

Using a loop:

keys = ['a', 'b', 'c']

values = [1, 2, 3]

dict = {}

for i in range(len(keys)):

dict[keys[i]] = values[i]

**3.** **Counting occurrences of characters in a string using dictionaries**

**ANS**

string = "hello"

count = {}

for char in string:

count[char] = count.get(char, 0) + 1

**Functions**

**1. Defining functions in Python**

**ANS**

To define a function in Python, use the def keyword followed by the function name and parentheses:

def function\_name(parameters):

return value

Example:

def greet(name):

return f"Hello, {name}!"

**2.** **Different types of functions: with/without parameters, with/without return values.**

**ANS**

**Function with parameters and return value**:

def add(a, b):

return a + b

**Function with parameters and no return value**:

def greet(name):

print(f"Hello, {name}!")

**Function without parameters and with return value**:

def get\_number():

return 42

**Function without parameters and no return value**:

def say\_hello():

print("Hello!")

**3. Anonymous functions (lambda functions).**

**ANS**

**Syntax:**

lambda parameters: expression

**Example**

add = lambda x, y: x + y

print(add(2, 3)) # Output: 5

**Modules**

**1.** **Introduction to Python modules and importing modules.**

**ANS**

import math

print(math.sqrt(16))

from math import sqrt

print(sqrt(16))

**2. Standard library modules: math, random.**

**ANS**

**math module**: Provides mathematical functions like sqrt(), sin(), cos(), etc. Example:

* import math
* print(math.sqrt(16))

**random module**: Provides functions for generating random numbers, such as randint(), choice(), random(), etc. Example:

* import random
* print(random.randint(1, 10))

**3. Creating custom modules.**

**ANS**

To create a custom module, write your functions or variables in a Python file (e.g., mymodule.py). Then, import and use it in other scripts.

1. Create a file mymodule.py:
2. def greet(name):
3. return f"Hello, {name}!"
4. Import the custom module in another script:
5. import mymodule
6. print(mymodule.greet("Alice"))