Q.1 why might you choose a deque from the collection module to implement a queue instead of using a regular python list?

Ans – 1. efficient 0(1) appends and pops from both ends with a deque, you can efficiently append and pop items from both the front and back with constant time complexity,0(1)2

2. queue operations – a queue typically requires fast access to both ends (enqueue and deque operations) using a deque allows for optimal performance when you need to dequeue items from the front.

Q2. Can you explain a real-world scenario where using a stack would be a more practical choice than a list for data storage and retrieval?

Ans – scenario:

“Undo/redo functionality in text editor “

Imagine you’re building a text editor, and you want to implement the undo/redo functionality.

1.when the user types something you want to store the current state of the text.

2.when the user presses the “undo” button, you want to revert to the previous state of the text.

3.when the user presses the “redo” button, you want to revert to the next state of the text

Stack is better choice;

1.last-in-first-out order = which means the most recent state of the text is always at the top of the stack. This makes it easy to implement the undo functionality.

2. efficient insertion and deletion = which is perfect for storing and retrieving the states of the text

3. no need for random access= this makes them more memory-efficient and faster for this specific use case

Q.3 what is the primary advantage of using sets in python, and in what type of problem-solving scenarios are they most useful?

Ans- primary advantage of sets in python:

1.fast membership testing = sets allow for quick checking of element membership, with an average time complexity of 0(1)

2.efficient removal of duplicates = sets automatically eliminate duplicate elements, making them useful for data cleaning and processing.

3.fast set operations = sets support fast union, intersection, and operations with an average time complexity on 0(n)

Problem –solving scenarios where sets are most useful:

1.data duplication = sets help remove duplicate elements from datasets, ensuring data consistency and accuracy

2.set-based operations = intersection, union, difference

3.data processing and analysis = data cleaning, filtering

Q.4when might you choose to use an array instead of a list for storing numerical data in python? What benefits do arrays offer this content?

Ans=when to use arrays:

1.numerical computations – arrays provide faster access and manipulation of numerical data

2.large datasets=arrays store data in contiguous block of memory,

3.matrix operations= arrays are more suitable than list

Benefits of array:

1.faster access and manipulation

2.memory efficiency

3.interoperability with other libraries

When not to use arrays:

1.non-numerical data:

2.dynamic insertion/deletion

Q.5 In python, what's the primary difference between dictionaries and lists, and how does this difference impact their use cases in programming?

Ans- primary difference:

1. Order and indexing
2. Key –value pairs
3. Data retrieval

Impact on use cases

1.list =

Suitable for storing collections of data that need to be accessed by their index or position.

Ideal for tasks like storing and indexing

2.dictionaries=

Suitable for storing data as key-value pairs, where each key is unique and maps to specific value

Ideal for tasks like fast lookups, data caching, and configuration storage.