1. A singly linked list is a sequence of nodes where each node contains data and a reference to the next node.

2. Memory is allocated dynamically at runtime, typically using heap allocation when new nodes are created.

3. Create a new node, set its next pointer to the current head, and then update the head to be this new node.

4. Searching takes O(n) time in the worst case since each node may need to be examined.

5. By treating the head of the linked list as the top of the stack, you can push by inserting at the beginning and pop by removing the head.

6. Advantages include dynamic size, ease of insertion and deletion, and constant time operations for push and pop.

7. A queue is implemented by maintaining two pointers (front and rear); enqueue adds at the rear, and dequeue removes from the front.

8. To delete a node, update the previous node's pointer to skip the deleted node, and then free or dereference the removed node.

9. Polynomial addition involves representing each polynomial as a linked list of terms and then merging terms with the same exponent during addition.

10. Sparse matrices are represented by storing only the non-zero elements in linked lists along with their row and column indices.

11. A doubly linked list is a linked structure where each node contains pointers to both the next and previous nodes.

12. Inserting involves creating a new node and updating the previous and next pointers of the surrounding nodes to include the new node.

13. Dynamic storage management is the process of allocating and deallocating memory during program execution.

14. A circular linked list is one where the last node points back to the first, forming a continuous loop.

15. A cycle can be detected using Floyd's cycle-finding algorithm (the tortoise and hare method).

16. In Python, you define a node with a class that contains attributes for the data and a pointer (or reference) to the next node.

17. Example Python code for three nodes:

```
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None
head = Node(1)
second = Node(2)
third = Node(3)
head.next = second
second.next = third
```

18. To insert at the beginning, create a new node, set its next pointer to the current head, and then reassign the head to this new node.

19. Traversing a singly linked list is O(n) since every node is visited once.

20. Inserting at the end is O(n) if you don't maintain a tail pointer, as you must traverse the list to reach the end.

21. Deleting a node is O(n) if you need to search for the node; if the node is already referenced, deletion can be done in O(1) time.

22. Reverse the list by iteratively changing each node's next pointer to point to the previous node until all pointers are reversed.

23. A linked list offers dynamic sizing and efficient insertions/deletions compared to arrays.

24. Its disadvantages include extra memory overhead for pointers and the lack of random access.

25. In Python, implement a stack with a linked list by defining a Node class and using head insertion (push) and head removal (pop).

26. Pushing an element onto a linked stack is O(1) since it only involves inserting at the beginning.

27. Popping an element from a linked stack is O(1) because it removes the head node.

28. Implement a queue by linking nodes, with enqueue adding a node at the tail and dequeue removing a node from the head.

29. Enqueueing is O(1) if a tail pointer is maintained, allowing direct insertion at the end.

30. Dequeueing is O(1) since it involves removing the head node.

31. Represent a polynomial with a linked list where each node contains a coefficient and exponent, allowing operations on each term.

32. Adding two polynomials is typically O(n + m), where n and m are the number of terms in each polynomial, by traversing both lists simultaneously.

33. A singly linked list has a single pointer per node (to the next node), while a doubly linked list has two pointers (to both the next and previous nodes).

34. To insert at the end of a doubly linked list, traverse to the last node (or use a tail pointer), create a new node, and update the pointers of the last node and the new node accordingly.

35. Deleting a node from a doubly linked list can be O(1) if the node is directly accessible, as the previous pointer makes pointer adjustment straightforward.

36. A circular linked list allows for continuous traversal from any node without the need for a null reference to mark the end.

37. Create a circular linked list in Python by linking the last node's next pointer to the head node after constructing the list.

38. Traversal in a circular linked list is O(n), but you must stop when you reach the starting node

again to avoid an infinite loop.

39. Garbage collection in Python is an automatic process that frees up memory by removing objects that are no longer in use.

40. The gc module in Python provides tools to interact with the garbage collector, allowing manual control and debugging of memory management.