Data Structures in Python - Linked List, Stack, Queue, Graph

Singly Linked List

```
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None

class LinkedList:
    def __init__(self):
        self.head = None

    def insert(self, data):
        new_node = Node(data)
        if self.head is None:
            self.head = new_node
        else:
            current = self.head
            while current.next:
                current = current.next
            current.next = new_node

    def delete(self, key):
        current = self.head
        if current and current.data == key:
            self.head = current.next
            return

        prev = None
        while current and current.data != key:
            prev = current
            current = current.next

        if current:
            prev.next = current.next

    def display(self):
        current = self.head
        while current:
            print(current.data, end=" -> ")
            current = current.next
        print("None")
```

Stack using List

```
class Stack:
    def __init__(self):
        self.stack = []
    def push(self, data):
        self.stack.append(data)

    def pop(self):
        if not self.is_empty():
            return self.stack.pop()

    def peek(self):
        if not self.is_empty():
            return self.stack[-1]

    def is_empty(self):
        return len(self.stack) == 0

    def display(self):
        print(self.stack)
```

Queue using List

```
class Queue:
    def __init__(self):
        self.queue = []

    def enqueue(self, data):
        self.queue.append(data)

    def dequeue(self):
        if not self.is_empty():
            return self.queue.pop(0)

    def is_empty(self):
        return len(self.queue) == 0

    def display(self):
        print(self.queue)
```

Queue using deque from collections

```
from collections import deque

class DequeQueue:
    def __init__(self):
        self.queue = deque()

    def enqueue(self, data):
        self.queue.append(data)
    def dequeue(self):
        if not self.is_empty():
            return self.queue.popleft()

    def is_empty(self):
        return len(self.queue) == 0

    def display(self):
        print(list(self.queue))
```

Graph using Adjacency List

```
class Graph:
    def __init__(self):
        self.graph = {}

    def add_edge(self, u, v):
        if u not in self.graph:
            self.graph[u] = []
        self.graph[u].append(v)

    def display(self):
        for node in self.graph:
            print(f"{node} -> {self.graph[node]}")
```