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

1. Linked List

```
# Linked List Implementation in Python
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None

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

    def append(self, data):
        new_node = Node(data)
        if not self.head:
            self.head = new_node
            return
        temp = self.head
        while temp.next:
            temp = temp.next
        temp.next = new_node

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

2. Stack

```
# Stack Implementation Using List
class Stack:
    def __init__(self):
        self.stack = []

    def push(self, data):
        self.stack.append(data)
```

```
    def pop(self):
        if not self.stack:
            return "Underflow"
        return self.stack.pop()

    def peek(self):
        return self.stack[-1] if self.stack else "Empty"

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

3. Queue (Using deque)

```
# Queue Implementation Using collections.deque
from collections import deque

class Queue:
    def __init__(self):
        self.q = deque()

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

    def dequeue(self):
        if not self.q:
            return "Underflow"
        return self.q.popleft()

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

4. Graph (Using Adjacency List)

```
# Graph Implementation Using Adjacency List
class Graph:
    def __init__(self):
        self.adj = {}

    def add_edge(self, u, v):
```

```
        if u not in self.adj:
            self.adj[u] = []
        if v not in self.adj:
            self.adj[v] = []
        self.adj[u].append(v)
        self.adj[v].append(u)  # For undirected graph

    def display(self):
        for node in self.adj:
            print(node, "->", self.adj[node])
```