1. Arrays (Lists in Python)

Python uses lists as dynamic arrays.

```
# Example: Array operations
arr = [1, 2, 3, 4, 5]

# Access element
print(arr[2])  # O(1)

# Insert at end
arr.append(6)  # O(1)

# Insert at index
arr.insert(2, 10)  # O(n)

# Delete element
arr.remove(3)  # O(n)

# Search element
print(4 in arr)  # O(n)
```

2. Stack

Stack follows LIFO (Last In First Out).

```
# Example: Stack using list
stack = []

# Push
stack.append(10)  # O(1)
stack.append(20)

# Pop
print(stack.pop())  # O(1)

# Peek
print(stack[-1])  # O(1)
```

3. Queue

Queue follows FIFO (First In First Out).

```
from collections import deque

# Example: Queue
queue = deque()

# Enqueue
queue.append(10)  # O(1)
queue.append(20)

# Dequeue
print(queue.popleft())  # O(1)

# Peek
print(queue[0])  # O(1)
```

4. Linked List

Linked List implementation using classes.

```
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)
        new_node.next = self.head
        self.head = new_node

    def display(self):
        curr = self.head
        while curr:
            print(curr.data, end=" -> ")
            curr = curr.next

ll = LinkedList()
ll.insert(10)
ll.insert(20)
ll.display()  # O(n)
```

5. Binary Search

Efficient searching algorithm on sorted arrays.

```
def binary_search(arr, target):
    low, high = 0, len(arr) - 1
    while low <= high:
        mid = (low + high) // 2
        if arr[mid] == target:
            return mid
        elif arr[mid] < target:
            low = mid + 1
        else:
            high = mid - 1
    return -1

arr = [1, 2, 3, 4, 5, 6]
print(binary_search(arr, 4))  # O(log n)
```

6. Sorting Algorithms

Examples of common sorting techniques.

```
# Bubble Sort - O(n^2)
def bubble_sort(arr):
    n = len(arr)
    for i in range(n):
        for j in range(0, n - i - 1):
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]
    return arr

# Quick Sort - O(n log n) average
def quick_sort(arr):
    if len(arr) <= 1:
        return arr
    pivot = arr[len(arr)//2]
    left = [x for x in arr if x < pivot]
    middle = [x for x in arr if x == pivot]
    right = [x for x in arr if x > pivot]
    return quick_sort(left) + middle + quick_sort(right)

print(bubble_sort([64, 34, 25, 12, 22]))
print(quick_sort([64, 34, 25, 12, 22]))
```

7. Hash Map (Dictionary)

Python dictionary as hash map.

```
# Example: Dictionary operations
hash_map = {}

# Insert
hash_map["name"] = "Alice"  # O(1)

# Access
print(hash_map["name"])  # O(1)

# Delete
del hash_map["name"]  # O(1)

# Search
print("name" in hash_map)  # O(1)
```

8. Graph (Adjacency List)

Graph implementation using dictionary.

```
graph = {
    'A': ['B', 'C'],
    'B': ['A', 'D'],
    'C': ['A', 'D'],
    'D': ['B', 'C']
}

# BFS - O(V+E)
from collections import deque
def bfs(start):
    visited = set()
    queue = deque([start])
    while queue:
        node = queue.popleft()
        if node not in visited:
            print(node, end=" ")
            visited.add(node)
            queue.extend(graph[node])

bfs('A')
```