List

```
list = []
print(f'Empty List:\n\t {list}')
for x in range(0,3):
    name = str(input(f'Enter Name For {x+1}: '))
    age = int(input(f'Enter Age For {x+1}: '))
    #pust data in end
    list.append([name,age])
print(f'List:\n\t {list}')
#push data in start
list.insert(0,["Zohaib Malik",19])
print(f'List:\n\t {list}')
#removing data for end
print(f'Removing element from end: {list.pop()}')
#removing data for start
print(f'Removing element from start: { list.pop(0)}')


print(f'Final List:\n\t {list}')
```

```
Empty List:
	 []

Enter Name For 1:  Umer
Enter Age For 1:  22
Enter Name For 2:  Harris
Enter Age For 2:  22
Enter Name For 3:  Kumail
Enter Age For 3:  21

List:
	 [['Umer', 22], ['Harris', 22], ['Kumail', 21]]
List:
	 [['Zohaib Malik', 19], ['Umer', 22], ['Harris', 22], ['Kumail',
21]]
Removing element from end: ['Kumail', 21]
Removing element from start: ['Zohaib Malik', 19]
Final List:
	 [['Umer', 22], ['Harris', 22]]
```

Stack Last IN First OUT:

```
Stack = []

print("Enter 3 Elements in Stack: ")
for x in range(0,3):
    element = int(input())
    #pust data in end
    Stack.append(element)

print(f'Stack:\n\t {Stack}')

while Stack:
    print(Stack.pop())

#removing data for end
print(f'Final Stack:\n\t {Stack}')
```

```
Enter 3 Elements in Stack:

 1
 2
 3

Stack:
	 [1, 2, 3]
3
2
1
Final Stack:
	 []
```

Queue Last IN Last OUT:

```
Queue = []

print("Enter 3 Elements in Queue: ")
for x in range(0,3):
    element = int(input())
    #pust data in end
    Queue.append(element)

print(f'Queue:\n\t {Queue}')

while Queue:
    print(Queue.pop(0))

#removing data for end
print(f'Final Queue:\n\t {Queue}')
```

```
Enter 3 Elements in Queue:

 1
 2
 3

Queue:
	 [1, 2, 3]
1
```

```
2
3
Final Queue:
        []
```

PriorityQueue using `List Sort`

```
PriorityQueue=[]

# insecting elements PriorityQueue
PriorityQueue.append(("Umer",1))
PriorityQueue.append(("Ali",2))
PriorityQueue.append(("Harris",3))

print(f'Priority Queue After Inserting Elements:\n\t {PriorityQueue}')
PriorityQueue.sort(reverse=True)
print("Removing Elements For PriorityQueue: ")
while PriorityQueue:
    print(PriorityQueue.pop(0))

# insecting elements PriorityQueue
PriorityQueue.append((2,"Umer"))
PriorityQueue.append((3,"Ali"))
PriorityQueue.append((1,"Harris"))
PriorityQueue.sort(reverse=True)
print(f'Priority Queue After Inserting Elements:\n\t {PriorityQueue}')

print("Removing Elements For PriorityQueue: ")
while PriorityQueue:
    print(PriorityQueue.pop(0))
```

```
Priority Queue After Inserting Elements:
        [('Umer', 1), ('Ali', 2), ('Harris', 3)]
Removing Elements For PriorityQueue:
('Umer', 1)
('Harris', 3)
('Ali', 2)
Priority Queue After Inserting Elements:
        [(3, 'Ali'), (2, 'Umer'), (1, 'Harris')]
Removing Elements For PriorityQueue:
(3, 'Ali')
(2, 'Umer')
(1, 'Harris')
```

PriorityQueue using heapq

```
import heapq
PriorityQueue=[]
```

```
# insecting elements PriorityQueue
heapq.heappush(PriorityQueue,("Umer",1))
heapq.heappush(PriorityQueue,("Ali",2))
heapq.heappush(PriorityQueue,("Harris",3))

print(f'Priority Queue After Inserting Elements:\n\t {PriorityQueue}')

print("Removing Elements For PriorityQueue: ")
while PriorityQueue:
    print(heapq.heappop(PriorityQueue))

# insecting elements PriorityQueue
heapq.heappush(PriorityQueue,(2,"Umer"))
heapq.heappush(PriorityQueue,(3,"Ali"))
heapq.heappush(PriorityQueue,(1,"Harris"))

print(f'Priority Queue After Inserting Elements:\n\t {PriorityQueue}')

print("Removing Elements For PriorityQueue: ")
while PriorityQueue:
    print(heapq.heappop(PriorityQueue))
```

```
Priority Queue After Inserting Elements:
	 [('Ali', 2), ('Umer', 1), ('Harris', 3)]
Removing Elements For PriorityQueue: 
('Ali', 2)
('Harris', 3)
('Umer', 1)
Priority Queue After Inserting Elements:
	 [(1, 'Harris'), (3, 'Ali'), (2, 'Umer')]
Removing Elements For PriorityQueue: 
(1, 'Harris')
(2, 'Umer')
(3, 'Ali')
```

PriorityQueue using `queue.PriorityQueue`

```
from queue import PriorityQueue

PQueue = PriorityQueue()

# insecting elements PriorityQueue
PQueue.put(("Umer",1))
PQueue.put(("Ali",2))
PQueue.put(("Harris",3))

print("Removing Elements For PriorityQueue: ")
while PQueue:
    print(PQueue.get())
```

```python
# insecting elements PriorityQueue
PQueue.put((2,"Umer"))
PQueue.put((3,"Ali"))
PQueue.put((1,"Harris"))



print("Removing Elements For PriorityQueue: ")
while PQueue:
    print(PQueue.get())
```

```
Removing Elements For PriorityQueue:
('Ali', 2)
('Harris', 3)
('Umer', 1)
```

BFS

```python
# Graph represented as adjacency list
graph = {
    'A': ['B', 'C'],
    'B': ['D', 'E'],
    'C': ['F'],
    'D': [],
    'E': ['F'],
    'F': []
}

def bfs(graph, start):
    visited = set()
    queue = [start]

    while queue:
        vertex = queue.pop(0)
        if vertex not in visited:
            visited.add(vertex)
            print(vertex,end =" ")

            for neighbor in graph[vertex]:
                if neighbor not in visited:
                    queue.append(neighbor)

# Test the BFS function
bfs(graph, 'A')
```

```
A B C D E F
```

DFS

```python
# Graph represented as adjacency list
graph = {
    'A': ['B', 'C'],
    'B': ['D', 'E'],
    'C': ['F'],
    'D': [],
    'E': ['F'],
    'F': []
}

def dfs(graph, start, visited=None):
    if visited is None:
        visited = set()
    visited.add(start)
    print(start,end=" ")

    for neighbor in graph[start]:
        if neighbor not in visited:
            dfs(graph, neighbor, visited)

# Test the DFS function
dfs(graph, 'A')
```

```
A
B
D
E
F
C
```