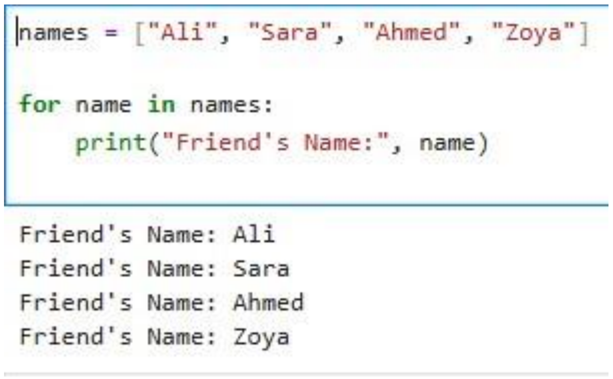
Lab 10

LIST, TUPLE, DICTIONARY, CLASS OBJECT AND IMPLEMENT PRIORITY QUEUE

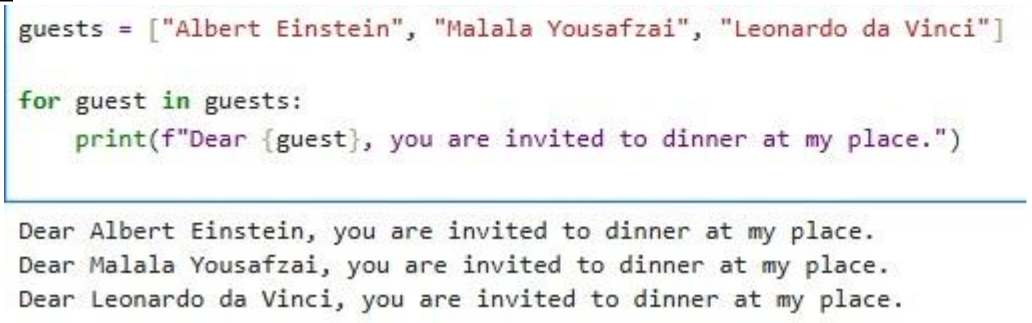
Task 01:



```python
names = ["Ali", "Sara", "Ahmed", "Zoya"]

for name in names:
    print("Friend's Name:", name)
```

```
Friend's Name: Ali
Friend's Name: Sara
Friend's Name: Ahmed
Friend's Name: Zoya
```

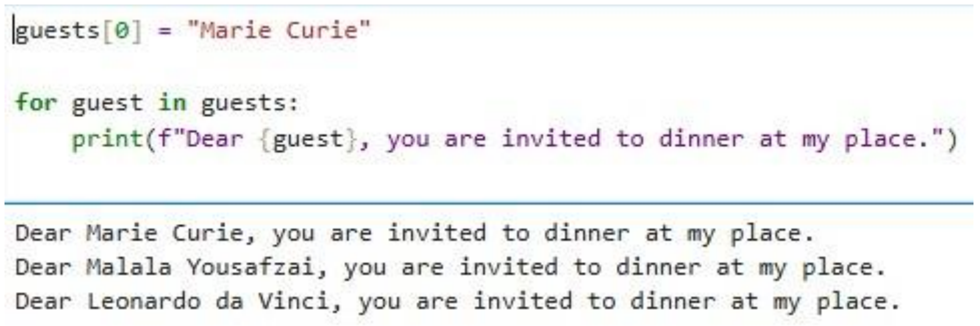
Task 02:



```python
guests = ["Albert Einstein", "Malala Yousafzai", "Leonardo da Vinci"]

for guest in guests:
    print(f"Dear {guest}, you are invited to dinner at my place.")
```

```
Dear Albert Einstein, you are invited to dinner at my place.
Dear Malala Yousafzai, you are invited to dinner at my place.
Dear Leonardo da Vinci, you are invited to dinner at my place.
```

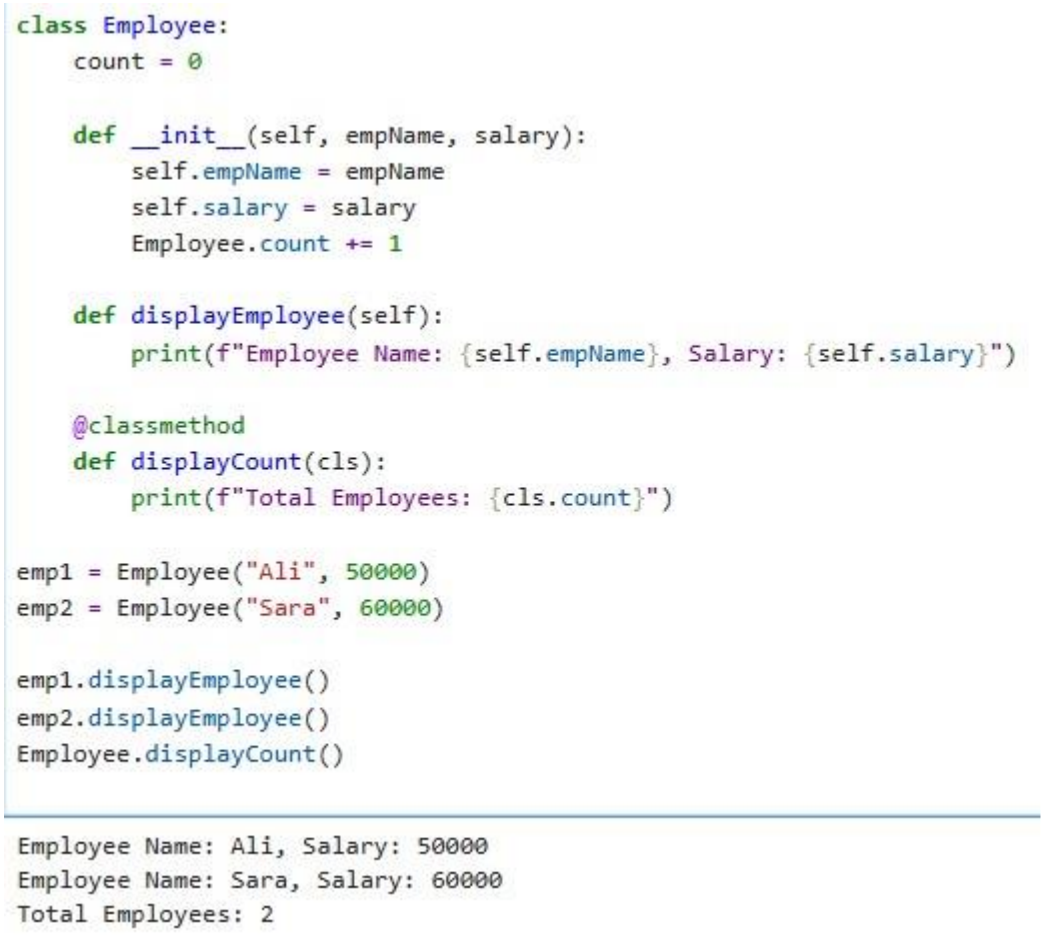
Task 03:



```python
guests[0] = "Marie Curie"

for guest in guests:
    print(f"Dear {guest}, you are invited to dinner at my place.")
```

```
Dear Marie Curie, you are invited to dinner at my place.
Dear Malala Yousafzai, you are invited to dinner at my place.
Dear Leonardo da Vinci, you are invited to dinner at my place.
```

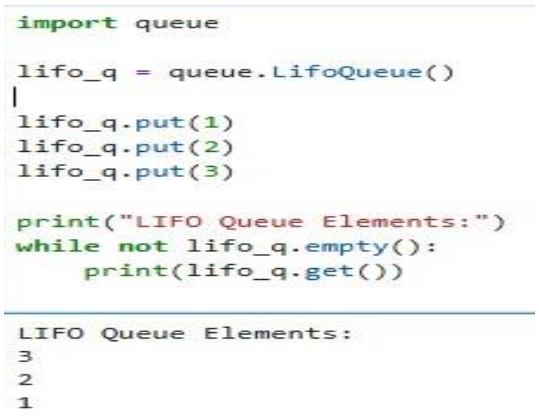
Task 04:



```
class Employee:
    count = 0

    def __init__(self, empName, salary):
        self.empName = empName
        self.salary = salary
        Employee.count += 1

    def displayEmployee(self):
        print(f"Employee Name: {self.empName}, Salary: {self.salary}")

    @classmethod
    def displayCount(cls):
        print(f"Total Employees: {cls.count}")

emp1 = Employee("Ali", 50000)
emp2 = Employee("Sara", 60000)

emp1.displayEmployee()
emp2.displayEmployee()
Employee.displayCount()
```

```
Employee Name: Ali, Salary: 50000
Employee Name: Sara, Salary: 60000
Total Employees: 2
```

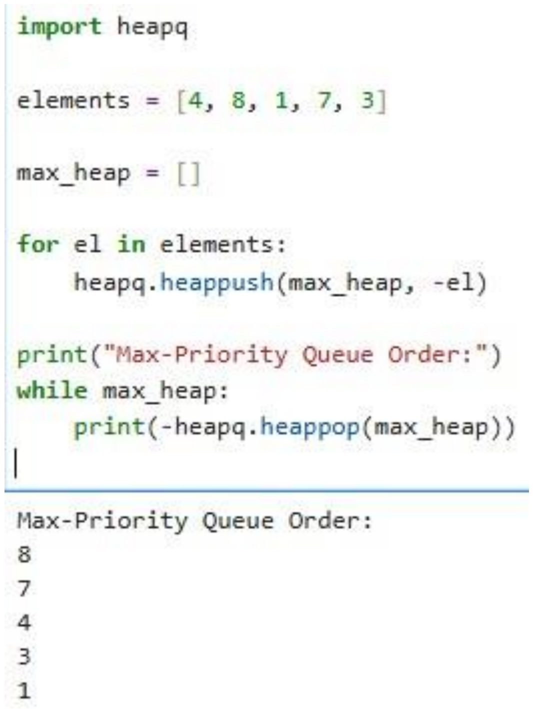
Task 05:



```
import queue

lifo_q = queue.LifoQueue()

lifo_q.put(1)
lifo_q.put(2)
lifo_q.put(3)

print("LIFO Queue Elements:")
while not lifo_q.empty():
    print(lifo_q.get())
```

```
LIFO Queue Elements:
3
2
1
```

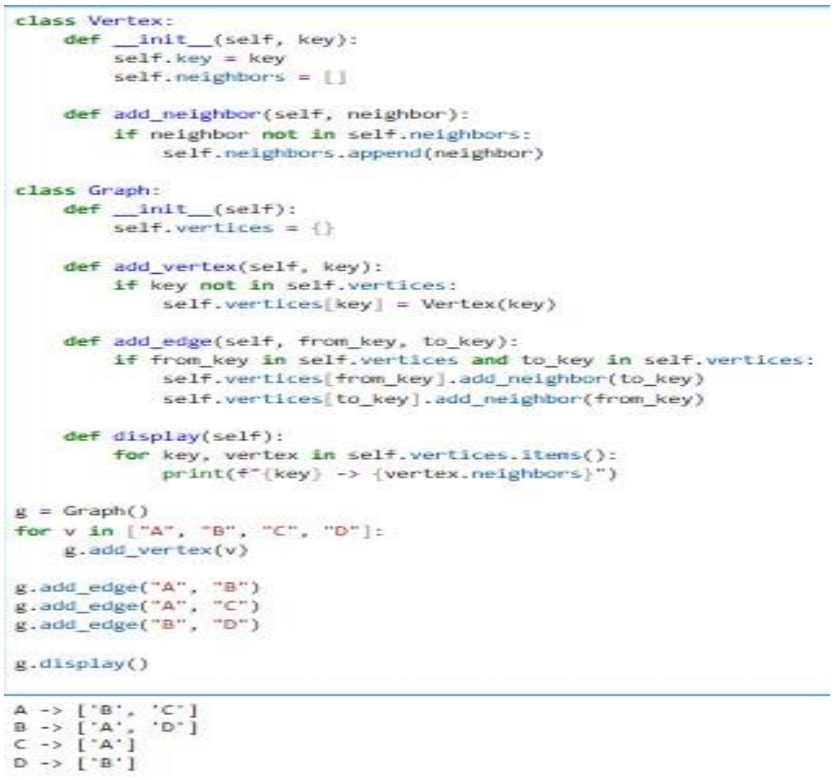
Task 06:



```python
import heapq

elements = [4, 8, 1, 7, 3]

max_heap = []

for el in elements:
    heapq.heappush(max_heap, -el)

print("Max-Priority Queue Order:")
while max_heap:
    print(-heapq.heappop(max_heap))
```

```
Max-Priority Queue Order:
8
7
4
3
1
```

Task 07:



```python
class Vertex:
    def __init__(self, key):
        self.key = key
        self.neighbors = []

    def add_neighbor(self, neighbor):
        if neighbor not in self.neighbors:
            self.neighbors.append(neighbor)

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

    def add_vertex(self, key):
        if key not in self.vertices:
            self.vertices[key] = Vertex(key)

    def add_edge(self, from_key, to_key):
        if from_key in self.vertices and to_key in self.vertices:
            self.vertices[from_key].add_neighbor(to_key)
            self.vertices[to_key].add_neighbor(from_key)

    def display(self):
        for key, vertex in self.vertices.items():
            print(f"{key} -> {vertex.neighbors}")

g = Graph()
for v in ["A", "B", "C", "D"]:
    g.add_vertex(v)

g.add_edge("A", "B")
g.add_edge("A", "C")
g.add_edge("B", "D")

g.display()
```

```
A -> ['B', 'C']
B -> ['A', 'D']
C -> ['A']
D -> ['B']
```