(1)
```
import time
import random
DIRECTIONS = {"NORTH":0, "SOUTH":1, "EAST":2, "WEST":3}
traffic_data = {direction:{"intensity":0, "light": false} for
    direction in DIRECTIONS}

def sense_traffic():
  for direction in DIRECTIONS:
    traffic_data[direction]["intensity"] = random.randint(0,1)
def manage_traffic():
  max_traffic = max(data["intensity"] for data in trafficdata.
                                                values())

  for direction in DIRECTIONS:
    if traffic_data[direction]["intensity"] == max_traffic:
      traffic_data[direction]["light"] = True

    else:
      traffic_data[direction]["light"] = False

  if max_traffic == 0:
    print("Traffic clear in all directions; Proceed")
  else:
    for direction in DIRECTIONS:
      if traffic_data[direction]["light"]:
        print(f"Traffic in {direction} direction is
        significant. Giving green light")
        break

def main():
  t = int(input("How many seconds to simulate traffic:"))
  while t:
    sense_traffic()
    manage_traffic()
    time.sleep(2)
    t -= 1
if __name__ == "__main__":
  main()
```

ASSIGNMENT-4

Output

How many seconds to simulate traffic : 6

Traffic in SOUTH direction is significant. Giving green light
Traffic in NORTH direction is significant. Giving green light
Traffic clear in all directions, Proceed.
Traffic in NORTH direction is significant. Giving green light.
Traffic in SOUTH direction is significant. Giving green light.
Traffic in EAST direction is significant. Giving green light.

(2)
```
import random
class Elevator:
    def __init__(self, n):
        self.n = n
        self.c = 1
        self.d = "UP"
    def move(self):
        if self.d == "UP":
            self.c += 1
        else:
            self.c -= 1
    def process_request(self, c, df):
        print(f"Elevator is on floor {self.c}")
        print(f"Request from {c} to go to {df}")
        fm = 0
        if self.c < c:
            self.d = "UP"
            while self.c < c:
                self.move()
                fm += 1
        elif self.c > c:
            self.d = "DOWN"
            while self.c > c:
                self.move()
                fm += 1
        print(f"Elevator moved {fm} floors {self.d}")
        print(f"Elevator has arrived at floor {c}")
        self.d = "UP" if df > c else "DOWN"
        fm = 0
        while self.c != df:
            self.move()
            fm += 1
```

```
print(f"Elevator moved {fm} floors {self.d}.")
print(f"Elevator has arrived at floor {df}")
```

~~class~~

```
@classmethod
def handle_requests(cls, n, r):
    elevator = cls(n)

    for _ in range(r):
        c = random.randint(1, n)
        df = random.randint(1, n)
        elevator.process_request(c, df)

if __name__ == "__main__":
    n = 10
    r = 3
    Elevator.handle_requests(n, r)
```

OUTPUT

Elevator is on floor 1
Request from 6 to go to 9
Elevator moved 5 floors up.
Elevator arrived at 6.
Elevator moved 3 floors up
Elevator arrived at floor 9
Elevator is on floor 9
Request from floor 1 to go to 1
Elevator moved 8 floorsdown.
Elevator arrived at floor 1
Elevator is on floor 1.

PEAS parameters for an intelligent elevator system -

(1) Performance - It depends on the average waiting time of passengers to get on elevator. Also time taken by elevator to reach its distination floor. Efficiency in terms of power consumption.

(2) Environment - It depends on how many floors does that particular building have. It also depends on the capacity of the elevator. It also depends on particular time period (peak hours)

(3) Actuators - The opening and closing of the doors of the elevator. Motors moving the elevator up and down. Displaying screens and buttons.

(4) Sensors - Floor sensors can be used to detect elevator's current floor position. Weight sensors can be used to check the weight of the elevator Infared sensors to check if passengers or any obstacle present at the door.