```
from itertools import combinations
class Person:
        def __init__(self, name, mass):
        self.name = name
        self.mass = mass

side_1 = [Person("Roman", 90),
        Person("Verlyn", 80),
        Person("Lloyd", 60),
        Person("Robin", 40),
        Person("Supplies", 20)]
side_2 = []

class Boat:
        def __init__(self, boat_limit):
        self.boat_limit = boat_limit
        self.boat = []

        def boarding(self, passengers):
        if sum(p.mass for p in passengers) <= self.boat_limit:
        for p in passengers:
                if p in side_1:
                side_1.remove(p)
                elif p in side_2:
                side_2.remove(p)
                self.boat.append(p)
        print(f"{', '.join(p.name for p in passengers)} boarded the boat.")
        else:
        print("Overweight! Cannot board these passengers.")
        print("")

        def leave_boat(self, position, side):
        if self.boat:
        if position < len(self.boat):
                passenger = self.boat.pop(position)
                if side == 1:
                side_1.append(passenger)
                print(f"{passenger.name} moved to Side 1.")
                else:
                side_2.append(passenger)
                print(f"{passenger.name} moved to Side 2.")
        else:
                print("Invalid position. No such passenger in the boat.")
        else:
```

```
        print("The boat is empty, no one to leave.")
        print("")

        def show_boat(self):
        if self.boat:
        print("Boat is carrying:")
        for p in self.boat:
                print(f"{p.name} ({p.mass} kg)")
        else:
        print("Boat is empty.")
        print("")

        def show_side_1(self):
        if side_1:
        print("Side 1 has:")
        for p in side_1:
                print(f"{p.name} ({p.mass} kg)")
        else:
        print("Side 1 is empty.")
        print("")

        def show_side_2(self):
        if side_2:
        print("Side 2 has:")
        for p in side_2:
                print(f"{p.name} ({p.mass} kg)")
        else:
        print("Side 2 is empty.")
        print("")

def solving():
  while len(side_1) == 0:
        boat.boarding(side_1[n],side)

boat = Boat(100)
boat.show_side_1()
boat.show_boat()
boat.show_side_2()
boat.boarding([side_1[3], side_1[4]])
boat.show_side_1()
boat.show_boat()
boat.show_side_2()
boat.leave_boat(1, 2)
boat.show_side_1()
```

```
boat.show_boat()
boat.show_side_2()
print(f"Robin return boat to side 1\n")
boat.boarding([side_1[2]])
boat.show_side_1()
boat.show_boat()
boat.show_side_2()
boat.leave_boat(1, 2)
boat.show_side_1()
boat.show_boat()
boat.show_side_2()
print("Robin return boat to side 1")
boat.leave_boat(0, 1)
boat.boarding([side_1[1]])
boat.show_side_1()
boat.show_boat()
boat.show_side_2()
boat.leave_boat(0, 2)
boat.boarding([side_2[1]])
boat.show_side_1()
boat.show_boat()
boat.show_side_2()
print(f"Lloyd return boat to side 1\n")
boat.boarding([side_1[1]])
boat.show_side_1()
boat.show_boat()
boat.show_side_2()
boat.leave_boat(0, 2)
print(f"Robin return boat to side 1\n")
boat.leave_boat(0, 1)
boat.boarding([side_1[0]])
boat.show_side_1()
boat.show_boat()
boat.show_side_2()
boat.leave_boat(0, 2)
boat.boarding([side_2[2]])
boat.show_side_1()
boat.show_boat()
boat.show_side_2()
print("Robin return boat to side 1\n")
boat.boarding([side_1[0]])
boat.show_side_1()
boat.show_boat()
boat.show_side_2()
```

```
boat.leave_boat(0, 2)
boat.leave_boat(0, 2)
boat.show_side_1()
boat.show_boat()
boat.show_side_2()
```

Google colab link:

https://colab.research.google.com/drive/1OAop4JeodgQ88k6gdLrlUiZt_m_Ho_xy?usp=sharing