Class ParkingLot:

Def \_\_init\_\_(self, capacity):

Self.capacity = capacity

Self.spaces = [None] \* capacity

Def is\_full(self):

Return None not in self.spaces

Def park(self, vehicle):

If self.is\_full():

Print(“Sorry, the parking lot is full.”)

Return

Index = self.spaces.index(None)

Self.spaces[index] = vehicle

Print(f”Vehicle {vehicle} parked at space {index + 1}.”)

Def leave(self, space\_number):

If space\_number < 1 or space\_number > self.capacity:

Print(“Invalid space number.”)

Return

If self.spaces[space\_number – 1] is None:

Print(f”There is no vehicle parked at space {space\_number}.”)

Return

Vehicle = self.spaces[space\_number – 1]

Self.spaces[space\_number – 1] = None

Print(f”Vehicle {vehicle} left space {space\_number}.”)

Def available\_spaces(self):

Return self.capacity – self.spaces.count(None)

# Create a parking lot with 5 spaces

Parking\_lot = ParkingLot(5)

# Park vehicles

Parking\_lot.park(“ABC123”) # Vehicle ABC123 parked at space 1.

Parking\_lot.park(“XYZ789”) # Vehicle XYZ789 parked at space 2.

Parking\_lot.park(“DEF456”) # Vehicle DEF456 parked at space 3.

# Check available spaces

Print(parking\_lot.available\_spaces()) # 2

# Try to park another vehicle (parking lot is full)

Parking\_lot.park(“GHI789”) # Sorry, the parking lot is full.

# Vehicle leaves

Parking\_lot.leave(2) # Vehicle XYZ789 left space 2.

# Park another vehicle

Parking\_lot.park(“GHI789”) # Vehicle GHI789 parked at space 2.

# Check available spaces

Print(parking\_lot.available\_spaces()) # 1