**SMART PARKING SYSTEM**

**SORUCE CODE:**

class ParkingSpace:  
    def \_\_init\_\_(self, id):  
        [self.id](http://self.id/) = id  
        self.available = True  
  
class SmartParkingSystem:  
    def \_\_init\_\_(self, num\_spaces):  
        self.spaces = [ParkingSpace(i) for i in range(1, num\_spaces + 1)]  
  
    def display\_available\_spaces(self):  
        print("Available Parking Spaces:")  
        for space in self.spaces:  
            if space.available:  
                print(f"Space {[space.id](http://space.id/)}")  
  
    def allocate\_space(self):  
        for space in self.spaces:  
            if space.available:  
                space.available = False  
                return [space.id](http://space.id/)  
        return -1  
  
    def free\_space(self, space\_id):  
        for space in self.spaces:  
            if [space.id](http://space.id/) == space\_id:  
                space.available = True  
                return  
        print(f"Invalid space ID {space\_id}")  
  
    def find\_shortest\_space(self):  
        shortest\_space = None  
        shortest\_distance = float('inf')  
  
        for space in self.spaces:  
            if space.available:  
                # In this simplified example, we assume distance is equal to space ID  
                if [space.id](http://space.id/) < shortest\_distance:  
                    shortest\_space = space  
                    shortest\_distance = [space.id](http://space.id/)  
  
        return shortest\_space  
  
# Example usage:  
if \_\_name\_\_ == "\_\_main\_\_":  
    parking\_system = SmartParkingSystem(5)  
  
    # Display available spaces  
    parking\_system.display\_available\_spaces()  
  
    # Allocate a space  
    allocated\_space = parking\_system.allocate\_space()  
    if allocated\_space != -1:  
        print(f"Allocated Space {allocated\_space}")  
    else:  
        print("No available spaces")  
  
    # Display available spaces after allocation  
    parking\_system.display\_available\_spaces()  
  
    # Find the shortest space from the entrance  
    shortest\_space = parking\_system.find\_shortest\_space()  
    if shortest\_space:  
        print(f"The shortest parking space is Space {[shortest\_space.id](http://shortest_space.id/)}")  
    else:  
        print("No available spaces")  
  
    # Free the allocated space  
    parking\_system.free\_space(allocated\_space)  
  
    # Display available spaces after freeing  
    parking\_system.display\_available\_spaces()

