```
#############################################################################
#########
#############################################################################
#########
# Defining classes
#############################################################################
#########
#############################################################################
#########

class CARPARK:
  def __init__(self,buildingname,parkingslot,price):
    self.bname=buildingname
    self.slot=parkingslot
    self.price=price

#############################################################################
#########
#############################################################################
#########
# Database
#############################################################################
#########
#############################################################################
#########

carpark1=CARPARK("NCP London Bloomsbury Square Car Park",168,"$10")
carpark2=CARPARK("Waterloo Car Park",153,"$10")
carpark3=CARPARK("The Mayfair Car Park",124,"$15")

#############################################################################
#########
#############################################################################
#########
# Parking System Logic
#############################################################################
#########
#############################################################################
#########

def detectobject():
    objectlist=["empty","dynamic","static"]
    obj=999
    while obj > len(objectlist):
      obj=int(input("Scenario: Any obstdacle in the slot now [0:Empty 1:Moving
object (Human/Animal) 2:Static object]?"))
    return objectlist[obj]

#############################################################################
#########
#############################################################################
#########
# Main program
# Step 1: Choose carpark
```

```
#         Variable used: icp, j, pk, cp
#################################################################################
#########
#################################################################################
#########


for icp in range(30): print("\n")
carparklist=[carpark1, carpark2, carpark3]
j=0
for pk in carparklist:
  print(j, pk.bname, "-", pk.price,"per hour")
  j=j+1

cp=999
while cp > j:
  print("====================================================================")
  cp=int(input("Please select the carpark ?"))
  print("====================================================================")
selectpark=carparklist[cp]
print("====================================================================")
print("\nBooking selected carpark at ",selectpark.bname)
print("The car is going to ",selectpark.bname)
print("The car arrived the carpark slot", selectpark.slot)
print("====================================================================")

obstdacle_object = False
while obstdacle_object != "empty" :
  obstdacle_object = detectobject()
  if obstdacle_object == "dynamic" :
    print("====================================================================")
    print("The carpark slot has a person, an animal, or anything else, wait for
clearing")
    print("====================================================================")
  if obstdacle_object == "static":
    print("====================================================================")
    print("Call building management and notice the driver. Wait for clearing")
    print("====================================================================")

print("====================================================================")
print("Car is parked. Informed the driver")
print("Program end")
print("====================================================================")
print("====================================================================")
```

```
#########################################################################################
#########
#Module: Start a car
#########################################################################################
#########

print(" ")
print("======================== SAFETY CHECK EBGINS ========================")


#########################################################################################
#########
#########################################################################################
#########
# Defining classes
#########################################################################################
#########
#########################################################################################
#########

#=========================================================================================
=======#
#PART A: Defining the class CAR - Seting car details, especially the weight
threshold for an adult
#=========================================================================================
=======#
class CAR:
  def __init__(self, brand, model, year, driver_verifier, gps_system,
adult_threshold, mpassword, mpin, baseline_fuel):
    self.brand = brand
    self.model = model
    self.year = year
    self.driver_verifier = driver_verifier
    self.gps_system = gps_system
    self.adult_threshold = adult_threshold
    self.mpassword = mpassword
    self.mpin=mpin
    self.baseline_fuel = baseline_fuel

  def __str__(self):
    return f"Brand: {self.brand}, Model: {self.model}, Year of Manufacturing:
{self.year}\n Driver Identifying Agent: {self.driver_verifier}, GPS and traffic
watch by: {self.gps_system}\n Default Adult Weight threshold:
{self.adult_threshold}\n Manufactuer's driver password: {self.mpassword},
Manufacturer's pin for exceptions: {self.mpin},\n Baseline Level of Fuel to keep
it moving for 10 km: {self.baseline_fuel}"

#=========================================================================================
=======#
#PART B: Defining the class USER & sub-class DRIVER - Registering driver
identification
#=========================================================================================
=======#
class USER():
```

```
    def __init__(self, user_name, user_age, user_weight,user_seatNum):
        self.user_name = user_name
        self.user_age = user_age
        self.user_weight = user_weight
        self.user_seatNum = user_seatNum

    def displayU(self):
        print("The user details are:")
        print("User Name:", self.user_name)
        print("User Age:", self.user_age)
        print("User weight", self.user_weight)
        print("User seat:", self.user_seatNum)

# subclass
class DRIVER(USER):
    def __init__(self, driver_name, driver_age, driver_license, driver_weight,
dpassword,dpin,d_consumption,d_millage, h_add, o_add):
        USER.__init__(self, driver_name, driver_age, driver_weight,"Driver")
        self.driver_license = driver_license
        self.driver_weight = driver_weight
        self.dpassword = dpassword
        self.dpin = dpin
        self.d_consumption = d_consumption
        self.d_millage = d_millage
        self.h_add = h_add
        self.o_add = o_add

    def displayD(self):
        print("Details of the driver are:")
        USER.displayU(self)
        print("License:", self.driver_license)
        print(self.user_name, "'s usual weight: ", self.driver_weight)
        print("Pre-set password of", self.user_name, ": ", self.dpassword, ",
Pin for skipping safty test of",self.user_name,":", self.dpin)
        print("Total consumption to-date of ", self.user_name,": ",
self.d_consumption, ", Total millage to-date before this trip
of",self.user_name,": ", self.d_millage)
        print(driver1.user_name,"'s home address is :", self.h_add, ",",
driver1.user_name, "'s office address is:", self.o_add)


#=============================================================================
=======#
#PART C: Defining the class SEAT - for safety check function
#(a) child safty check (b) child lock function (c) door lock check (d) safty
belt check
#=============================================================================
=======#

class SEAT:
  def __init__(self,seat_num,seat_weight,door_status,childlock,belt_status):
    self.seat_num = seat_num
    self.seat_weight = seat_weight
    self.door_status = door_status
```

```
    self.childlock = childlock
    self.belt_status = belt_status

  def __str__(self):
    return f"Seat Number: {self.seat_num}, Weight detected:
{self.seat_weight},\nDoor Status: {self.door_status}, Child Lock Enabledness :
{self.childlock},\nSeat Belt Status: {self.belt_status}"


#=============================================================================
=======#
#PART D: Defining the class FUEL CONTROL SYSTEM
#=============================================================================
=======#
class FUELSYS:
  def __init__(self, engine_model, cfuel_reading, ffuel_reading, n_consumption,
n_millage, ttl_consumption, ttl_millage, warning_level):
    self.engine_model = engine_model
    self.cfuel_reading = cfuel_reading
    self.ffuel_reading = ffuel_reading
    self.n_consumption = n_consumption
    self.n_millage = n_millage
    self.ttl_consumption = ttl_consumption
    self.ttl_millage = ttl_millage
    self.warning_level = warning_level

  def __str__(self):
   return f"Engine Model: {self.engine_model},\n Current Fuel Reading:
{self.cfuel_reading}, Full Fuel Reading:, {self.ffuel_reading},\n Coming Trip
Fuel Consumption, {self.n_consumption},\n Coming Trip Millage,
{self.n_millage},\n Total To-date Fuel Consumpted Before the Comming Trip:
{self.ttl_consumption},\n Total To-date Millage of the Car Before the Comming
Trip: {self.ttl_millage}\n Fuel Refill Warning Level : {self.warning_level}"

#=============================================================================
=======#
#PART E: Defining the class ROUTE
#=============================================================================
=======#
class ROUTE:
  def __init__(self, r_time, r_millage, r_consumption):
    self.r_time = r_time
    self.r_millage = r_millage
    self.r_consumption = r_consumption

  def __str__(self):
   return f"Route's time required: {self.r_time},\n Route's millage:
{self.r_millage},\n Route's fuel consumpted: {self.r_consumption}"

##############################################################################
#########
##############################################################################
#########
# PART F: Defining class CARPARK
```

```
#################################################################################
#########
#################################################################################
#########

class CARPARK:
  def __init__(self,buildingname,parkingslot,price):
    self.bname=buildingname
    self.slot=parkingslot
    self.price=price

#################################################################################
#########
#################################################################################
#########
# Database
#################################################################################
#########
#################################################################################
#########

#================================================================================
=======#
# SECTION 1: Database of Car1
#================================================================================
=======#
car1 = CAR("Tesla","Model Dream Car","2022", "IDnow","Carmenta TrafficWatch",
45, 0000,0,0.05)

# Database testing:
#print("================== The default car attributes are ================")
#print("The current car is:\n", car1)
#print("================================================================")


#================================================================================
=======#
# SECTION 2: Database of driver1 identity
#            Veriable ued: pw = password of the driver, pin = pin of the driver
#================================================================================
=======

pw = "1234" # Driver's pre-set password
pin = "1" # Driver's pre-set pin
driver1=DRIVER("John", 30, "LX123-555808", 39, pw, pin, 6000, 30000, "2 Happy
Grove, London SW6 1AB", "1 Rainbow Street, London, NW1 1AB")  # creating object
of subclass
#passenage1=USER("Mary", 8, 20,"P1")
#passenage3=USER("David", 20, 20,"P3")
#passlist=[passenage1, passenage3]

# Database testing:
#print("================================================================")
#driver1.displayD()
```

```
#print("====================================================================")


#=============================================================================
=======#
# SECTION 3: Database of the weight detected by the weight sensor and other seat
details
#            Veriable: x, p1, p2, p3 = dedected weight of the driver, passenager
1, passenager 2, and passenager 3
#=============================================================================
=======#
unlock = 0
lock =1
disable=0
enable=1
unbuckle = 0
buckle = 1

#Detected data:
x=38
p1=20
p2=0
p3=10

driver_seat = SEAT("Driver",x,lock,disable,buckle)
p_seat1 = SEAT("P1",p1,lock,disable, buckle)
p_seat2 = SEAT("P2",p2,lock,disable, unbuckle)
p_seat3 = SEAT("P3",p3,lock,disable, unbuckle)

Pseat=[driver_seat,p_seat1,p_seat2,p_seat3]

# Database testing:
#print("====================================================================")
#for s in Pseat:
#  print(s)
#print("====================================================================")


#=============================================================================
=======#
# SECTION 4: Database of the fuel control system
#            Variable: tconsumption = total consumption of this car to-date
before this trip
#                      tmillage = total millage of this car to-dat before this
trip
#=============================================================================
=======#

tconsumption = 10000
tmillage = 50000
nconsumption = 0
nmillage = 0
cfuel=50
ffuel=100
```

```
fuel1 = FUELSYS("Model Future", cfuel, ffuel,nconsumption, nmillage,
tconsumption, tmillage,20)

print("====================+================================================")
print("The current fule system data are:\n", fuel1)
print("====================================================================")


#=============================================================================
=======#
# SECTION 5: Database of Carpark
#            Variable: tconsumption = total consumption of this car to-date
before this trip
#                      tmillage = total millage of this car to-dat before this
trip
#=============================================================================
=======#

carpark1=CARPARK("NCP London Bloomsbury Square Car Park",168,"$10")
carpark2=CARPARK("Waterloo Car Park",153,"$10")
carpark3=CARPARK("The Mayfair Car Park",124,"$15")



###############################################################################
#########
###############################################################################
#########
# OPERATION 1: Safety System Logic
###############################################################################
#########
###############################################################################
#########

#=============================================================================
=======#
# Useful functions: (1) empty lines
#                   Variable used: expw = exception password keyed in
#=============================================================================
=======#
def lines():
  for i in range(3):
    print("")


#=============================================================================
=======#
# Useful functions: (2) waiting time display
#                    Veriable used: i = time
#=============================================================================
=======#
def waiting():
  for i in range(5):
```

```
    print(".")
    for i in range(1000):
      for i in range(10000):
        i=i+1

#=============================================================================
=======#
# Step 1: Driver's verification:
#=============================================================================
=======#
print("**********************************************************************")
print("****************** OPERATION 1: STARTING A CAR ******************")
print("**********************************************************************")
lines()
# 3 methods are used to verify the driver:

# (1) check driving license
print("======================================================================")
print("============== IDENTITY VERIFICATION STARTED ====================")
print("======================================================================")
print("please put driving license on the detector")
print("detecting...",end='')
waiting()
print("Driving license validated :)")

# (2) match outlook of the driver
print("======================================================================")
print("put your face in the circle on the detector")
print("confirming...")
waiting()
print("Face validated :>")
# activated detector
# detector recognised the driving license number and face, and send data to the
license verifier
# result return by the verifier and assume the returned value is Correct

# (3) check driver's password
print("======================================================================")
print("")
kpw = input("Please enter PASSWORD [Hint: please refer to README]: " )
print("")
while kpw != pw:
    print("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!")
    print("!!!!!!!!!!!!!! VERIFICATION FAILURE !!!!!!!!!!!!!!!!!!!!!!!!")
    print("Incorrect password :<")
    print("Engine module locked: Password not correct!")
    #Engine module locked, engine could not be started
    print("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!")
    print("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!")
    print("Please enter the correct PASSWORD: ")
    kpw = input()
print("======================================================================")
print("================== IDENTITY VERIFIED ============================")
print("======================================================================")
```

```
print("Password correct,")
print("Good day,", driver1.user_name,"! How are you today?")
print(car1.brand, car1.year, car1.model, "is at your service!" )
print("=====================================================================")
print("=====================================================================")
input("Press [Enter] to continue...")
lines()
lines()

#=============================================================================
=======#
# Step 2: Customised driver's adult weight threshold to the registered driver
#         Check if registered driver weight below manufacuter's adult threshold,
the adult threshold weight of driver seat will changed to the registered
driver's weight
#         Veriable used: dt = adult weight threashold for driver
#=============================================================================
=======#
#print("=====================================================================")
#print("The manufacturer defaulted driver weight threshold is:")
#print(car1.adult_threshold)

dt=int(car1.adult_threshold)
if dt>driver1.driver_weight:
     dt = driver1.driver_weight
#print("The revised driver weight threshold is:")
#print(dt)
#print("=====================================================================")

#=============================================================================
=======#
# Step 3a: Check if driver's weight is below the driver's adult weight
threshold:
#         Veriable used: dt = adult weight threshold for driver
#=============================================================================
=======#

print("=====================================================================")
print("================== CHILD SAFETY CHECK STARTED ==================")
print("=====================================================================")
print("Checking driver seat...")
driver_seat.seat_weight = 38
print("Returning driver seat weight sensor data...")
print("Weight detected by driver seat weight sensor:",driver_seat.seat_weight)


if dt > driver_seat.seat_weight:
  print("!!!!!!!!!!!!!!!! IMPORTANT WARNING !!!!!!!!!!!!!!!!!!!!!!!!!")
  print("!!!!!!!!!!!!!! ON DRIVING SEAT USER !!!!!!!!!!!!!!!!!!!!!!!!")
  print("---------------------------------------------------------")
  print("Warning: Child sitting on the driver seat!")
  print("         Engine module locked:")
  print("         Car engine could not be started if a child is sitting on the
driver seat!")
```

```
  #Engine module locked, engine cannot be started
  print("")
  expw = input("If the occupant is NOT a child, please enter the ONE-DIGIT PIN
[Hint: please refer to README]: ")
  while expw != driver1.dpin:

print("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!")
      print("Your ONE-DIGTI PIN is not correct!")
      print("Sorry, engine module locked, engine could not be started!")

print("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!")
      print("Please enter the correct ONE-DIGTI PIN: ")
      expw = input()
  print("Thank you, your ONE-DIGTI PIN is correct! Warning cancelled!")
  print("Tne driver seat weight test passed!")
  print("======================================================================")
else:
  print("======================================================================")
  print("Tne driver seat weight test passed!")
  print("======================================================================")
input("Press [enter] to continue...")
lines()

#================================================================================
=======#
# Step 3b: Check if other passenagers' weight are below the adult weight
threshold:
#         Variable used: pt = adult weight threshold for passenagers,
#                        ex2pw = excption key,
#================================================================================
=======#

pt = int(car1.adult_threshold)
#p_seat1 = SEAT("P1",p1,lock,disable, unbuckle)

print("Checking front passenager seat...")
print("Returning front passenager seat weight sensor data...")
print("Weight detected by front passenager seat weight
sensor:",p_seat1.seat_weight)

if p_seat1.seat_weight >0 and p_seat1.seat_weight < pt:
  print("!!!!!!!!!!!!!!!! IMPORTANT WARNING !!!!!!!!!!!!!!!!!!!!!!!!")
  print("!!!!!!!!!!!!!!!! ON FRONT SEAT USER !!!!!!!!!!!!!!!!!!!!!!!!")
  print("----------------------------------------------------------------")
  print("Warning: Child cannot sit on the front seat!")
  print("         Engine module locked: ")
  print("         Car engine could not be started if a child is sitting on the
front seat!")
  #Engine module locked, engine cannot be started
  print("")
  ex1pw = input("If the occupant is NOT a child, please enter the ONE-DIGIT PIN
[Hint: please refer to README]: ")
  while ex1pw != driver1.dpin:
```

```
print("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!")
      print("Your ONE-DIGTI PIN is not correct!")
      print("Sorry, engine module locked, engine could not be started!")

print("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!")
      print("Please enter the correct ONE-DIGTI PIN: ")
      ex1pw = input()
  print("Thank you, your ONE-DIGTI PIN is correct! Warning cancelled!")
  print("Front passenager seat weight test passed!")
  print("======================================================================")
else:
  print("======================================================================")
  print("Front seat passenage weight test passed!")
  print("======================================================================")
input("Press [Enter] to continue...")
lines()

print("Checking back left seat...")
print("Returning back left seat weight sensor data...")
print("Weight detected by back left seat weight sensor:",p_seat2.seat_weight)

if p_seat2.seat_weight >0 and p_seat2.seat_weight < pt:
  print("======================================================================")
  print("Warm reminder: The seat may be occupied by a child,")
  ex2pw = input("if it is not occupied by a child, please '0' to continue: ")
  if ex2pw  != "0":
     print("Warm reminder: Child seat is suggested for ",s.seat_num,"!")
     print("Childlock enabled!")
     s.childlock = enable
     print("Childlock status: ",s.childlock)

print("======================================================================")
  else:
     print("Reminder cancelled!")
     print("Back left seat weight test passed")

print("======================================================================")
else:
  print("Back left seat weight test passed!")
lines()

print("Checking back right seat...")
print("Returning back right seat weight sensor data...")
print("Weight detected by back right seat weight sensor:",p_seat3.seat_weight)

if p_seat3.seat_weight >0 and p_seat3.seat_weight < pt:
  print("======================================================================")
  print("Warm reminder: The seat may be occupied by a child,")
  print("Childlock enabled")
  print("======================================================================")

  #ex2pw = input("if it is not occupied by a child, please '0' to continue: ")
  #if ex2pw  != "0":
  #    print("Warm reminder: Child seat is suggested for ",s.seat_num,"!")
```

```
  #    print("Childlock enabled!")
  #    s.childlock = enable
  #else:
  #    print("Reminder cancelled!")
  #    print("Back right seat weight test passed!")
  #
print("=====================================================================")
else:
  print("Back right seat weight test passed!")
  print("=====================================================================")
input("Press [Enter] to continue...")
lines()

#=============================================================================
=======#
# Step 4: Check if the doors are closed
#         Variable used: dstatus = statuse of seat door [1 = properly closed]
#=============================================================================
=======#
print("=====================================================================")
print("==================== DOOR CLOSE CHECK STARTED ====================")
print("=====================================================================")

for s in Pseat:
    #print(s.seat_num,"'s door propertly closed? Please enter 1 for yes.")
    dstatus = lock
    s.door_status = dstatus

    while s.door_status != lock:
      print("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!")
      print("!!!!!!!!!!!!!! DOOR NOT PROPERTLY CLOSED !!!!!!!!!!!!!!!!!!!!")
      print("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!")
      print("Warning : ",s.seat_num,"'s door is not properly closed!")
      print("         Engine module locked: Car engine could not be started if
door is not propertly closed!")
      #Engine module locked, engine cannot be started
      print("         Please close the door properly!")
      print("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!")
      print(s.seat_num,"'s door propertly closed? Please enter 1 for yes.")
      dstatus = input()
      s.door_status = dstatus
    print(s.seat_num,"seat's door is locked")
lines()

#=============================================================================
=======#
# Step 5: Check if the safty belt is buckled
#         Variable used: bstatus = statuse of seat belt [1 = properly buckled],
#                        ex3pw = exception key
#=============================================================================
=======#

print("=====================================================================")
print("==================== SEAT BELT CHECK STARTED ====================")
```

```
print("====================================================================")

for s in Pseat:
    if s.seat_weight == 0:
      print("No passenager on the seat", s.seat_num, ", no seat belt is
required")
      continue
    else:
      if s.belt_status == unbuckle:
        if s.seat_num == "Driver":

print("====================================================================")
          print("Warm reminder: Driver's seat belt is not properly buckled,")
          print("               please buckle your seat belt as soon as
practicable!")

print("====================================================================")
        else:
          print("Warm reminder: seat belt of ", s.seat_num,"is not properly
buckled,")
          #ex3pw = input("If the seat is NOT occupied by a human, enter '0' to
continue: ")
          #if ex3pw  == "0":
          #   print("Seat belt reminder cancelled!")
          #
print("====================================================================")
          #else:
          print("               please buckle your seat belt as soon as
practicable!")

print("====================================================================")
      else:
        print(s.seat_num,"'s seat belt is buckled")
input("Press [Enter] to continue...")
lines()

#================================================================================
=======#
# Step 6: Check if fuel is over the baseline level
#         Variable used: cfuel = current fuel reading, ffuel = full fuel
reading, bl = baseline fuel reading
#================================================================================
=======#
# get fuel reading from fuel meter

print("====================================================================")
print("================== BASELINE FUEL CHECK STARTED ====================")
print("====================================================================")

cfuel = int(fuel1.cfuel_reading)
ffuel = int(fuel1.ffuel_reading)
bl = int(car1.baseline_fuel)

if cfuel/ffuel < bl:
```

```
  print("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!")
  print ("Warning: Fuel is below baseline level,")
  print ("         Engine module locked: Car engine could not be started due to
insufficient fuel!")
  #Engine module locked, engine cannot be started
  print ("         Please refill before starting the engine")
  print("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!")
else:
  print("=====================================================================")
  print("Fuel level is above baseline level! Safe to start the engine!")
  print("=====================================================================")

#=============================================================================
=======#
# Step 7: Enable engine module [For the use of next operation]
#         Variable used: e = True = engine module is on
#=============================================================================
=======#
e = True

lines()
print("================== SAFETY CHECK COMPLETED ========================")
print("================== ENGINE MODULE ENABLED ========================")
input("press [Enter] to continue...")
lines()
```

```
#############################################################################
#########
#############################################################################
#########
# OPERATION 2: Navigation System Logic
#############################################################################
#########
#############################################################################
#########

#############################################################################
#########
#############################################################################
#########
# Defining classes
#############################################################################
#########
#############################################################################
#########

#============================================================================
=======#
#PART A: Defining the class CAR - Seting car details, especially the weight
threshold for an adult
#============================================================================
=======#
class CAR:
  def __init__(self, brand, model, year, driver_verifier, gps_system,
adult_threshold, mpassword, mpin, baseline_fuel):
    self.brand = brand
    self.model = model
    self.year = year
    self.driver_verifier = driver_verifier
    self.gps_system = gps_system
    self.adult_threshold = adult_threshold
    self.mpassword = mpassword
    self.mpin=mpin
    self.baseline_fuel = baseline_fuel

  def __str__(self):
    return f"Brand: {self.brand}, Model: {self.model}, Year of Manufacturing:
{self.year}\n Driver Identifying Agent: {self.driver_verifier}, GPS and traffic
watch by: {self.gps_system}\n Default Adult Weight threshold:
{self.adult_threshold}\n Manufactuer's driver password: {self.mpassword},
Manufacturer's pin for exceptions: {self.mpin},\n Baseline Level of Fuel to keep
it moving for 10 km: {self.baseline_fuel}"

#============================================================================
=======#
#PART B: Defining the class USER & sub-class DRIVER - Registering driver
identification
#============================================================================
=======#
class USER():
```

```
    def __init__(self, user_name, user_age, user_weight,user_seatNum):
        self.user_name = user_name
        self.user_age = user_age
        self.user_weight = user_weight
        self.user_seatNum = user_seatNum

    def displayU(self):
        print("The user details are:")
        print("User Name:", self.user_name)
        print("User Age:", self.user_age)
        print("User weight", self.user_weight)
        print("User seat:", self.user_seatNum)

# subclass
class DRIVER(USER):
    def __init__(self, driver_name, driver_age, driver_license, driver_weight,
dpassword,dpin,d_consumption,d_millage, h_add, o_add):
        USER.__init__(self, driver_name, driver_age, driver_weight,"Driver")
        self.driver_license = driver_license
        self.driver_weight = driver_weight
        self.dpassword = dpassword
        self.dpin = dpin
        self.d_consumption = d_consumption
        self.d_millage = d_millage
        self.h_add = h_add
        self.o_add = o_add

    def displayD(self):
        print("Details of the driver are:")
        USER.displayU(self)
        print("License:", self.driver_license)
        print(self.user_name, "'s usual weight: ", self.driver_weight)
        print("Pre-set password of", self.user_name, ": ", self.dpassword, ",
Pin for skipping safty test of",self.user_name,":", self.dpin)
        print("Total consumption to-date of ", self.user_name,": ",
self.d_consumption, ", Total millage to-date before this trip
of",self.user_name,": ", self.d_millage)
        print(driver1.user_name,"'s home address is :", self.h_add, ",",
driver1.user_name, "'s office address is:", self.o_add)


#=============================================================================
=======#
#PART C: Defining the class SEAT - for safety check function
#(a) child safty check (b) child lock function (c) door lock check (d) safty
belt check
#=============================================================================
=======#

class SEAT:
  def __init__(self,seat_num,seat_weight,door_status,childlock,belt_status):
    self.seat_num = seat_num
    self.seat_weight = seat_weight
    self.door_status = door_status
```

```
    self.childlock = childlock
    self.belt_status = belt_status

  def __str__(self):
    return f"Seat Number: {self.seat_num}, Weight detected:
{self.seat_weight},\nDoor Status: {self.door_status}, Child Lock Enabledness :
{self.childlock},\nSeat Belt Status: {self.belt_status}"


#=============================================================================
=======#
#PART D: Defining the class FUEL CONTROL SYSTEM
#=============================================================================
=======#
class FUELSYS:
  def __init__(self, engine_model, cfuel_reading, ffuel_reading, n_consumption,
n_millage, ttl_consumption, ttl_millage, warning_level):
    self.engine_model = engine_model
    self.cfuel_reading = cfuel_reading
    self.ffuel_reading = ffuel_reading
    self.n_consumption = n_consumption
    self.n_millage = n_millage
    self.ttl_consumption = ttl_consumption
    self.ttl_millage = ttl_millage
    self.warning_level = warning_level

  def __str__(self):
   return f"Engine Model: {self.engine_model},\n Current Fuel Reading:
{self.cfuel_reading}, Full Fuel Reading:, {self.ffuel_reading},\n Coming Trip
Fuel Consumption, {self.n_consumption},\n Coming Trip Millage,
{self.n_millage},\n Total To-date Fuel Consumpted Before the Comming Trip:
{self.ttl_consumption},\n Total To-date Millage of the Car Before the Comming
Trip: {self.ttl_millage}\n Fuel Refill Warning Level : {self.warning_level}"

#=============================================================================
=======#
#PART E: Defining the class ROUTE
#=============================================================================
=======#
class ROUTE:
  def __init__(self, r_time, r_millage, r_consumption):
    self.r_time = r_time
    self.r_millage = r_millage
    self.r_consumption = r_consumption

  def __str__(self):
   return f"Route's time required: {self.r_time},\n Route's millage:
{self.r_millage},\n Route's fuel consumpted: {self.r_consumption}"

##############################################################################
#########
##############################################################################
#########
# Database
```

```
#########################################################################################
#########
#########################################################################################
#########
#=========================================================================================
=======#
# SECTION 1: Database of Car1
#=========================================================================================
=======#
car1 = CAR("Tesla","Model Dream Car","2022", "IDnow","Carmenta TrafficWatch",
45, 0000,0,0.05)

#=========================================================================================
=======#
# SECTION 2: Database of driver1 identity
#            Veriable ued: pw = password of the driver, pin = pin of the driver
#=========================================================================================
=======

pw = "1234" # Driver's pre-set password
pin = "1" # Driver's pre-set pin
driver1=DRIVER("John", 30, "LX123-555808", 39, pw, pin, 6000, 30000, "2 Happy
Grove, London SW6 1AB", "1 Rainbow Street, London, NW1 1AB")  # creating object
of subclass
#passenage1=USER("Mary", 8, 20,"P1")
#passenage3=USER("David", 20, 20,"P3")
#passlist=[passenage1, passenage3]

#=========================================================================================
=======#
# SECTION 4: Database of the fuel control system
#            Variable: tconsumption = total consumption of this car to-date
before this trip
#                      tmillage = total millage of this car to-dat before this
trip
#=========================================================================================
=======#

tconsumption = 10000
tmillage = 50000
nconsumption = 0
nmillage = 0
cfuel=50
ffuel=100

fuel1 = FUELSYS("Model Future", cfuel, ffuel,nconsumption, nmillage,
tconsumption, tmillage,20)

print("====================+==============================================")
print("The current fule system data are:\n", fuel1)
print("=================================================================")


#=========================================================================================
```

```
=======#
#Useful function : (3) Routes calculation
# Variable used: two_routes, third_route
# Variable used: fr/nr/sr_time, fr/nr/sr_millage, fr/nr/sr_consumption
#================================================================================
=======#
def two_routes():
  fr = ROUTE(fr_time, fr_millage, fr_consumption) # Fastest Route
  nr = ROUTE(nr_time, nr_millage, nr_consumption) # Nearest Route

  print("The fastest route (Route A) is shown in the map below in red:\n",fr)
  print("")
  print("The nearest route (Route B) is shown in the map below in blue:\n",nr)
  print("")
  return [fr,nr]

def third_routes(routelist):
  sr = ROUTE(sr_time, sr_millage, sr_consumption) # a Route via a power station
  print("The route via a power station (Route C) is shown in the map below in
green:\n", sr)
  routelist.append(sr)


#================================================================================
=======#
# Functions: (4) Checking adequacy of fuel
#            Function name used: checkfuel
#            Variable used: fu = fu required for current action
#================================================================================
=======#
# cfuel = 50 as per above
# ffuel = 100 as per above
#def checkfuel():
#  fu = int(fu)
#  if cfuel - fu < fuel1.warning_level:
#    print ("Warm reminder: Fuel level is low, suggest power refill")
#    print ("====================== The routes available are:
============================")
#    #two_routes()
#    #third_routes()
#  else:
#    two_routes()
#print ("====================== The routes available are:
============================")

#================================================================================
=======#
# Function: (4) Choose routes
#           Variable used: f, r, z
#================================================================================
=======#
def choose_route(routelist):
  fr=routelist[0]
  nr=routelist[1]
```

```
  print("Please choose a route :")
  print("(0) the fastest route (Route A)?")
  print("(1) the nearest route (Route B)?")
  if len(routelist)==3 :
    print("(2) a route via a power station (Route C)?")
    sr = routelist[2]

  maxchoose=len(routelist)-1
  r=999
  while r > maxchoose :
    r = int(input("please enter the appropriate number: "))
    if r == 0:
        print
("================================================================================
")
        print ("Thank you for choosing, going by Route A")
        print ("Time required = ",fr.r_time, "Distance = ", fr.r_millage,
"Expected fuel consumption = ", fr.r_consumption)
        print
("================================================================================
")
        nconsumption = int(fr.r_consumption)
        nmillage = int(fr.r_millage)
        break
    elif r == 1:
        print
("================================================================================
")
        print ("Thank you for choosing, going by Route B")
        print ("Time required = ",nr.r_time, "Distance = ", nr.r_millage,
"Expected fuel consumption = ", nr.r_consumption)
        print
("================================================================================
")
        nconsumption = int(nr.r_consumption)
        nmillage = int(nr.r_millage)
        break
    elif r == 2 and maxchoose ==2:
        print
("================================================================================
")
        print ("Thank you for choosing, going by Route C")
        print ("Time required = ", sr.r_time, "Distance = ", sr.r_millage,
"Expected fuel consumption = ", sr.r_consumption)
        print
("================================================================================
")
        nconsumption = int(sr.r_consumption)
        nmillage = int(sr.r_millage)
        break
    else:
        print("Sorry, you are not entering (1), (2) or (3),")
        print("please enter again, (1)Route A, (2) Route B or (3) Route C?")
```

```
  return routelist[r]



def lines():
  for i in range(3):
    print("")

#=============================================================================
=======#
# Main program
# Step 8: Enter destination and choose a route
#         Variable used: ades1 = actual destination
#=============================================================================
=======#
lines()
lines()
print("**********************************************************************")
print("****************** OPERATION 2: NAVIGATION ***********************")
print("**********************************************************************")
lines()
# Engine moduel activitated after completing safety check

print("======================================================================")
print("================= ENTERING DESTINATION STARTED ====================")
print("======================================================================")

e = True
if e == True:
  while e:
    print ("Where do you want to go, ", driver1.user_name, "? (1) Home, (2)
Office or (3) Anywhere else? (1), (2) or (3)")
    des1 = input()
    if des1 == "1":
      des1 = driver1.h_add
      break
    elif des1 == "2":
      des1 = driver1.o_add
      break
    elif des1 == "3":
      des1 = input("Please enter address: ")
      break
    else:
      print("Error, not entering (1), (2) or (3),")
else:
  print("Sorry, the Engine module is LOCKED, repeat the Operation 'STARTING THE
CAR!'")
  quit()

print("======================================================================")
print("going to", des1, "...")
print("connecting to", car1.gps_system, "...")
print(car1.gps_system, "calculating the suggested routes...")
print ("====================== The routes available are:
```

```
============================")

# Data of available routes in 1st route calculation:
fr_time = 60
fr_millage = 100
fr_consumption = fr_millage/5
nr_time = 70
nr_millage = 90
nr_consumption = nr_millage/5
sr_time = 65
sr_millage = 105
sr_consumption = sr_millage/5

routelist=two_routes()

#checkfuel
fu = max(fr_millage/5, nr_millage/5)
if cfuel - fu < fuel1.warning_level :
    f=1
    print ("Warm reminder: Fuel level is low, additional routes via a power is
suggested")
    third_routes(routelist)
else:
    f=0
print
("================================================================================
")

selectRoute=choose_route(routelist)
wait=input("Press the <Enter> key to continue...")

#================================================================================
=======#
# Step 9: Re-enter and choose a route
#         Variable used: a des1 = actual destination
#================================================================================
=======#
lines()
print ("After driving for 10 minutes")
print ("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!")
print ("Signal from Carmenta TrafficWatch ...")
print ("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!")
print ("There is an accident on the choosen route!")
print ("New routes are caclculated and suggested.")
input("Press [Enter] to continue...")

fr_time = 40
fr_millage = 80
fr_consumption = fr_millage/5
nr_time = 42
nr_millage = 75
nr_consumption = nr_millage/5
sr_time = 45
sr_millage = 80
```

```
sr_consumption = sr_millage/5

newroutelist=two_routes()

#checkfuel
fu = max(fr_millage/5, nr_millage/5)
if cfuel - fu < fuel1.warning_level :
    f=1
    print ("Warm reminder: Fuel level is low, additional routes via a power is
suggested")
    third_routes(newroutelist)
else:
    f=0
print
("================================================================================
")
selectedRoute=choose_route(newroutelist)
input("Press [Enter]to continue...")

#================================================================================
=======#
# Step 10: Updating millage and consumption
#================================================================================
=======#
lines()
lines()
lines()
lines()
print("=====================================================================")
print("========== UPDATING MILLAGE AND FUEL CONSUMPTION STARTED ==========")
print("=====================================================================")

print("Record of millage and fuel consumption before this trip:")
print("Total millage and fuel consumption of the
car:",fuel1.ttl_millage,fuel1.ttl_consumption)
print("Total millage and fuel consumption of", driver1.user_name,
":",driver1.d_millage,driver1.d_consumption)

fuel1.ttl_millage = fuel1.ttl_millage + selectedRoute.r_millage
fuel1.ttl_consumption = fuel1.ttl_consumption + selectedRoute.r_consumption
driver1.d_millage = driver1.d_millage + selectedRoute.r_millage
driver1.d_consumption = driver1.d_consumption + selectedRoute.r_consumption

print("Record of millage and fuel consumption after this trip:")
print("Total millage and fuel consumption of the car:
",fuel1.ttl_millage,fuel1.ttl_consumption)
print("Total millage and fuel consumption of", driver1.user_name,":
",driver1.d_millage,driver1.d_consumption)

print("=================== RECORDS UPDATED ==========================")
print("=====================================================================")
```