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