**Name**: Steve Hommy

**Pair: -**

**Amount of completed tasks: 9**

**Which tasks were left undone or incomplete: 0**

Self-assessment:

This exercise was easy/difficult/ok/etc. for me because…

This exercise was quite ok for me because it was mainly repeating same thing all the time. Doing this exercise, I learned how to call classes from different files and using \_\_str\_\_ method.

## Test report

Write the test report yourself to each coding task (task number, input/action, desired output and then the testing evidence (actual output)). Add rows if necessary. Include answers to theoretical questions and pseudocode to this return document as well in addition to code screen captures. Actual output can be a screen capture of the terminal showing the output.

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Input / action** | **Desired output** | **Actual output (use red color if desired output != actual output)** |
| **4** | User runs the program  <Run the program so that you get the desired output.> | Here is the data that you provided:  Manufacturer: Apple  Model number: Iphone7  Retail price: 500 | Enter manufacturer: Apple  Enter model: Iphone7  Enter retail price: 500  Here is the data that you provided:  Manufacturer: Apple  Model number: Iphone7  Retail price: 500 |
| **5** | User runs the program  <Run the program so that you get the desired output.> | Here is the data that you provided:  Manufacturer: Apple  Model number: Iphone7  Retail price: 500  Id number: 1  Manufacturer: Apple  Model number: Iphone7  Retail price: 500  Id number: 1 | Enter manufacturer: Apple  Enter model: Iphone7  Enter retail price: 500  Enter id number between 1-6: 1  Here is the data that you provided:  Manufacturer: Apple  Model number: Iphone7  Retail price: 500  Id number: 1  Manufacturer: Apple  Model number: Iphone7  Retail price: 500  Id number: 1 |
| **6** | User runs the program  <Write test case depending on your implementation.> | Here is the data that you provided:  Manufacturer: Samsung  Model number: Galaxy S7  Retail price: 600  Id number: 1  Manufacturer: Apple  Model number: Iphone6  Retail price: 500  Id number: 2 | How many cellphones would you like to create? 2  Enter new object name: cellphone1  Enter manufacturer: Samsung  Enter model: Galaxy S7  Enter retail price: 600  Enter new object name: cellphone2  Enter manufacturer: Apple  Enter model: Iphone6  Enter retail price: 500  Here is the data that you provided:  Manufacturer: Samsung  Model number: Galaxy S7  Retail price: 600  Id number: 1  Manufacturer: Apple  Model number: Iphone6  Retail price: 500  Id number: 2 |
| **7** | User runs the program  <Run the program so that you get the desired output.> | Dice number is: 6  Here is the data that you provided:  Manufacturer: Apple  Model number: Iphone5  Retail price: 250  Id number: 1  Manufacturer: Samsung  Model number: Galaxy S7  Retail price: 650  Id number: 2  Manufacturer: Apple  Model number: Iphone6  Retail price: 600  Id number: 3  Manufacturer: OnePlus  Model number: 6T  Retail price: 450  Id number: 4  Manufacturer: OnePlus  Model number: 8  Retail price: 700  Id number: 5  Manufacturer: Apple  Model number: Iphone8  Retail price: 900  Id number: 6  Here is cellphone based on dice roll:  Manufacturer: Apple  Model number: Iphone8  Retail price: 900  Id number: 6 | How many cellphones would you like to create? 6  Enter new object name: cellphone1  Enter manufacturer: Apple  Enter model: Iphone5  Enter retail price: 250  Enter new object name: cellphone2  Enter manufacturer: Samsung  Enter model: Galaxy S7  Enter retail price: 650  Enter new object name: cellphone3  Enter manufacturer: Apple  Enter model number: Iphone6  Enter retail price: 600  Enter new object name: cellphone4  Enter manufacturer: OnePlus  Enter model number: 6T  Enter retail price: 450  Enter new object name: cellphone5  Enter manufacturer: OnePlus  Enter model number: 8  Enter retail price: 700  Enter new object name: cellphone6  Enter manufacturer: Apple  Enter model number: Iphone8  Enter retail price: 900  Dice number is: 6  Here is the data that you provided:  Manufacturer: Apple  Model number: Iphone5  Retail price: 250  Cellphone ID: 1  Manufacturer: Samsung  Model number: Galaxy S7  Retail price: 650  Cellphone ID: 2  Manufacturer: Apple  Model number: Iphone6  Retail price: 600  Cellphone ID: 3  Manufacturer: OnePlus  Model number: 6T  Retail price: 450  Cellphone ID: 4  Manufacturer: OnePlus  Model number: 8  Retail price: 700  Cellphone ID: 5  Model number: Iphone8  Retail price: 900  Cellphone ID: 6  Here is cellphone based on dice roll:  Manufacturer: Apple  Model number: Iphone8  Retail price: 900  Cellphone ID: 6 |
| **8** | User runs the program  <Run the program so that you get the desired output.> | This is your car:  Make: VW  Model: Golf VII  Mileage: 100000 miles  Price: 8500 €  Color: Black  Maximum load limit 1500 Kg  Size of trunk 350 litres | Enter car maker: VW  Enter car model: Golf VII  Enter mileage of the car: 100000  Enter price of the car: 8500  Enter car color: Black  Enter maximum load limit for the car: 1500  Enter size of the trunk for the car: 350  This is your car  Make: VW  Model: Golf VII  Mileage: 100000 miles  Price: 8500 €  Color: Black  Maximum load limit 1500 Kg  Size of trunk 350 litres |
| **9** | User runs the program  <Run the program so that you get the desired output.> | ID: 1  Species: Cat  Name: Angel  Size 2 m  Weight 30 Kg  ID: 2  Species: Dog  Name: Diamond  Size 3 m  Weight 40 Kg  ID: 3  Species: Wolf  Name: Fleur  Size 4 m  Weight 150 Kg  ID: 4  Species: Leopard  Name: Bambi  Size 5 m  Weight 160 Kg  ID: 5  Species: Seal  Name: Flower  Size 3 m  Weight 300 Kg  ID: 6  Species: Monkey  Name: Moony  Size 2 m  Weight 170 Kg  Make: VW  Model: Golf VII  Mileage: 100000 miles  Price: 10000.0 €  Color: Black  Maximum load limit 100 Kg  Size of trunk 350 litres  Dice number is: 4  Here is mammal based on dice roll:  ID: 4  Species: Leopard  Name: Bambi  Size 5 m  Weight 160 Kg  Mammal too big | ID: 1  Species: Cat  Name: Angel  Size 2 m  Weight 30 Kg  ID: 2  Species: Dog  Name: Diamond  Size 3 m  Weight 40 Kg  ID: 3  Species: Wolf  Name: Fleur  Size 4 m  Weight 150 Kg  ID: 4  Species: Leopard  Name: Bambi  Size 5 m  Weight 160 Kg  ID: 5  Species: Seal  Name: Flower  Size 3 m  Weight 300 Kg  ID: 6  Species: Monkey  Name: Moony  Size 2 m  Weight 170 Kg  Make: VW  Model: Golf VII  Mileage: 100000 miles  Price: 10000.0 €  Color: Black  Maximum load limit 100 Kg  Size of trunk 350 litres  Dice number is: 4  Here is mammal based on dice roll:  ID: 4  Species: Leopard  Name: Bambi  Size 5 m  Weight 160 Kg  Mammal too big |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

1. Explain the following terms and what they are used for:

a. Inheritance (in object-oriented programming)

**Inheritance is the procedure in which one class inherits the attributes and methods of another class. The class whose properties and methods are inherited is known as the Parent class. And the class that inherits the properties from the parent class is the Child class**

b. Multiple inheritance

**An inheritance becomes multiple inheritances when a class inherits more than one parent class. The child class after inheriting properties from various parent classes has access to all of their objects.**

c. UML

**Unified Modeling Language, is a standardized modeling language consisting of an integrated set of diagrams, developed to help system and software developers for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems.**

d. UML class diagram

The UML Class diagram is a graphical notation used to construct and visualize object oriented systems. A class diagram in the UML is a type of static structure diagram that describes the structure of a system by showing the systems

2. True or false?

a. The practice of procedural programming is centered on the creation of objects.

**False**

b. Object reusability has been a factor in the increased use of object-oriented programming.

**True**

c. It is a common practice in object-oriented programming to make all of a class’s data attributes accessible to statements outside the class.

**False**

d. Class methods do not have to have a self-parameter.

**False**

e. Starting an attribute name with two underscores will hide the attribute from code outside the class.

**True**

f. You cannot directly call the \_\_str\_\_ method.

**True**

3. Answer the following question: When you model using UML diagrams, why is it important to follow the UML syntax strictly?

**Efficient and appropriate use of notations is very important for making a complete and meaningful model. The model is useless, unless its purpose is depicted properly.**

## Screen captures of all code here

# File name: exercise4.py

# Author: Steve Hommy

# Description: Create a CellPhone Class

class Cellphone:

    def \_\_init\_\_(self):

        self.manufact = ""

        self.model = ""

        self.retail\_price = 0

    def set\_manufact(self):

        self.manufact = str(input("Enter manufacturer: "))

    def set\_model(self):

        self.model = str(input("Enter model: "))

    def set\_retail\_price(self):

        self.retail\_price = int(input("Enter retail price: "))

    def get\_manufact(self):

        return self.manufact

    def get\_model(self):

        return self.model

    def get\_retail\_price(self):

        return self.retail\_price

# File name: exercise4\_main.py

# Author: Steve Hommy

# Description: Main function for CellPhone Class

# Importing Cellphone class into main file

from exercise4 import Cellphone

def main():

    cellphone = Cellphone()

    cellphone.set\_manufact()

    cellphone.set\_model()

    cellphone.set\_retail\_price()

    print("Here is the data that you provided:")

    print("Manufacturer:", cellphone.get\_manufact())

    print("Model number:", cellphone.get\_model())

    print("Retail price:", cellphone.get\_retail\_price())

main()

# File name: exercise5.py

# Author: Steve Hommy

# Description: Create a CellPhone Class

# Defining class Cellphone

class Cellphone:

    # Defining method init method with private attributes

    def \_\_init\_\_(self):

        self.\_\_manufact = ""

        self.\_\_model = ""

        self.\_\_retail\_price = 0

        self.\_\_id = 0

    # Defining method str method that represents the class objects as a string

    def \_\_str\_\_(self):

        # Here I'm using str.format(). Using { and } to mark where a variable will be substituted and can provide detailed formatting directives

        return """

        Manufacturer: {}

        Model number: {}

        Retail price: {}

        Id number: {}

        """.format(self.\_\_manufact, self.\_\_model, self.\_\_retail\_price, self.\_\_id)

        # Or you can do this using formatted string literals,

        # begin a string with f or F before the opening quotation mark or triple quotation mark.

        # Inside this string, you can write a Python expression between { and } characters that can refer to variables or literal values.

        # return f"""

        # Manufacturer: {self.\_\_manufact}

        # Model number: {self.\_\_model}

        # Retail price: {self.\_\_retail\_price}

        # Id: {self.\_\_id}

        # """

    # Defining Mutator Method

    def set\_manufact(self):

        self.\_\_manufact = str(input("Enter manufacturer: "))

    def set\_model(self):

        self.\_\_model = str(input("Enter model: "))

    def set\_retail\_price(self):

        self.\_\_retail\_price = int(input("Enter retail price: "))

    def set\_id(self):

        self.\_\_id = int(input("Enter id number between 1-6: "))

        if self.\_\_id <= 0 or self.\_\_id >= 7:

            print("Id must be between 1-6")

            self.set\_id()

    # Defining Accessor Method

    def get\_manufact(self):

        return self.\_\_manufact

    def get\_model(self):

        return self.\_\_model

    def get\_retail\_price(self):

        return self.\_\_retail\_price

    def get\_id(self):

        return self.\_\_id

# File name: exercise5\_main.py

# Author: Steve Hommy

# Description: Main function for CellPhone Class

# Importing Cellphone class into main file

from exercise5 import Cellphone

def main():

    cellphone = Cellphone()

    cellphone.set\_manufact()

    cellphone.set\_model()

    cellphone.set\_retail\_price()

    cellphone.set\_id()

    print("Here is the data that you provided:")

    print("Manufacturer:", cellphone.get\_manufact())

    print("Model number:", cellphone.get\_model())

    print("Retail price:", cellphone.get\_retail\_price())

    print("Id number: ", cellphone.get\_id())

    print(cellphone)

main()

# File name: exercise6.py

# Author: Steve Hommy

# Description: Create a CellPhone Class

# Defining class Cellphone

class Cellphone:

    # Defining method init method with private attributes

    def \_\_init\_\_(self):

        self.\_\_manufact = ""

        self.\_\_model = ""

        self.\_\_retail\_price = 0

        self.\_\_id = 0

    # Defining method str method that represents the class objects as a string

    # Defining Accessor Method

    def \_\_str\_\_(self):

        # Using formatted string literals

        return f"""

        Manufacturer: {self.\_\_manufact}

        Model number: {self.\_\_model}

        Retail price: {self.\_\_retail\_price}

        Id number: {self.\_\_id}

        """

    # Defining Mutator Method

    def set\_manufact(self):

        self.\_\_manufact = str(input("Enter manufacturer: "))

    def set\_model(self):

        self.\_\_model = str(input("Enter model: "))

    def set\_retail\_price(self):

        self.\_\_retail\_price = int(input("Enter retail price: "))

    def set\_id(self, id):

        self.\_\_id = id

# File name: exercise6\_main.py

# Author: Steve Hommy

# Description: Main function for CellPhone Class

# Importing Cellphone class into main file

from exercise6 import Cellphone

cellphone\_list = []

def new\_cellphone():

    id = 0

    for i in range(int(input("How many cellphones would you like to create? "))):

        # Creating new objects

        cellphone = input("\nEnter new object name: ")

        cellphone = Cellphone()

        cellphone.set\_manufact()

        cellphone.set\_model()

        cellphone.set\_retail\_price()

        id += 1

        cellphone.set\_id(id)

        cellphone\_list.append(cellphone)

def main():

    new\_cellphone()

    print("Here is the data that you provided:")

    for cellphones in cellphone\_list:

        print(cellphones)

main()

# File name: exercise7\_cellphone.py

# Author: Steve Hommy

# Description: Create a CellPhone Class

# Defining class Cellphone

class Cellphone:

    # Defining method init method with private attributes

    def \_\_init\_\_(self):

        self.\_\_manufact = ""

        self.\_\_model = ""

        self.\_\_retail\_price = 0

        self.\_\_id = 0

    # Defining method str method that represents the class objects as a string

    # Defining Accessor Method

    def \_\_str\_\_(self):

        # Using formatted string literals

        return f"""

        Manufacturer: {self.\_\_manufact}

        Model number: {self.\_\_model}

        Retail price: {self.\_\_retail\_price}

        ID number: {self.\_\_id}

        """

    # Defining Mutator Method

    def set\_manufact(self):

        self.\_\_manufact = str(input("Enter manufacturer: "))

    def set\_model(self):

        self.\_\_model = str(input("Enter model: "))

    def set\_retail\_price(self):

        self.\_\_retail\_price = int(input("Enter retail price: "))

    def set\_id(self, id):

        self.\_\_id = id

    def get\_id(self):

        return self.\_\_id

# File name: exercise7\_dice.py

# Author: Steve Hommy

# Description: Create a Dice Class

import random

# Class definition and initializing attributes.

class Dice:

    def \_\_init\_\_(self):

        self.number = 1

    # Defining roll\_the\_dice. Getting random integer between 1 to 6.

    # Using if, elif and else statment to choose specific value

    def roll\_the\_dice(self):

        random\_number = random.randint(1, 6)

        self.number = random\_number

    # Defining functions and returning their values

    def get\_number(self):

        return self.number

# File name: exercise7\_main.py

# Author: Steve Hommy

# Description: Main function file

# Importing Cellphone and Dice class into main file

from exercise7\_cellphoneClass import Cellphone

from exercise7\_diceClass import Dice

cellphone\_list = []

def new\_cellphone():

    ID = 0

    for i in range(int(input("How many cellphones would you like to create? "))):

        # Creating new objects

        cellphone = input("\nEnter new object name: ")

        cellphone = Cellphone()

        cellphone.set\_manufact()

        cellphone.set\_model()

        cellphone.set\_retail\_price()

        ID += 1

        cellphone.set\_id(ID)

        cellphone\_list.append(cellphone)

def main():

    new\_cellphone()

    dice = Dice()

    dice.roll\_the\_dice()

    print("\nDice number is:", dice.get\_number())

    print("\nHere is the data that you provided:")

    for cellphones in cellphone\_list:

        print(cellphones)

    print("Here is cellphone based on dice roll: ")

    for cellphones in cellphone\_list:

        if dice.number == int(cellphones.get\_id()):

            print(cellphones)

main()

# File name: exercise8.py

# Author: Steve Hommy

# Description: Create a Car Class

# Defining class Car

class Car:

    def \_\_init\_\_(self):

        # Defining method init method with private attributes

        self.\_\_make = ""

        self.\_\_model = ""

        self.\_\_mileage = 0

        self.\_\_price = 0

        self.\_\_color = ""

        self.\_\_maximum\_load\_limit = 0

        self.\_\_size\_of\_trunk = 0

    # Defining method str method that represents the class objects as a string

    def \_\_str\_\_(self):

        return f"""

        Make: {self.\_\_make}

        Model: {self.\_\_model}

        Mileage: {self.\_\_mileage} miles

        Price: {self.\_\_price} €

        Color: {self.\_\_color}

        Maximum load limit {self.\_\_maximum\_load\_limit} Kg

        Size of trunk {self.\_\_size\_of\_trunk} litres

        """

    # Defining Mutator Methods

    def set\_make(self):

        self.\_\_make = input("Enter car maker: ")

    def set\_model(self):

        self.\_\_model = input("Enter car model: ")

    def set\_milage(self):

        self.\_\_mileage = int(input("Enter mileage of the car: "))

    def set\_price(self):

        self.\_\_price = int(input("Enter price of the car: "))

    def set\_color(self):

        self.\_\_color = input("Enter car color: ")

    def set\_maximum\_load\_limit(self):

        self.\_\_maximum\_load\_limit = input("Enter maximum load limit for the car: ")

    def set\_size\_of\_trunk(self):

        self.\_\_size\_of\_trunk = input("Enter size of the trunk for the car: ")

    # All of the get methods.

    def get\_make(self):

        return self.\_\_make

    def get\_model(self):

        return self.\_\_model

    def get\_milage(self):

        return self.\_\_mileage

    def get\_price(self):

        return self.\_\_price

    def get\_color(self):

        return self.\_\_color

    def get\_maximum\_load\_limit(self):

        return self.\_\_maximum\_load\_limit

    def get\_size\_of\_trunk(self):

        return self.\_\_size\_of\_trunk

# File name: exercise8\_main.py

# Author: Steve Hommy

# Description: Main function file

from exercise8 import Car

def main():

    car = Car()

    car.set\_make()

    car.set\_model()

    car.set\_milage()

    car.set\_price()

    car.set\_color()

    car.set\_maximum\_load\_limit()

    car.set\_size\_of\_trunk()

    print("\nThis is your car:")

    print(car)

main()

# File name: exercise9\_carClass.py

# Author: Steve Hommy

# Description: Create a Car Class# File name: exercise8\_carClass.py

# Defining class Car

class Car:

    def \_\_init\_\_(self):

        # Defining method init method with private attributes

        self.\_\_make = ""

        self.\_\_model = ""

        self.\_\_mileage = 0

        self.\_\_price = 0

        self.\_\_color = ""

        self.\_\_maximum\_load\_limit = 0

        self.\_\_size\_of\_trunk = 0

    # Defining method str method that represents the class objects as a string

    def \_\_str\_\_(self):

        return f"""

        Make: {self.\_\_make}

        Model: {self.\_\_model}

        Mileage: {self.\_\_mileage}

        Price: {self.\_\_price}

        Color: {self.\_\_color}

        Maximum load limit: {self.\_\_maximum\_load\_limit}

        Size of trunk: {self.\_\_size\_of\_trunk}

        """

    # Defining Mutator Methods

    def set\_make(self):

        self.\_\_make = input("\nEnter car maker: ")

    def set\_model(self):

        self.\_\_model = input("Enter car model: ")

    def set\_milage(self):

        self.\_\_mileage = int(input("Enter mileage of the car: "))

    def set\_price(self):

        self.\_\_price = int(input("Enter price of the car: "))

    def set\_color(self):

        self.\_\_color = input("Enter car color: ")

    def set\_maximum\_load\_limit(self):

        self.\_\_maximum\_load\_limit = input("Enter maximum load limit for the car: ")

    def set\_size\_of\_trunk(self):

        self.\_\_size\_of\_trunk = input("Enter size of the trunk for the car: ")

    # All of the get methods.

    def get\_make(self):

        return self.\_\_make

    def get\_model(self):

        return self.\_\_model

    def get\_milage(self):

        return self.\_\_mileage

    def get\_price(self):

        return self.\_\_price

    def get\_color(self):

        return self.\_\_color

    def get\_maximum\_load\_limit(self):

        return self.\_\_maximum\_load\_limit

    def get\_size\_of\_trunk(self):

        return self.\_\_size\_of\_trunk

# File name: exercise9\_diceClass.py

# Author: Steve Hommy

# Description: Create a Dice Class

import random

# Class definition and initializing attributes.

class Dice:

    def \_\_init\_\_(self):

        self.number = 1

    # Defining roll\_the\_dice. Getting random integer between 1 to 6.

    # Using if, elif and else statment to choose specific value

    def roll\_the\_dice(self):

        random\_number = random.randint(1, 6)

        self.number = random\_number

    # Defining functions and returning their values

    def get\_number(self):

        return self.number

# File name: exercise9\_mammalClass.py

# Author: Steve Hommy

# Description: Create a Mammal Class

class Mammal:

    def \_\_init\_\_(self):

        self.\_\_id = 0

        self.\_\_species = ""

        self.\_\_name = ""

        self.\_\_size = 0

        self.\_\_weight = 0

    def \_\_str\_\_(self):

        return f"""

        ID: {self.\_\_id}

        Species: {self.\_\_species}

        Name: {self.\_\_name}

        Size: {self.\_\_size}

        Weight: {self.\_\_weight}

        """

    def set\_id(self, id):

        self.\_\_id = id

    def set\_species(self):

        self.\_\_species = str(input("Enter species: "))

    def set\_name(self):

        self.\_\_name = str(input("Enter name: "))

    def set\_size(self):

        self.\_\_size = int(input("Enter size: "))

    def set\_weight(self):

        self.\_\_weight = int(input("Enter weight: "))

    def get\_id(self):

        return self.\_\_id

    def get\_species(self):

        return self.\_\_species

    def get\_name(self):

        return self.\_\_name

    def get\_size(self):

        return self.\_\_size

    def get\_weight(self):

        return self.\_\_weight

# File name: exercise9\_main.py

# Author: Steve Hommy

# Description: Main function file

from exercise9\_carClass import Car

from exercise9\_diceClass import Dice

from exercise9\_mammalClass import Mammal

mammal\_list = []

def new\_mammal():

    ID = 0

    for i in range(int(input("How many mammals would you like to create? "))):

        mammal = input("\nEnter new object name: ")

        mammal = Mammal()

        ID += 1

        mammal.set\_id(ID)

        mammal.set\_species()

        mammal.set\_name()

        mammal.set\_size()

        mammal.set\_weight()

        mammal\_list.append(mammal)

# Check if the mammal fits into the car and if it does then proceedes to check weight limit

def mammal\_into\_trunk(mammals, car):

    if int(mammals.get\_size()) <= int(car.get\_size\_of\_trunk()):

        print(f"""Mammal will fit into the trunk, because:

The trunk size is {car.get\_size\_of\_trunk()} and mammal size is {mammals.get\_size()}\n""")

        mammal\_load\_limit(mammals, car)

    else:

        print(f"""Mammal will not fit into the trunk, because:

The trunk size is {car.get\_size\_of\_trunk()} and mammal size is {mammals.get\_size()}""")

def mammal\_load\_limit(mammals, car):

    if int(mammals.get\_weight()) <= int(car.get\_maximum\_load\_limit()):

        print(f"""Mammal does not exceed the car's load limit, because:

The car's maximum load limit is {car.get\_maximum\_load\_limit()} and mammal weight is {mammals.get\_weight()}""")

    else:

        print(f"""Mammal have exceeded the car's load limit, because:

The car's maximum load limit is {car.get\_maximum\_load\_limit()} and mammal weight is {mammals.get\_weight()}""")

def main():

    new\_mammal()

    car = Car()

    car.set\_make()

    car.set\_model()

    car.set\_milage()

    car.set\_price()

    car.set\_color()

    car.set\_maximum\_load\_limit()

    car.set\_size\_of\_trunk()

    print("\nThis is your car:")

    print(car)

    dice = Dice()

    dice.roll\_the\_dice()

    print("\nDice number is:", dice.get\_number())

    for mammals in mammal\_list:

        if dice.number == int(mammals.get\_id()):

            print("\nHere is mammal based on dice roll: ")

            print(mammals)

            mammal\_into\_trunk(mammals, car)

main()