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	How many cellphones would you like to create? 2 Enter new object name: cellphone1

		Id number: 1 Manufacturer: Apple Model number: Iphone6 Retail price: 500 Id number: 2	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	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

		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	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	This is your car:	Enter car maker: VW Enter car model: Golf VII

	<Run the program so that you get the desired output.>	Make: VW Model: Golf VII Mileage: 100000 miles Price: 8500 € Color: Black Maximum load limit 1500 Kg Size of trunk 350 litres	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	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	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

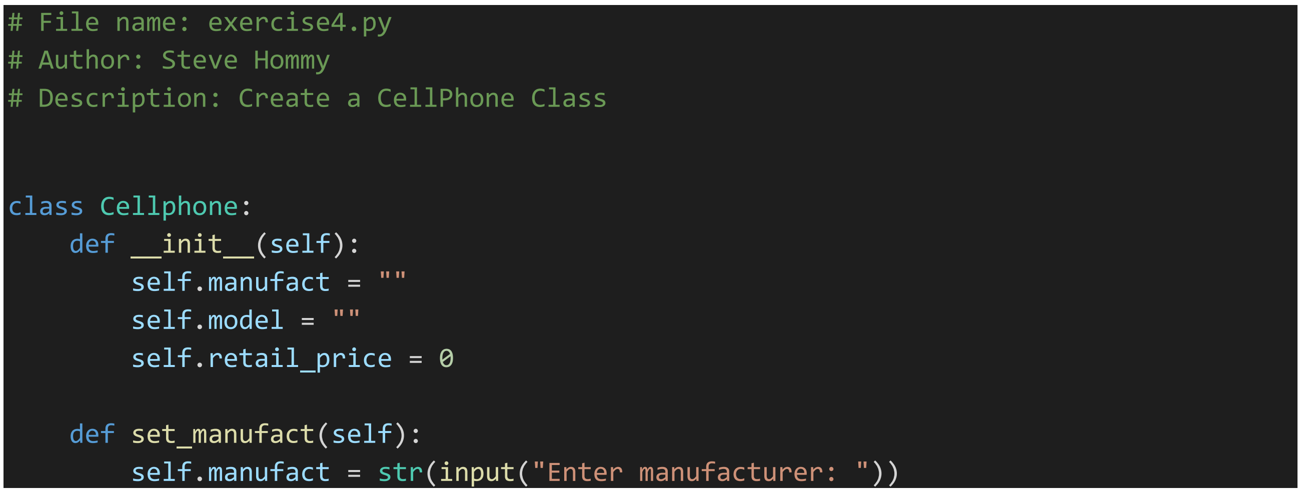
f. You cannot directly call the __str__ method.

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



```python
# 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
```

```python
# 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()
```

```python
# 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
```

```python
# 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()
```

```python
# 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
```

```python
# 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()
```

```python
# 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
```

```python
# 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)
```

```python
main()
```

```python
# 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()
```

```python
    print("\nThis is your car:")
    print(car)


main()
```

```python
# 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.
```

```python
    # 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
```

```python
# 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
```

```python
# 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:
```

```python
        if dice.number == int(mammals.get_id()):
            print("\nHere is mammal based on dice roll: ")
            print(mammals)
            mammal_into_trunk(mammals, car)


main()
```