ASSIGNMENT 6

1) Hello World of Machine Learning

The best small project to start with on a new tool is the classification of iris flowers (e.g. the iris dataset).

- **Attributes are numeric so you have to figure out how to load and handle data.**
- **It is a classification problem, allowing you to practice with perhaps an easier type of supervised learning algorithm.**
- **It is a multi-class classification problem (multi-nominal) that may require some specialized handling.**
- **It only has 4 attributes and 150 rows, meaning it is small and easily fits into memory (and a screen or A4 page).**
- **All of the numeric attributes are in the same units and the same scale, not requiring any special scaling or transforms to get started.**

To do

1. **Installing the Python and SciPy platform.**
2. **Loading the dataset.**
3. **Summarizing the dataset.**

- **Dimensions of the dataset.**
- **Peek at the data itself.**
- **Statistical summary of all attributes.**
- **Breakdown of the data by the class variable.**

4. **Visualizing the dataset.**

- **Univariate plots to better understand each attribute.**
- **Multivariate plots to better understand the relationships between attributes.**

5. **Evaluating some algorithms.**

- **Separate out a validation dataset.**

- **Set-up the test harness to use 10-fold cross validation.**
- **Build multiple different models to predict species from flower measurements**
- **Select the best model.**

test 6 different algorithms:

  - **Logistic Regression (LR)**
  - **Linear Discriminant Analysis (LDA)**
  - **K-Nearest Neighbors (KNN).**
  - **Classification and Regression Trees (CART).**
  - **Gaussian Naive Bayes (NB).**
  - **Support Vector Machines (SVM).**

6. **Making some predictions.**

```
import pandas as pd
data = pd.read_csv(r"C:\Users\Sutirtha Samanta\Desktop\CSVfiles\lab6\Iris.csv")
data.head()
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

In [32]:
```
# Dimensions of the dataset
data.shape
```

Out[32]: (150, 6)

In [33]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

In [34]:
```python
# Statistical summary
data.describe()
```

Out[34]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

In [35]:
```python
# Breakdown by class
data['Species'].value_counts()
```

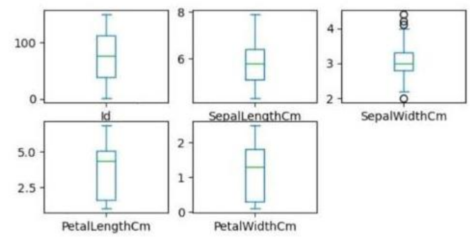
Out[35]:
```
Species
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: count, dtype: int64
```

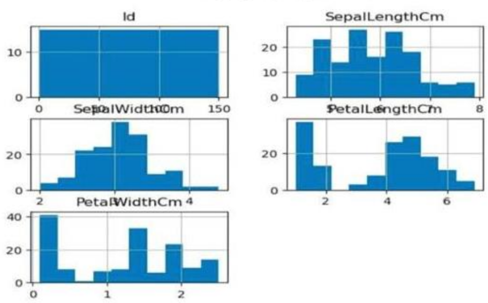
In [45]:
```python
# Univariate Plots

import seaborn as sns
import matplotlib.pyplot as plt

# Boxplot
data.plot(kind='box', subplots=True, layout=(3,3), sharex=False, sharey=False)
plt.show()

# Histogram
data.hist()
plt.show()
```

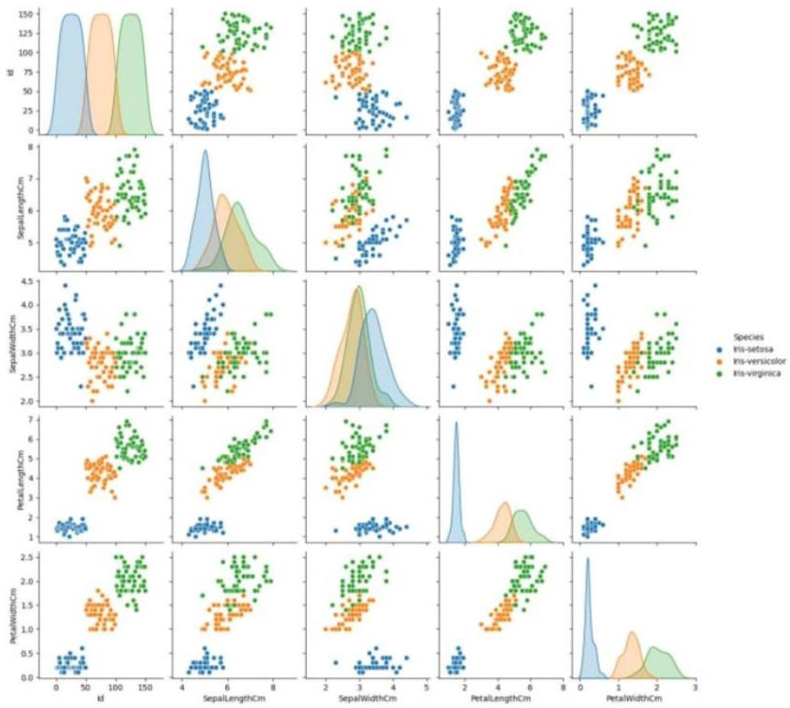




In [37]:
```
# Multivariate Plots

# Pairplot
sns.pairplot(data, hue='Species')
plt.show()
```

```
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWa
rning: use_inf_as_na option is deprecated and will be removed in a future ver
sion. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWa
rning: use_inf_as_na option is deprecated and will be removed in a future ver
sion. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWa
rning: use_inf_as_na option is deprecated and will be removed in a future ver
sion. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWa
rning: use_inf_as_na option is deprecated and will be removed in a future ver
sion. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWa
rning: use_inf_as_na option is deprecated and will be removed in a future ver
sion. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

In [38]:
```
# Evaluating Algorithms
from sklearn.model_selection import train_test_split

# Split dataset into training and validation sets
X = data.drop(columns=['Species'])
y = data['Species']
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_

# Set-Up the Test Harness Using 10-Fold Cross-Validation
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold

kfold = StratifiedKFold(n_splits=10, shuffle=True, random_state=1)


from sklearn.linear_model import LogisticRegression
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC

# List of models to evaluate
models = []
models.append(('LR', LogisticRegression()))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC()))

# Evaluate each model
results = []
names = []
for name, model in models:
    cv_results = cross_val_score(model, X_train, y_train, cv=kfold, scoring='a
    results.append(cv_results)
    names.append(name)
    print(f"{name}: {cv_results.mean():.3f} ({cv_results.std():.3f})")
```

```
LR: 1.000 (0.000)
LDA: 0.992 (0.025)
KNN: 1.000 (0.000)
CART: 0.992 (0.025)
NB: 0.992 (0.025)
SVM: 0.975 (0.053)
```



```
t-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regres
sion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
ession)
  n_iter_i = _check_optimize_result(
```

In [39]:
```
# Predictions

# Train the best model
model = LinearDiscriminantAnalysis()
model.fit(X_train, y_train)

# Make predictions on the validation set
predictions = model.predict(X_val)

# Evaluate predictions
from sklearn.metrics import accuracy_score, classification_report
print(accuracy_score(y_val, predictions))
print(classification_report(y_val, predictions))
```

```
1.0
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        11
Iris-versicolor       1.00      1.00      1.00        13
 Iris-virginica       1.00      1.00      1.00         6

       accuracy                           1.00        30
      macro avg       1.00      1.00      1.00        30
   weighted avg       1.00      1.00      1.00        30
```

In [ ]:

2) Write a Python program to show method overriding.

```
class Animal:
  def sound(self):
    return "Some generic animal sound"
class Dog(Animal):
  def sound(self):
    return "Bark"
class Cat(Animal):
  def sound(self):
    return "Meow"
generic_animal = Animal()
dog = Dog()
cat = Cat()
print("Animal Sound:", generic_animal.sound())
print("Dog Sound:", dog.sound())
print("Cat Sound:", cat.sound())
```

OUTPUT-

```
= RESTART: C:/Users/Sutirtha/AppData/Local/Programs/Python/Python312/PYTHON 6.py
Animal Sound: Some generic animal sound
Dog Sound: Bark
Cat Sound: Meow
```

3) Write a Python program to show method hiding.

```
class Parent:
  def show(self):
    return "This is the Parent class method."
class Child(Parent):
  def show(self):
    return "This is the Child class method."
class GrandChild(Child):
```

```
    def show(self):
        return "This is the GrandChild class method."
parent = Parent()
child = Child()
grandchild = GrandChild()
print("Parent class output:", parent.show())
print("Child class output:", child.show())
print("GrandChild class output:", grandchild.show())
print("Parent class method accessed from Child:", super(Child, child).show())
print("Child class method accessed from GrandChild:", super(GrandChild, grandchild).show())
```

OUTPUT-

```
= RESTART: C:/Users/Sutirtha/AppData/Local/Programs/Python/Python312/PYTHON 6.py
Parent class output: This is the Parent class method.
Child class output: This is the Child class method.
GrandChild class output: This is the GrandChild class method.
Parent class method accessed from Child: This is the Parent class method.
Child class method accessed from GrandChild: This is the Child class method.
```

4) Create a general class ThreeDObject and derive the classes Box, Cube, Cylinder and Cone from it. The class ThreeDObject has methods wholeSurfaceArea ( ) and volume ( ). Override these two methods in each of the derived classes to calculate the volume and whole surface area of each type of three-dimensional objects. The dimensions of the objects are to be taken from the users and passed through the respective constructors of each derived class. Write a main method to test these classes.

```
import math
class ThreeDObject:
    def wholeSurfaceArea(self):
        pass
    def volume(self):
        pass
class Box(ThreeDObject):
    def __init__(self, length, width, height):
        self.length = length
        self.width = width
        self.height = height
```

```
    def wholeSurfaceArea(self):
        return 2 * (self.length * self.width + self.width * self.height + self.height * self.length)
    def volume(self):
        return self.length * self.width * self.height
class Cube(ThreeDObject):
    def___init__(self, side):
        self.side = side
    def wholeSurfaceArea(self):
        return 6 * self.side ** 2
    def volume(self):
        return self.side ** 3
class Cylinder(ThreeDObject):
    def___init__(self, radius, height):
        self.radius = radius
        self.height = height
    def wholeSurfaceArea(self):
        return 2 * math.pi * self.radius * (self.radius + self.height)
    def volume(self):
        return math.pi * self.radius ** 2 * self.height
class Cone(ThreeDObject):
    def___init__(self, radius, height):
        self.radius = radius
        self.height = height
    def wholeSurfaceArea(self):
        slant_height = math.sqrt(self.radius ** 2 + self.height ** 2)
        return math.pi * self.radius * (self.radius + slant_height)
    def volume(self):
        return (1/3) * math.pi * self.radius ** 2 * self.height
def main():
    print("Testing the 3D Objects")
    length = float(input("\nEnter the length of the Box: "))
```

```
    width = float(input("Enter the width of the Box: "))
    height = float(input("Enter the height of the Box: "))
    box = Box(length, width, height)
    print(f"Box Surface Area: {box.wholeSurfaceArea()}")
    print(f"Box Volume: {box.volume()}")
    side = float(input("\nEnter the side length of the Cube: "))
    cube = Cube(side)
    print(f"Cube Surface Area: {cube.wholeSurfaceArea()}")
    print(f"Cube Volume: {cube.volume()}")
    radius = float(input("\nEnter the radius of the Cylinder: "))
    height = float(input("Enter the height of the Cylinder: "))
    cylinder = Cylinder(radius, height)
    print(f"Cylinder Surface Area: {cylinder.wholeSurfaceArea()}")
    print(f"Cylinder Volume: {cylinder.volume()}")
    radius = float(input("\nEnter the radius of the Cone: "))
    height = float(input("Enter the height of the Cone: "))
    cone = Cone(radius, height)
    print(f"Cone Surface Area: {cone.wholeSurfaceArea()}")
    print(f"Cone Volume: {cone.volume()}")
if __name___== "_main_":
    main()
```

OUTPUT-

```
= RESTART: C:/Users/Sutirtha/AppData/Local/Programs/Python/Python312/PYTHON 6.py
Testing the 3D Objects

Enter the length of the Box: 5
Enter the width of the Box: 4
Enter the height of the Box: 3
Box Surface Area: 94.0
Box Volume: 60.0

Enter the side length of the Cube: 3
Cube Surface Area: 54.0
Cube Volume: 27.0

Enter the radius of the Cylinder: 3
Enter the height of the Cylinder: 5
Cylinder Surface Area: 150.79644737231007
Cylinder Volume: 27.0

Enter the radius of the Cone: 3
Enter the height of the Cone: 4
Cone Surface Area: 75.39822368615503
Cone Volume: 37.69911184307752
```

5) Write a program to create a class named Vehicle having protected instance variables regnNumber, speed, color, ownerName and a method showData ( ) to show “This is a vehicle class”. Inherit the Vehicle class into subclasses named Bus and Car having individual private instance variables routeNumber in Bus and manufacturerName in Car and both of them having showData ( ) method showing all details of Bus and Car respectively with content of the super class’s showData ( ) method.

```
class Vehicle:
    def___init_(self, regnNumber, speed, color, ownerName):
        self._regnNumber = regnNumber
        self._speed = speed
        self._color = color
        self._ownerName = ownerName
    def showData(self):
        print("This is a Vehicle class")
        print(f"Registration Number: {self._regnNumber}")
        print(f"Speed: {self._speed} km/h")
        print(f"Color: {self._color}")
        print(f"Owner Name: {self._ownerName}")
class Bus(Vehicle):
    def___init_(self, regnNumber, speed, color, ownerName, routeNumber):
        super()._init_(regnNumber, speed, color, ownerName)
        self._routeNumber = routeNumber
    def showData(self):
        super().showData()  # Call the showData() method of the Vehicle class
        print(f"Route Number: {self._routeNumber}")
class Car(Vehicle):
    def___init_(self, regnNumber, speed, color, ownerName, manufacturerName):
        super()._init_(regnNumber, speed, color, ownerName)
        self._manufacturerName = manufacturerName
    def showData(self):
        super().showData()  # Call the showData() method of the Vehicle class
        print(f"Manufacturer Name: {self._manufacturerName}")
def main():
    bus = Bus("KA-01-AB-1234", 60, "Yellow", "Mr. Naresh", 15)
```

```
    print("Bus Details:")

    bus.showData()

    print("\n---------------------------------\n")

    car = Car("MH-02-CD-5678", 120, "Red", "Ms. Anjali", "Toyota")

    print("Car Details:")

    car.showData()

if __name___== "_main_":

    main()
```

OUTPUT-

```
= RESTART: C:/Users/Sutirtha/AppData/Local/Programs/Python/Python312/PYTHON 6.py
Testing the 3D Objects

Bus Details:
This is a Vehicle class
Registration Number: KA-01-AB-1234
Speed: 60 km/h
Color: Yellow
Owner Name: Mr. Naresh
Route Number: 15

---------------------------------------

Car Details:
This is a Vehicle class
Registration Number: MH-02-CD-5678
Speed: 120 km/h
Color: Red
Owner Name: Ms. Anjali
Manufacturer Mane: Toyota
```

6) An educational institution maintains a database of its employees. The database is divided into a number of classes whose hierarchical relationships are shown below. Write all the classes and define the methods to create the database and retrieve individual information as and when needed. Write a driver program to test the classes. Staff (code, name) Officer (grade) is a Staff RegularTypist (remuneration) is a Typist Teacher (subject, publication) is a Staff Typist (speed) is a Staff CasualTypist (daily wages) is a Typist.

```
class Staff:

    def__init_(self, code, name):

        self.code = code

        self.name = name

    def showData(self):

        print(f"Staff Code: {self.code}")

        print(f"Staff Name: {self.name}")

class Officer(Staff):

    def__init_(self, code, name, grade):

        super()._init_(code, name)

        self.grade = grade
```

```
    def showData(self):
        super().showData()
        print(f"Grade: {self.grade}")
class Typist(Staff):
    def___init_(self, code, name, speed):
        super()._init_(code, name)
        self.speed = speed
    def showData(self):
        super().showData()
        print(f"Typing Speed: {self.speed} wpm")
class RegularTypist(Typist):
    def___init_(self, code, name, speed, remuneration):
        super()._init_(code, name, speed)
        self.remuneration = remuneration
    def showData(self):
        super().showData()
        print(f"Remuneration: {self.remuneration}"
class CasualTypist(Typist):
    def___init_(self, code, name, speed, daily_wages):
        super()._init_(code, name, speed)
        self.daily_wages = daily_wages
    def showData(self):
        super().showData()
        print(f"Daily Wages: {self.daily_wages}")
class Teacher(Staff):
    def___init_(self, code, name, subject, publication):
        super()._init_(code, name)
        self.subject = subject
        self.publication = publication
    def showData(self):
        super().showData()
        print(f"Subject: {self.subject}")
        print(f"Publication: {self.publication}")
```

```
def main():
    print("Officer Details:")
    officer = Officer("O1001", "Alice", "Grade A")
    officer.showData()
    print("\n--------------------------------------\n")
    print("Regular Typist Details:")
    reg_typist = RegularTypist("T2001", "Bob", 80, 30000)
    reg_typist.showData()
    print("\n--------------------------------------\n")
    print("Casual Typist Details:")
    casual_typist = CasualTypist("T2002", "Charlie", 70, 500)
    casual_typist.showData()
    print("\n--------------------------------------\n")
    print("Teacher Details:")
    teacher = Teacher("T3001", "David", "Mathematics", "Maths Today")
    teacher.showData()
if __name__ == "_main_":
    main()
```

OUTPUT-

```
= RESTART: C:/Users/Sutirtha/AppData/Local/Programs/Python/Python312/PYTHON 6.py
Officer Details:
Staff Code: O1001
Staff Name: Alice
Grade: Grade A

--------------------------------------

Regular Typist Details:
Staff Code: T2001
Staff Name: Bob
Typing Speed: 80 wpm
Remuneration: 30000

--------------------------------------

Casual Typist Details:
Staff Code: T2002
Staff Name: Charlie
Typing Speed: 70 wpm
Daily Wages: 500

--------------------------------------

Teacher Details:
Staff Code:T3001
Staff Name: David
Subject: Mathematics
Publication: Maths Today
```

7) Create a base class Building that stores the number of floors of a building, number of rooms and it's total footage. Create a derived class House that inherits Building and also stores the number of bedrooms and bathrooms. Demonstrate the working of the classes.

```
class Building:
   def___init_(self, floors, rooms, total_footage):
      self.floors = floors
      self.rooms = rooms
      self.total_footage = total_footage
   def showData(self):
      print(f"Number of Floors: {self.floors}")
      print(f"Number of Rooms: {self.rooms}")
      print(f"Total Footage: {self.total_footage} sqft")
class House(Building):
   def__init_(self, floors, rooms, total_footage, bedrooms, bathrooms):
      super()._init_(floors, rooms, total_footage)
      self.bedrooms = bedrooms
      self.bathrooms = bathrooms
   def showData(self):
      super().showData()  # Call the showData method of the Building class
      print(f"Number of Bedrooms: {self.bedrooms}")
      print(f"Number of Bathrooms: {self.bathrooms}")
def main():
   # Creating and testing a Building
   print("Building Details:")
   building = Building(5, 20, 10000)
   building.showData()
   print("\n..................................\n")
   print("House Details:")
   house = House(2, 8, 2500, 3, 2)
   house.showData()
if __name___== "_main_":
   main()
```

OUTPUT-

```
= RESTART: C:/Users/Sutirtha/AppData/Local/Programs/Python/Python312/PYTHON 6.py
Building Details:
Number of Floors: 5
Number of Rooms: 20
Total Footage: 10000 sqft

------------------------------

House Details:
Number of Floors: 2
Number of Rooms: 8
Total Footage: 2500 sqft
Number of Bedrooms: 3
Number of Bathrooms: 2
```

8) In the earlier program, create a second derived class Office that inherits Building and stores the number of telephones and tables. Now demonstrate the working of all three classes.

```
class Building:
   def__init_(self, floors, rooms, total_footage):
      self.floors = floors
      self.rooms = rooms
      self.total_footage = total_footage
   def showData(self):
      print(f"Number of Floors: {self.floors}")
      print(f"Number of Rooms: {self.rooms}")
      print(f"Total Footage: {self.total_footage} sqft")
class House(Building):
   def__init_(self, floors, rooms, total_footage, bedrooms, bathrooms):
      super()._init_(floors, rooms, total_footage)
      self.bedrooms = bedrooms
      self.bathrooms = bathrooms
   def showData(self):
      super().showData()  # Call the showData method of the Building class
      print(f"Number of Bedrooms: {self.bedrooms}")
      print(f"Number of Bathrooms: {self.bathrooms}")
class Office(Building):
   def__init_(self, floors, rooms, total_footage, telephones, tables):
      super()._init_(floors, rooms, total_footage)
      self.telephones = telephones
      self.tables = tables
   def showData(self):
      super().showData()  # Call the showData method of the Building class
```

```
        print(f"Number of Telephones: {self.telephones}")
        print(f"Number of Tables: {self.tables}")
def main():
    print("Building Details:")
    building = Building(3, 15, 8000)
    building.showData()
    print("\n-------------------------------\n")
    print("House Details:")
    house = House(2, 8, 2500, 3, 2)
    house.showData()
    print("\n-------------------------------\n")
    print("Office Details:")
    office = Office(4, 10, 5000, 20, 25)
    office.showData()
if __name___== "_main_":
    main()
```

OUTPUT-

```
= RESTART: C:/Users/Sutirtha/AppData/Local/Programs/Python/Python312/PYTHON 6.py
Building Details:
Number of Floors: 3
Number of Rooms: 15
Total Footage: 8000 sqft

-------------------------------

House Details:
Number of Floors: 2
Number of Rooms: 8
Total Footage: 2500 sqft
Number of Bedrooms: 3
Number of Bathrooms: 2

-------------------------------

Office Details:
Number of Floors: 4
Number of Rooms: 10
Total Footage: 5000 sqft
Number of Telephones: 20
Number of Tables: 25
```

9) Write a Python program which creates a base class Num and contains an integer number along with a method shownum() which displays the number. Now create a derived class HexNum which inherits Num and overrides shownum() which displays the hexadecimal value of the number. Demonstrate the working of the classes.

```
class Num:
    def__init_(self, number):
        self.number = number
```

```
    def shownum(self):
        print(f"The number is: {self.number}")
class HexNum(Num):
    def shownum(self):
        print(f"The hexadecimal value of the number is: {hex(self.number)}")
def main():
    num = Num(255)
    print("Num class output:")
    num.shownum()
    print("\n.................................\n")
    hex_num = HexNum(255)
    print("HexNum class output:")
    hex_num.shownum()
if __name___== "_main_":
    main()
```

OUTPUT-

```
= RESTART: C:/Users/Sutirtha/AppData/Local/Programs/Python/Python312/PYTHON 6.py
Num class output:
The number is: 255


--------------------------

HexNum class output:
The hexadecimal value of the number is: 0xff
```

10) Write a Python program which creates a base class Num and contains an integer number along with a method shownum() which displays the number. Now create a derived class OctNum which inherits Num and overrides shownum() which displays the octal value of the number. Demonstrate the working of the classes.

```
class Num:
    def__init_(self, number):
        self.number = number
    def shownum(self):
        print(f"The number is: {self.number}")
class OctNum(Num):
    def shownum(self):
        print(f"The octal value of the number is: {oct(self.number)}
def main():
```

```
    num = Num(255)
    print("Num class output:")
    num.shownum()
    print("\n-------------------------\n")
    oct_num = OctNum(255)
    print("OctNum class output:")
    oct_num.shownum()
if __name__ == "__main__":
    main()
```

OUTPUT-

```
= RESTART: C:/Users/Sutirtha/AppData/Local/Programs/Python/Python312/PYTHON 6.py
Num class output:
The number is: 255


-------------------------

OctNum class output:
The octal value of the number is: 0o377
```

11) Combine Question number 10 and 11 and have all the three classes together. Now describe the working of all classes.

```
class Num:
    def __init__(self, number):
        self.number = number
    def shownum(self):
        print(f"The number is: {self.number}")
class HexNum(Num):
    def shownum(self):
        print(f"The hexadecimal value of the number is: {hex(self.number)}")
class OctNum(Num):
    def shownum(self):
        print(f"The octal value of the number is: {oct(self.number)}")
def main():
    num = Num(255)
    print("Num class output:")
    num.shownum()
    print("\n-------------------------\n")
```

```
    hex_num = HexNum(255)
    print("HexNum class output:")
    hex_num.shownum()
    print("\n_______________________________\n")
    oct_num = OctNum(255)
    print("OctNum class output:")
    oct_num.shownum()
if __name___== "_main_":
    main()
```

OUTPUT-

```
= RESTART: C:/Users/Sutirtha/AppData/Local/Programs/Python/Python312/PYTHON 6.py
Num class output:
The number is: 255


--------------------------

HexNum class output:
The hexadecimal value of the number is: 0xff

--------------------------

OctNum class output:
The octal value of the number is: 0o377
```