Python Lab session - 5

(Q-1) Explain Python Ecosystem for Data Science.

```
===>> For the Data Science ,We know given librery is used for his functionality
or easy to used.
    That's why going futher one by one lib.

    1) Numpy  --> It is stand for Numric Python.
                  NumPy is a python library used for working with arrays.
                  It also has functions for working in domain of linear
                algebra, fourier transform, and matrices.
```

In [1]:
```
import numpy as np
a = np.array([1, 2, 3], dtype = complex)
print(a)
```

```
[1.+0.j 2.+0.j 3.+0.j]
```

```
      2) Pandas -->Pandas is an open-source library that is built on top of
NumPy library. It is a Python package that offers various data structures and
operations for manipulating numerical data and time series. It is mainly
popular for importing and analyzing data much easier. Pandas is fast and it has
high-performance & productivity for users.
```

In [3]:
```
import pandas as pd
import numpy as np
data = np.array(['a','b','c','d'])
s = pd.Series(data)
print(s)
```

```
0    a
1    b
2    c
3    d
dtype: object
```

In [4]:
```
data = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/iri
data.head()
```

Out[4]:

	sepal_length	sepal_width	petal_length	petal_width	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

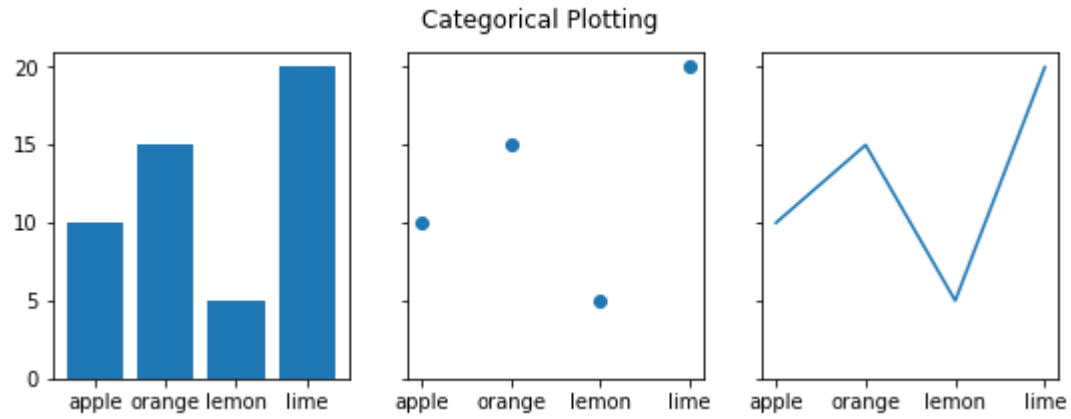
3)Matplotlib -->Matplotlib is one of the most popular Python packages used for data visualization. It is a cross-platform library for making 2D plots from data in arrays. It provides an object-oriented API that helps in embedding plots in applications using Python GUI toolkits such as PyQt, WxPythonotTkinter. It can be used in Python and IPython shells, Jupyter notebook and web application servers also.

In [5]:
```
import matplotlib.pyplot as plt

data = {'apple': 10, 'orange': 15, 'lemon': 5, 'lime': 20}
names = list(data.keys())
values = list(data.values())

fig, axs = plt.subplots(1, 3, figsize=(9, 3), sharey=True)
axs[0].bar(names, values)
axs[1].scatter(names, values)
axs[2].plot(names, values)
fig.suptitle('Categorical Plotting')
```

Out[5]: Text(0.5, 0.98, 'Categorical Plotting')

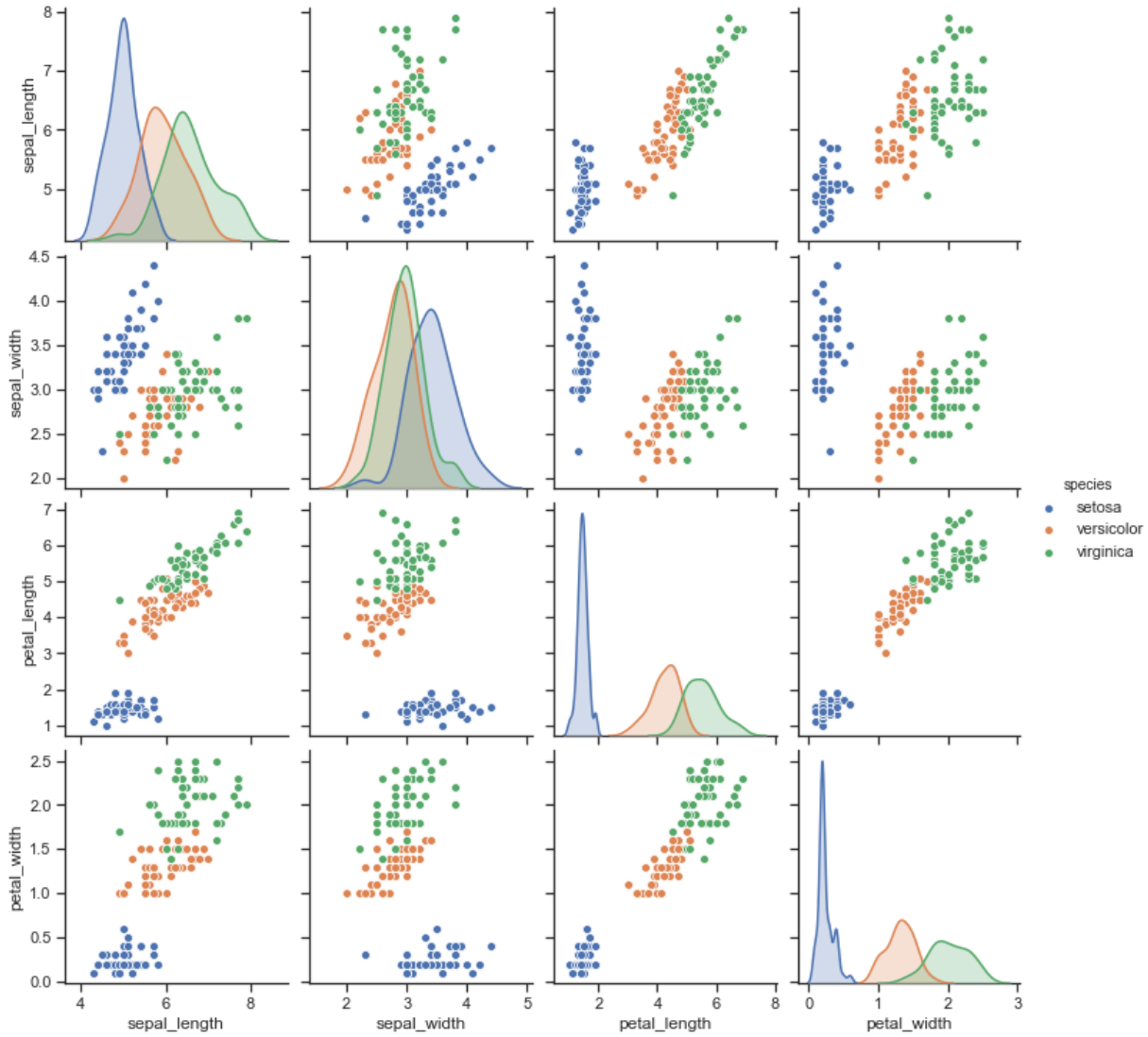


4) Seaborn --> Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics. For a brief introduction to the ideas behind the library, you can read the introductory notes. Visit the installation page to see how you can download the package.

In [6]:
```
import seaborn as sns
sns.set(style="ticks")

df = sns.load_dataset("iris")
sns.pairplot(df, hue="species")
```

Out[6]: <seaborn.axisgrid.PairGrid at 0x29a29e35198>



5) Sckit-Learn --> Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python.

In [8]:
```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics

iris = load_iris()
X = iris.data
y = iris.target
feature_names = iris.feature_names
target_names = iris.target_names
print("Feature names:", feature_names)
print("Target names:", target_names)
print("\nFirst 10 rows of X:\n", X[:10])  # 30% for Testing and 70% for training

X_train, X_test, y_train, y_test = train_test_split(
   X, y, test_size = 0.3, random_state = 1
)

print(X_train.shape)
print(X_test.shape)

print(y_train.shape)
print(y_test.shape)
```

```
Feature names: ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)',
'petal width (cm)']
Target names: ['setosa' 'versicolor' 'virginica']

First 10 rows of X:
 [[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]
 [5.4 3.9 1.7 0.4]
 [4.6 3.4 1.4 0.3]
 [5.  3.4 1.5 0.2]
 [4.4 2.9 1.4 0.2]
 [4.9 3.1 1.5 0.1]]
(105, 4)
(45, 4)
(105,)
(45,)
```

In [12]:
```
classifier_knn = KNeighborsClassifier(n_neighbors = 3)
classifier_knn.fit(X_train, y_train)
y_pred = classifier_knn.predict(X_test)
# Finding accuracy by comparing actual response values(y_test)with predicted resp
print("Accuracy:", metrics.accuracy_score(y_test, y_pred)*100)
# Providing sample data and the model will make prediction out of that data
```

```
Accuracy: 97.77777777777777
```

```
    6)Keras  --> Keras is an open-source neural-network library written in
Python. It is capable of running on top of TensorFlow, Microsoft Cognitive
Toolkit, R, Theano, or PlaidML. Designed to enable fast experimentation with
deep neural networks, it focuses on being user-friendly, modular, and
extensible.
```

```
   7)Tensorflow  -->TensorFlow is an open source machine learning framework for
all developers. It is used for implementing machine learning and deep learning
applications. To develop and research on fascinating ideas on artificial
intelligence, Google team created TensorFlow. TensorFlow is designed in Python
programming language, hence it is considered an easy to understand framework.
```

In [ ]:

(Q-2) Expain program in jupyter and perform it

1)Count the vowel in given string

In [13]:
```
class Vowelchar():
     def vowel(self):
         vowels = ('a', 'A', 'e', 'E', 'i', 'I', 'o', 'O', 'u', 'U')
         string = input("Enter the string:")
         output = {}
         for char in vowels:
            x = string.count(char)
            output[char] = x
         print(output)

Vowelchar().vowel()
```

```
Enter the string:This is Dummy String
{'a': 0, 'A': 0, 'e': 0, 'E': 0, 'i': 3, 'I': 0, 'o': 0, 'O': 0, 'u': 1, 'U':
0}
```

2) Password Validation ...

In [14]:
```
def check(passwd):
    val = True

    if len(passwd) < 9:
        print('length should be at least 9')
        val = False

    if not any(char.isdigit() for char in passwd):
        print('Password should have at least one numeral')
        val = False

    if not any(char.isupper() for char in passwd):
        print('Password should have at least one uppercase letter')
        val = False

    if not any(char.islower() for char in passwd):
        print('Password should have at least one lowercase letter')
        val = False

    Specialchar =['', '%', '#', '$']
    if not any(char in Specialchar for char in passwd):
        print('Password should have at least one of the symbols _ Or % Or * Or $'
        val = False
    if val:
        return val

print("Enter password ...")
passwd = input()
if (check(passwd)):
        print("Password Accepted")
else:
        print("Invalid Password !!")
```

```
Enter password ...
M!croSoft$987
Password Accepted
```

3) Caesar cipher encryption:

In [15]:
```
def Encryption(s,k):
    encstr=""
    for i in s:
        if(ord(i))>=65 and (ord(i)<=90):
            temp=(ord(i)+k)
            if temp>90:
                temp=temp%90+64
            encstr=encstr+chr(temp)
        elif(ord(i))>=97 and (ord(i)<=122):
            temp=(ord(i)+k)
            if temp>122:
                temp=temp%122+96
            encstr=encstr+chr(temp)
        else:
            encstr=encstr+chr(ord(i)+k)
    return encstr
def Decryption(k):
    p=Encryption(s,k)
    decstr=""
    for i in p:
        if((ord(i))>=65) and (ord(i))<=90:
            decstr=decstr+chr((ord(i) - k-65) % 26 + 65)
        elif((ord(i))>=97) and (ord(i))<=122:
            decstr=decstr+chr((ord(i) - k - 97) % 26 + 97)
        else:
            decstr=decstr+chr(ord(i)-k)
    return decstr
print("Enter the string to Encrypt and decrypt : ")
s=input()
k=9
k=k%26
print("Encrypted String : ",Encryption(s,k))
print("Decrypted String : ",Decryption(k))
```

```
Enter the string to Encrypt and decrypt :
This is DUMMY String
Encrypted String :  Cqrb)rb)MDVVH)Bcarwp
Decrypted String :  This is DUMMY String
```

4) Numpy using Example

In [1]:
```
import numpy as np


a = np.arange(20)
print(a)
print(a[8:12])
print(a[1:100:3])


cosvalue = np.cos(4)
print("Cos value" ,cosvalue)
sinvalue = np.sin(4)
print("Cos value" ,sinvalue)


print(np.random.rand())
print(np.random.ranf())


print("\n1-D array")
a = np.array([1,2,3,4,5,6])
print(a)

print("\n1-D array with arange function to interval ")
a1 = np.arange(2,12,3)
print(a1)

print("\n1-D array with linspace ")
a2 = np.linspace(1,10,4)
print(a2)


print("\n2-D array")
b = np.array([[1,2,3],[4,5,6]])
print(b)

print("\n2-D array with ones element ")
b1 = np.ones((2,2))
print(b1)

print("\n2-D array with random array ")
b2 = np.random.rand(2,2)
print(b2)

print("\n2-D array with identity")
b3 = np.identity(2)
print(b3)


print("\n3-D array")
c = np.array([[[1,2,3],
               [4,5,6]],
              [[7,8,9],
               [10,11,12]]])
print(c)
```

```
print("\n3-D array with zeros ")
c1 = np.zeros((2,3,4))
print(c1)


print("B\n", np.linspace(1.0, 3.0, num=5, retstep=True), "\n")



arr = np.array([[14, 70], [42, 60]],
                 dtype = np.float64)

print(c.dtype)

print("\nAddition of Array elements: ")
print(np.sum(arr))

print("\nSquare root of Array1 elements: ")
print(np.sqrt(arr))

print("\nTranspose of Array: ")
print(arr.T)


# Printing type of arr object
print("Array is of type: ", type(arr))

# Printing array dimensions (axes)
print("No. of dimensions: ", arr.ndim)

# Printing shape of array
print("Shape of array: ", arr.shape)

# Printing size (total number of elements) of array
print("Size of array: ", arr.size)

# Printing type of elements in array
print("Array stores elements of type: ", arr.dtype)  # complex
```

```
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19]
[ 8  9 10 11]
[ 1  4  7 10 13 16 19]
Cos value -0.6536436208636119
Cos value -0.7568024953079282
0.37463574140319633
0.8037711756216795

1-D array
[1 2 3 4 5 6]

1-D array with arange function to interval
[ 2  5  8 11]
```

```
1-D array with linspace
[ 1.  4.  7. 10.]

2-D array
[[1 2 3]
 [4 5 6]]

2-D array with ones element
[[1. 1.]
 [1. 1.]]

2-D array with random array
[[0.6169639  0.50275067]
 [0.23072728 0.49661341]]

2-D array with identity
[[1. 0.]
 [0. 1.]]

3-D array
[[[ 1  2  3]
  [ 4  5  6]]

 [[ 7  8  9]
  [10 11 12]]]

3-D array with zeros
[[[0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]]

 [[0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]]]
B
 (array([1. , 1.5, 2. , 2.5, 3. ]), 0.5)

int32

Addition of Array elements:
186.0

Square root of Array1 elements:
[[3.74165739 8.36660027]
 [6.4807407  7.74596669]]

Transpose of Array:
[[14. 42.]
 [70. 60.]]
Array is of type:  <class 'numpy.ndarray'>
No. of dimensions:  2
Shape of array:  (2, 2)
Size of array:  4
Array stores elements of type:  float64
```

In [2]:
```
import numpy as np

print("Copy array operation ")
a = np.array([1,2,3])
copy1 = a.copy()
a[2] = 10
print(a)
print(copy1)

print("View array operation ...It's one type of cloning")
view1 = a.view()
a[2] = 10
print(a)
print(view1)


print("Shape array operation ")
arr1 = np.array([[1, 2, 3, 4], [5, 6, 7, 8]])
print("Shape of array :",arr1.shape)


print("Reshape array operation ")
arr = np.array([1,2,3,4,5,6,7,8,9,10,11,12])
reshape_array = arr.reshape(3,4)
reshape_array1 = arr.reshape(3,2,2)
print("2-D array..\n",reshape_array)
print("3-D array ..\n",reshape_array1)


print("Flatten array operation..")
print("Given array\n",reshape_array1)
print("Flatten array\n ", reshape_array1.flatten())

print("Flatten array operation.. example ")
array = np.arange(15).reshape(3, 5)
print("Given array\n",array)
print("Flatten array \n", array.flatten())
```

```
Copy array operation
[ 1  2 10]
[1 2 3]
View array operation ...It's one type of cloning
[ 1  2 10]
[ 1  2 10]
Shape array operation
Shape of array : (2, 4)
Reshape array operation
2-D array..
 [[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
3-D array ..
 [[[ 1  2]
  [ 3  4]]
```

```
 [[ 5  6]
  [ 7  8]]

 [[ 9 10]
  [11 12]]]
Flatten array operation..
Given array
 [[[ 1  2]
  [ 3  4]]

 [[ 5  6]
  [ 7  8]]

 [[ 9 10]
  [11 12]]]
Flatten array
  [ 1  2  3  4  5  6  7  8  9 10 11 12]
Flatten array operation.. example
Given array
 [[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]]
Flatten array
 [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14]
```

In [3]:
```
import numpy as np
print("Matrix operation ..\n")
matrix = np.array( [ [ 4, 5, 6 ],
                     [ 7, 8, 9 ],
                     [ 10, 11, 12 ] ] )
matrix1 = np.array( [ [ 1, 2, 3 ],
                     [ 4, 5, 6 ],
                     [ 7, 8, 9 ] ] )

mat = np.random.rand(4)

print(matrix)
print(matrix1)
print("\nTranspot of matrix\n",matrix.transpose())
print("Array of random ",mat)
mat.sort()
print("\n sort all element from matrix\n",mat)
print("\nAddition of two matrix \n",matrix+matrix1)
print("\nMultiplication of two matrix \n",matrix*matrix1)
print("\nMinimum element from matrix \n",matrix.max())
print("\nMultiplication of from matrix \n",matrix.min())
print("\nSum of the matrix \n",matrix.sum())
print("\nAverage of matrix \n",matrix.mean())
print("\nProduct of matrix \n",matrix.prod())
```

```
Matrix operation ..

[[ 4  5  6]
 [ 7  8  9]
 [10 11 12]]
[[1 2 3]
 [4 5 6]
 [7 8 9]]

Transpot of matrix
 [[ 4  7 10]
 [ 5  8 11]
 [ 6  9 12]]
Array of random  [0.56406907 0.84831072 0.35068213 0.2041597 ]

 sort all element from matrix
 [0.2041597  0.35068213 0.56406907 0.84831072]

Addition of two matrix
 [[ 5  7  9]
 [11 13 15]
 [17 19 21]]

Multiplication of two matrix
 [[  4  10  18]
 [ 28  40  54]
 [ 70  88 108]]
```

```
Minimum element from matrix
 12

Multiplication of from matrix
 4

Sum of the matrix
 72

Average of matrix
 8.0

Product of matrix
 79833600
```

In [ ]: