In [1]:
```
#importing numpy package
import numpy as np
b=np.array([[[1,2,3,5],[2,3,4,4]],[[1,2,3,5],[2,3,4,4]]])
#printing the data type
print(b.dtype)
#printing the dimension of the NumPy array
print(b.ndim)
#printing the shape the NumPy array
print(b.shape)
# printing the size the NumPy array i.e. total number of elements
print(b.size)
#to generate an array of numerical numbers from 10 to 100 with 2 steps
c=np.arange(10,100,2)
print(c)
#to generate an array of zeros
b=np.zeros(10)
print(b)
#to generate a array of ones
b=np.ones(10)
print(b)
```

```
int32
3
(2, 2, 4)
16
[10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50 52 54 56
 58 60 62 64 66 68 70 72 74 76 78 80 82 84 86 88 90 92 94 96 98]
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
```

In [3]:
```
#Loading an example dataset
from sklearn import datasets
iris = datasets.load_iris()
digits = datasets.load_digits()
```

In [15]:
```
#Task No 1

import numpy as np

x = np.random.rand(10, 4)

print("\nOriginal array:")
print(x)

y = x[:5, :]

print("\nFirst 5 rows of the above array:")
print(y)
```

```
Original array:
[[0.45638837 0.30099651 0.15797998 0.78256523]
 [0.06283015 0.24717785 0.08985883 0.83489706]
 [0.74489734 0.81480129 0.56449061 0.71883719]
 [0.733074   0.76852731 0.67058921 0.48522199]
 [0.07191213 0.84205563 0.7112761  0.12664657]
 [0.83471874 0.05409502 0.24275927 0.47548663]
 [0.85457395 0.94587432 0.76636105 0.0954728 ]
 [0.66286294 0.64041693 0.13005505 0.710507  ]
 [0.79978782 0.27660149 0.34924898 0.82552885]
 [0.19499456 0.37923555 0.28797709 0.6150078 ]]

First 5 rows of the above array:
[[0.45638837 0.30099651 0.15797998 0.78256523]
 [0.06283015 0.24717785 0.08985883 0.83489706]
 [0.74489734 0.81480129 0.56449061 0.71883719]
 [0.733074   0.76852731 0.67058921 0.48522199]
 [0.07191213 0.84205563 0.7112761  0.12664657]]
```

In [9]:

```
#Task 2

import pandas as pd
import numpy as np

exam_data = {
    'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jona
    'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
    'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
    'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']
}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
df = pd.DataFrame(exam_data, index=labels)

print(df)

result = df[df['attempts'] > 2]

print("\nNumber of attempts in the examination is greater than 2:")
print(result)
```

```
        name  score  attempts qualify
a  Anastasia   12.5         1     yes
b       Dima    9.0         3      no
c  Katherine   16.5         2     yes
d      James    NaN         3      no
e      Emily    9.0         2      no
f    Michael   20.0         3     yes
g    Matthew   14.5         1     yes
h      Laura    NaN         1      no
i      Kevin    8.0         2      no
j      Jonas   19.0         1     yes

Number of attempts in the examination is greater than 2:
      name  score  attempts qualify
b     Dima    9.0         3      no
d    James    NaN         3      no
f  Michael   20.0         3     yes
```

In [11]:
```
#Task 3
import numpy as np

exam_data = {
    'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew',
             'Laura', 'Kevin', 'Jonas'],
    'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
    'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
    'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']
}

average_score = np.nanmean(exam_data['score'])

print("The average score is:", average_score)
```

```
The average score is: 13.5625
```

In [21]:
```
import pandas as pd
import numpy as np
exam_data = {
    'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jona
    'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
    'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
    'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']
}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

df = pd.DataFrame(exam_data, index=labels)
print(df)

result = df[df['attempts'] > 2]

print("\nNumber of attempts in the examination is greater than 2:")
print(result)
```

```
        name  score  attempts qualify
a  Anastasia   12.5         1     yes
b       Dima    9.0         3      no
c  Katherine   16.5         2     yes
d      James    NaN         3      no
e      Emily    9.0         2      no
f    Michael   20.0         3     yes
g    Matthew   14.5         1     yes
h      Laura    NaN         1      no
i      Kevin    8.0         2      no
j      Jonas   19.0         1     yes

Number of attempts in the examination is greater than 2:
      name  score  attempts qualify
b     Dima    9.0         3      no
d    James    NaN         3      no
f  Michael   20.0         3     yes
```

In [22]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
data=pd.read_csv(r'C:\Users\Student\Desktop\iris-flower-dataset.csv')
print(data.head())
x=data.drop('species','columns')
y=data['species']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20)
#print(x_train)
scaler=StandardScaler()
scaler.fit(x_train)
#print(x_train)
x_train=scaler.transform(x_train)
x_test=scaler.transform(x_test)
print(x_train)
print(x_test)
from sklearn.neighbors import KNeighborsClassifier
classifier=KNeighborsClassifier(n_neighbors=3)
classifier.fit(x_train,y_train)
result=classifier.predict(x_test)
print(result)
from sklearn.metrics import confusion_matrix,classification_report
print(classification_report(y_test,result))
print(confusion_matrix(y_test,result))
```

```
 'Iris-setosa' 'Iris-versicolor' 'Iris-setosa' 'Iris-versicolor'
 'Iris-setosa' 'Iris-virginica' 'Iris-setosa' 'Iris-virginica'
 'Iris-setosa' 'Iris-virginica' 'Iris-virginica' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-setosa' 'Iris-versicolor' 'Iris-setosa'
 'Iris-setosa' 'Iris-setosa' 'Iris-virginica' 'Iris-setosa'
 'Iris-virginica' 'Iris-versicolor' 'Iris-virginica' 'Iris-versicolor'
 'Iris-setosa' 'Iris-virginica']
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        11
Iris-versicolor       1.00      0.73      0.84        11
 Iris-virginica       0.73      1.00      0.84         8

       accuracy                           0.90        30
      macro avg       0.91      0.91      0.89        30
   weighted avg       0.93      0.90      0.90        30

[[11  0  0]
 [ 0  8  3]
 [ 0  0  8]]
```

In [23]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix

data = pd.read_csv(r'C:\Users\Student\Desktop\iris-flower-dataset.csv')

x = data.drop('species', axis=1)
y = data['species']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20)

# Standardize the features
scaler = StandardScaler()
scaler.fit(x_train)
x_train = scaler.transform(x_train)
x_test = scaler.transform(x_test)

classifier = SVC(kernel='linear')
classifier.fit(x_train, y_train)

result = classifier.predict(x_test)

print("Classification Report:")
print(classification_report(y_test, result))
print("Confusion Matrix:")
print(confusion_matrix(y_test, result))
```

```
Classification Report:
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        11
Iris-versicolor       1.00      1.00      1.00         7
 Iris-virginica       1.00      1.00      1.00        12

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

Confusion Matrix:
[[11  0  0]
 [ 0  7  0]
 [ 0  0 12]]
```

```
In [ ]:
```