1.Write a program to distinguish between Array Indexing and Fancy Indexing.

```
import numpy as np

#array indexing
arr=np.array([2,4,6,8,10,12])
print(arr)

#fancy indexing
fancy_array = np.array([2,4,6,8,10,12])
selected_index = fancy_array[[0,2,3,5]]
print(selected_index)
```

```
[ 2  4  6  8 10 12]
[ 2  6  8 12]
```

2.Execute the 2D array Slicing.

```
import numpy as np
arr=np.array([[1,2,3,4,5,6],[7,8,9,10,11,12]])
print(arr[1][1:5])
```

```
[ 8  9 10 11]
```

3.Create the 5-Dimensional arrays using 'ndmin'.

```
import numpy as np
arr1=np.array([9,8,7,6],ndmin=5)
print("The 5 dimensional array using ndmin is : ",arr1)
```

```
The 5 dimensional array using ndmin is :  [[[[[9 8 7 6]]]]]
```

4.Reshape the array from 1-D to 2-D array.

```
import numpy as np
arr=np.array([0,9,8,7,6,5,4,3,2,1])
print("The original array :",arr)
reshapearr=arr.reshape(2,5)
print("Array after reshapeing : \n",reshapearr)
```

```
The original array : [0 9 8 7 6 5 4 3 2 1]
Array after reshapeing :
 [[0 9 8 7 6]
 [5 4 3 2 1]]
```

5.Perform the Stack functions in Numpy arrays – Stack(), hstack(), vstack(), and dstack().

```
import numpy as np
arr1=np.array([00,11,22,33,44,55,66,77,88,99])
```

```
print("First array : ",arr1)
arr2=np.array([10,20,30,40,50,60,70,80,90,100])
print("Second array : ",arr2)

#Using stack
sarr=np.stack((arr1,arr2))
print("\nArrays after using stack : \n",sarr)

#Using hstack
harr=np.hstack((arr1,arr2))
print("\nArrays after using hstack : \n",harr)

#Using vstack
varray=np.vstack((arr1,arr2))
print("\nArrays after using vstack : \n",varray)

#Using dstack
darray=np.dstack((arr1,arr2))
print("\nArrays after using dstack :\n",darray)
```

```
First array :  [ 0 11 22 33 44 55 66 77 88 99]
Second array :  [ 10  20  30  40  50  60  70  80  90 100]

Arrays after using stack :
 [[  0  11  22  33  44  55  66  77  88  99]
 [ 10  20  30  40  50  60  70  80  90 100]]

Arrays after using hstack :
 [  0  11  22  33  44  55  66  77  88  99  10  20  30  40  50  60  70
80
  90 100]

Arrays after using vstack :
 [[  0  11  22  33  44  55  66  77  88  99]
 [ 10  20  30  40  50  60  70  80  90 100]]

Arrays after using dstack :
 [[[  0  10]
  [ 11  20]
  [ 22  30]
  [ 33  40]
  [ 44  50]
  [ 55  60]
  [ 66  70]
  [ 77  80]
  [ 88  90]
  [ 99 100]]]
```

6.Perform the searchsort method in Numpy array.

```
import numpy as np
arr=np.array([12,23,34,45,56,67,78])

sarr=np.searchsorted(arr,[45,56,78,12])
print("Search Sort : ",sarr)

sarray=np.searchsorted(arr,23,side='left')
print("(using side) : ",sarray)
```

```
Search Sort :  [3 4 6 0]
(using side) :  1
```

7.Create Numpy Structured array using your domain features.

```
import numpy as np
me=np.array([('sona','agricultural sector',2347258)],
            dtype=[('name',(np.str_,10)),('age',np.str_,100),
('registerno',np.int32)])
print("My domain features : ",me)
```

```
My domain features :  [('sona', 'agricultural sector', 2347258)]
```

8.Create Data frame using List and Dictionary.

```
import pandas as pd
#Dataframe using Dictionary
df={
    'Books':['Aarachar','Harry Potter 1','Percy Jackson','Chemmeen'],
    'Year':[2012,1997,2010,1956]
}
mybooks=pd.DataFrame(df)
print(mybooks)

#DataFrame using list
mylist=[2012,1997,2010,1956]
mydata=pd.DataFrame(mylist,index=['a','b','c','d'],
                    columns=['Years'])
print("\n\n",mydata)
```

```
            Books  Year
0        Aarachar  2012
1  Harry Potter 1  1997
2   Percy Jackson  2010
3        Chemmeen  1956


    Years
a   2012
b   1997
```

```
c   2010
d   1956
```

9.Create Data frame on your Domain area and perform the following operations to find and eliminate the missing data from the dataset. • isnull() • notnull() • dropna() • fillna() • replace() • interpolate()

```
import pandas as pd
import numpy as np
data = {
    'Year': [2010, 2011, 2012, 2013, 2014],
    'Crop_Yield': [500, 600, np.nan, 700, 750],
    'Rainfall_mm': [800, 850, 900, np.nan, 950]
}

df = pd.DataFrame(data)


# Check for missing data in the entire DataFrame
print(df.isnull())

# Check for non-missing data in the entire DataFrame
print(df.notnull())


# Drop rows with any missing data
df.dropna(axis=0, inplace=True)

# Drop columns with any missing data
df.dropna(axis=1, inplace=True)



# Fill missing values in the 'Crop_Yield' column with the mean value
of that column
mean_yield = df['Crop_Yield'].mean()
df['Crop_Yield'].fillna(mean_yield, inplace=True)

# Fill missing values in the 'Rainfall_mm' column with a specific
value, e.g., 0
df['Rainfall_mm'].fillna(0, inplace=True)



# Replace a specific value in the DataFrame
df.replace(700, 720, inplace=True)

# Replace multiple values in the DataFrame
df.replace({600: 620, 750: 770}, inplace=True)
```

```
# Interpolate missing values in a column using linear interpolation
df['Rainfall_mm'].interpolate(method='linear', inplace=True)
```

```
    Year  Crop_Yield  Rainfall_mm
0  False       False        False
1  False       False        False
2  False        True        False
3  False       False         True
4  False       False        False
   Year  Crop_Yield  Rainfall_mm
0  True        True         True
1  True        True         True
2  True       False         True
3  True        True        False
4  True        True         True
```

Q10. Perform the Hierarchical Indexing in the above created dataset.

```
import pandas as pd
import numpy as np

# Create the DataFrame
data = {
    'Year': [2010, 2011, 2012, 2013, 2014],
    'Crop_Yield': [500, 600, np.nan, 700, 750],
    'Rainfall_mm': [800, 850, 900, np.nan, 950]
}

df = pd.DataFrame(data)

# Perform Hierarchical Indexing
df.set_index(['Year', 'Crop_Yield'], inplace=True)

# Access data using hierarchical index
value = df.loc[(2010, 500)]

# Select all data for the year 2011
year_2011_data = df.xs(key=2011, level='Year')

# Select all data for a specific crop yield (e.g., 600)
crop_yield_600_data = df.xs(key=600, level='Crop_Yield')

print("Hierarchical Indexing Result:")
print(df)
print("\nAccessing Data:")
print("Value at (2010, 500):", value)
print("\nData for Year 2011:")
```

```python
print(year_2011_data)
print("\nData for Crop Yield 600:")
print(crop_yield_600_data)
```

```
Hierarchical Indexing Result:
                 Rainfall_mm
Year Crop_Yield
2010 500.0             800.0
2011 600.0             850.0
2012 NaN               900.0
2013 700.0               NaN
2014 750.0             950.0

Accessing Data:
Value at (2010, 500): Rainfall_mm    800.0
Name: (2010, 500.0), dtype: float64

Data for Year 2011:
            Rainfall_mm
Crop_Yield
600.0             850.0

Data for Crop Yield 600:
      Rainfall_mm
Year
2011        850.0
```