Python Advance Assignment 2

1. Explain three-dimensional data indexing.

Make three two-dimensional arrays of shape 3×3

x = np.arange(27).reshape(3,3,3)

Selecting the Two-Dimensional Arrays

```
>>> x[0]
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

Print the second row of first two-dimensional array

```
>>> x[0][1]
array([3, 4, 5])
```

Get the element 14 from the array.

```
>>> x[1][1][2]
14
```

Return the first rows of the last two two-dimensional array.

```
>>> x[1:, 0]
array([[ 9, 10, 11],
       [18, 19, 20]])
```

Slice through both columns and rows and print part of first two rows of the last two two-dimensional arrays

```
>>> x[1:, 0:2, 1:2]
array([[[10],
        [13]],
       [[19],
        [22]]])
```

Some other cases

```
>>> arr[:, :, 2]
array([[ 2,  5,  8],
       [11, 14, 17],
       [20, 23, 26]])
>>> arr[0:2, 1, :]
array([[ 3,  4,  5],
       [12, 13, 14]])
>>> arr[0, :2, 2]
array([2, 5])
>>> arr[1, :, ::-1]
array([[11, 10,  9],
       [14, 13, 12],
       [17, 16, 15]])
```

2. What's the difference between a series and a dataframe?

Series is a one-dimensional labeled array capable of holding any data type (integers, strings, floating point numbers, Python objects, etc.). The axis labels are collectively referred to as the index.

DataFrame is a 2-dimensional tabular labeled data structure with columns of potentially different types. You can think of it like a spreadsheet or SQL table, or a dict of Series objects. It is generally the most commonly used pandas object.

Use: Series: Series are for one-dimensional data, just like lists with a lot of functions. DataFrame: DataFrames are for multi-dimensional data, just like nested lists with a lot of functions.

3. What role does pandas play in data cleaning?

As we know that, Data Science is the discipline of study which involves extracting insights from huge amounts of data by the use of various scientific methods, algorithms, and processes. To extract useful knowledge from data, Data Scientists need raw data. This Raw data is a collection of information from

various outlines sources and an essential raw material of Data Scientists. It is additionally known as primary or source data. It consists of garbage, irregular and inconsistent values which lead to many difficulties. When using data, the insights and analysis extracted are only as good as the data we are using. Essentially, when garbage data is in, then garbage analysis comes out. Here Data cleaning comes into the picture, Data cleansing is an essential part of data science. Data cleaning is the process of removing incorrect, corrupted, garbage, incorrectly formatted, duplicate, or incomplete data within a dataset.

data cleaning: When working with multiple data sources, there are many chances for data to be incorrect, duplicated, or mislabeled. If data is wrong, outcomes and algorithms are unreliable, even though they may look correct. Data cleaning is the process of changing or eliminating garbage, incorrect, duplicate, corrupted, or incomplete data in a dataset. There's no such absolute way to describe the precise steps in the data cleaning process because the processes may vary from dataset to dataset. Data cleansing, data cleansing, or data scrub is that the initiative among the general data preparation process. Data cleaning plays an important part in developing reliable answers and within the analytical process and is observed to be a basic feature of the info science basics. The motive of data cleaning services is to construct uniform and standardized data sets that enable data analytical tools and business intelligence easy access and perceive accurate data for each problem.

Data cleaning with Pandas: Data scientists spend a huge amount of time cleaning datasets and getting them in the form in which they can work. It is an essential skill of Data Scientists to be able to work with messy data, missing values, inconsistent, noise, or nonsensical data. To work smoothly python provides a built-in module Pandas. Pandas is the popular Python library that is mainly used for data processing purposes like cleaning, manipulation, and analysis. Pandas stand for “Python Data Analysis Library”. It consists of classes to read, process, and write CSV data files. There are numerous Data cleaning tools present but, the Pandas library provides a really fast and efficient way to manage and explore data. It does that by providing us with Series and DataFrames, which help us not only to represent data efficiently but also manipulate it in various ways.

4. How do you use pandas to make a data frame out of n-dimensional arrays?

It is very simple to convert a 2-dimensional array into a data frame because pandas data frame objects are already 2-dimensional data structures.

For n-dimensional arrays we need to use itertools

```
import numpy as np

import pandas as pd

data = np.arange(27).reshape(3,3,3)

import itertools
```

```
data  = list(itertools.chain(*data))

df = pd.DataFrame.from_records(data)
```

Without itertools

```
data = [i for j in data for i in j]

df = pd.DataFrame.from_records(data)
```

we can use flatten() method directly

```
pd.DataFrame(data.flatten(),columns = ['col1'])
```

5. Explain the notion of pandas plotting.

```
import pandas as pd

import numpy as np

df = pd.DataFrame(np.random.randn(1000, 4), columns=list("ABCD"))

df = df.cumsum()

df.plot();

df.iloc[5].plot(kind="bar");
```

Plotting methods allow for a handful of plot styles other than the default line plot. These methods can be provided as the kind keyword argument to plot(), and include:

'bar' or 'barh' for bar plots, 'hist' for histogram, 'box' for boxplot, 'kde' or 'density' for density plots, 'area' for area plots, 'scatter' for scatter plots, 'hexbin' for hexagonal bin plots, 'pie' for pie plots

We can also create these other plots using the methods DataFrame.plot.<kind> instead of providing the kind keyword argument.

Eg

```
df2 = pd.DataFrame(np.random.rand(10, 4), columns=["a", "b", "c", "d"])

df2.plot.bar();

df2.plot.bar(stacked=True);

df2.plot.barh(stacked=True);

df = pd.DataFrame(np.random.rand(10, 3), columns=["Col1", "Col2", "Col3"])
```

```
df["X"] = pd.Series(["A", "A", "A", "A", "A", "B", "B", "B", "B", "B"])

bp = df.plot.box(column=["Col1", "Col2"], by="X")

df.plot.area();

df = pd.DataFrame(3 * np.random.rand(4, 2), index=["a", "b", "c", "d"], columns=["x", "y"])

df.plot.pie(subplots=True, figsize=(8, 4));
```

Andrews curves

```
from pandas.plotting import andrews_curves, parallel_coordinates, lag_plot, bootstrap_plot, radviz

data = pd.read_csv(r"data/iris.csv")

andrews_curves(data, "Name");

parallel_coordinates(data, "Name");

spacing = np.linspace(-99 * np.pi, 99 * np.pi, num=1000)

data = pd.Series(0.1 * np.random.rand(1000) + 0.9 * np.sin(spacing))

lag_plot(data);
```