Name : Parth Nandedkar
Date : 01 Feb 2024
Topics : Python (Numpy, Pandas, Dataframe APIs, Dataframe using dynamic column list)
Batch : Data Engineering Batch-1

Numpy :

NumPy, which stands for Numerical Python, is a powerful library in Python used for numerical and scientific computing. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays. Here's an overview of NumPy:

N-dimensional Arrays (ndarray):
NumPy's main object is the ndarray, which is a multidimensional array of elements of the same type.
Arrays can be one-dimensional, two-dimensional, or even higher-dimensional.

Array Creation:
Arrays in NumPy can be created using various functions like numpy.array(), numpy.zeros(), numpy.ones(), numpy.arange(), etc.

Array Operations:
NumPy provides a wide range of mathematical operations that can be performed directly on arrays, including element-wise operations, linear algebra operations, statistical operations, and more.

Universal Functions (ufunc):
NumPy functions that operate element-wise on arrays are called universal functions (ufunc).
These functions are highly optimized and efficiently handle large datasets.
Broadcasting:

NumPy supports broadcasting, which allows operations between arrays of different shapes and sizes. The smaller array is broadcasted across the larger array to perform the operation.
Indexing and Slicing:

NumPy provides powerful indexing and slicing capabilities for accessing and manipulating elements of arrays.

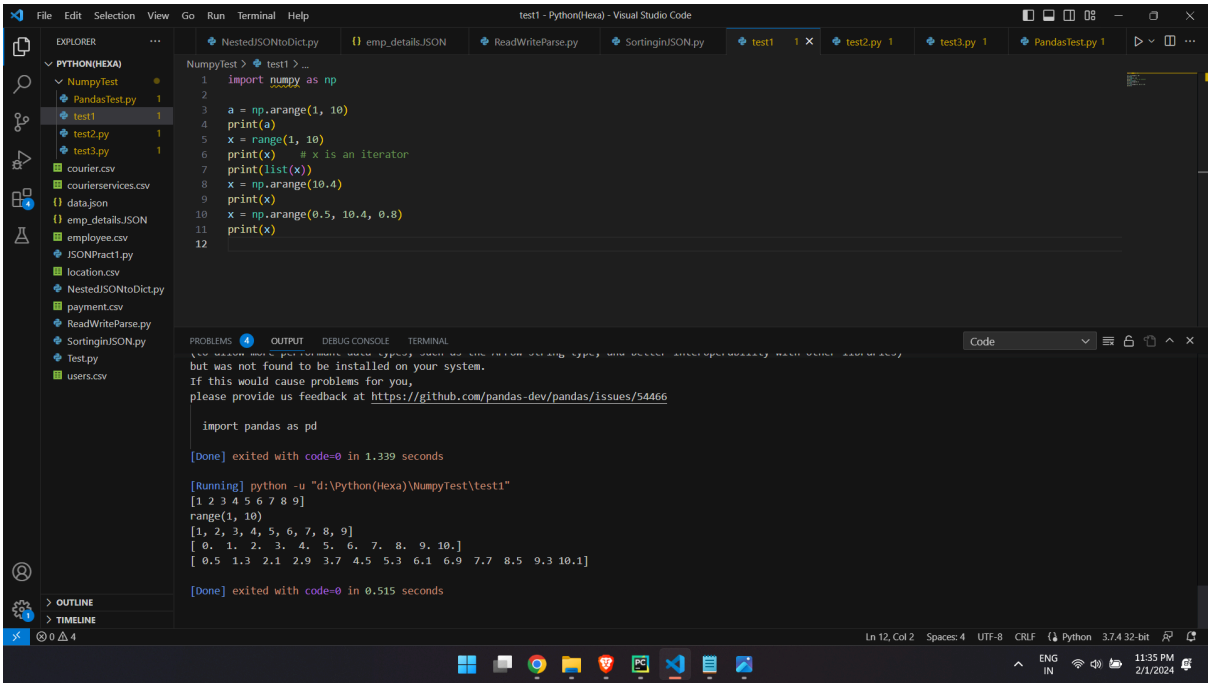
Linear Algebra:
NumPy includes a linear algebra module (numpy.linalg) that provides functions for matrix operations, eigenvalue problems, solving linear equations, and more.

Random Module:
The numpy.random module provides functions for generating random numbers and distributions.

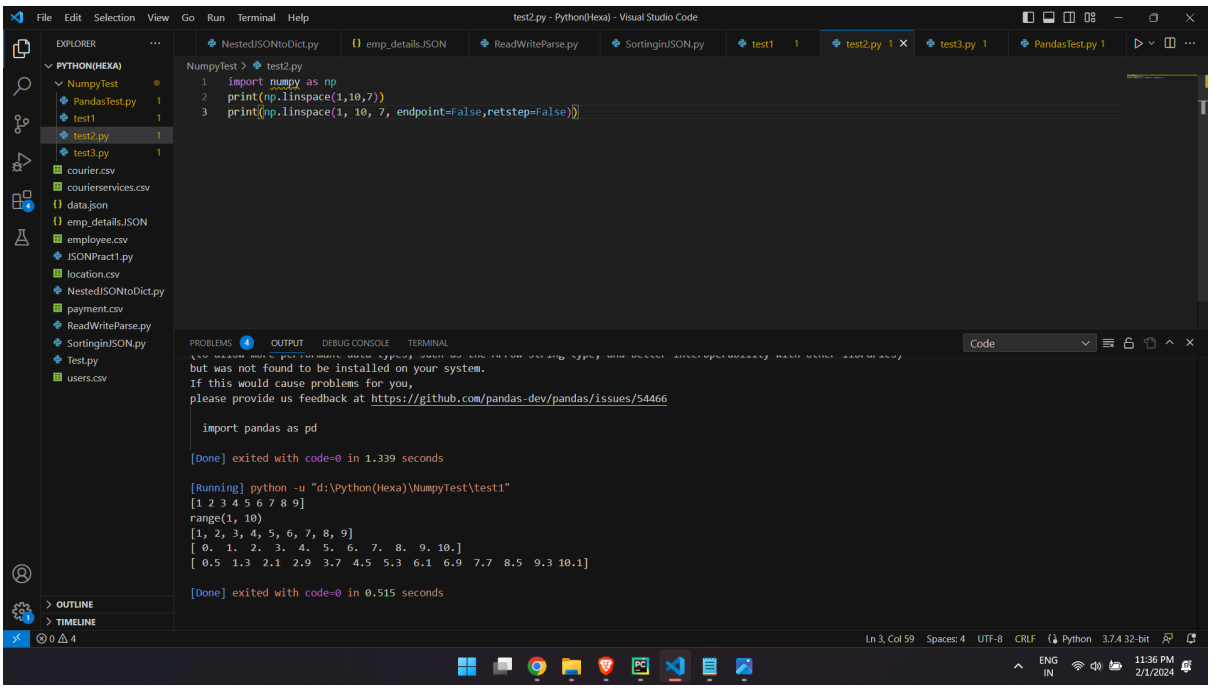
Performance:
NumPy is designed for efficiency and performance. Many of its operations are implemented in C and Fortran, making them faster than equivalent operations in pure Python.

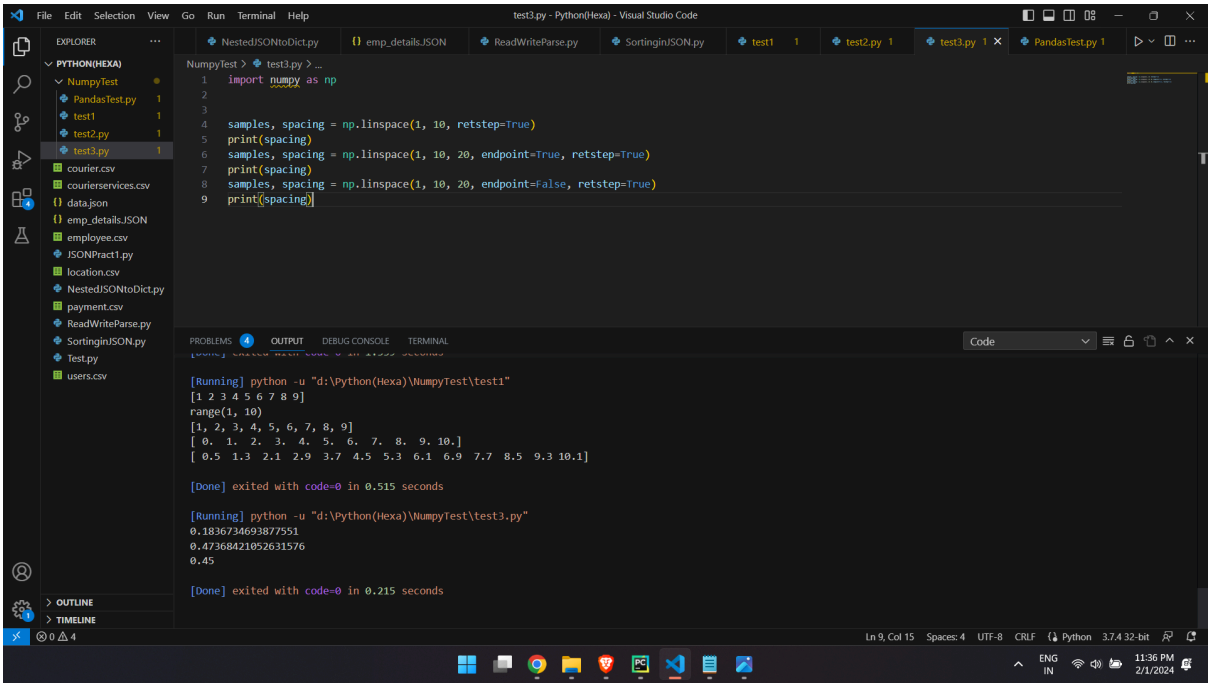
Interoperability:
NumPy arrays can be seamlessly integrated with other libraries written in languages like C and Fortran, making it a foundational component in the scientific Python ecosystem.

Example of creating NumPy array using arrange() and range() methods:



Examples on spacing using linespace() :





Pandas :

Pandas is a popular open-source data manipulation and analysis library for Python. It provides easy-to-use data structures such as Series and DataFrame, along with a variety of functions to perform efficient data manipulation, cleaning, and analysis. Here's a brief introduction to Pandas:

DataFrame:
The central data structure in Pandas is the DataFrame, which is a two-dimensional, labeled data structure with columns that can be of different data types.
It can be thought of as a table or spreadsheet.

Series:
A Series is a one-dimensional labeled array and is often used to represent a single column or row in a DataFrame.

Data Cleaning and Preparation:
Pandas provides functions for handling missing data, removing duplicates, and reshaping data for analysis.

Data Indexing and Selection:
Pandas offers powerful indexing and selection capabilities, allowing for easy retrieval and manipulation of data.

Data Alignment and Merging:
Data alignment is intrinsic to Pandas, and it automatically aligns data based on label indices.
The library provides functions for merging and joining datasets.

GroupBy:
Pandas supports the "split-apply-combine" paradigm with the groupby operation, enabling efficient grouping and aggregation of data.

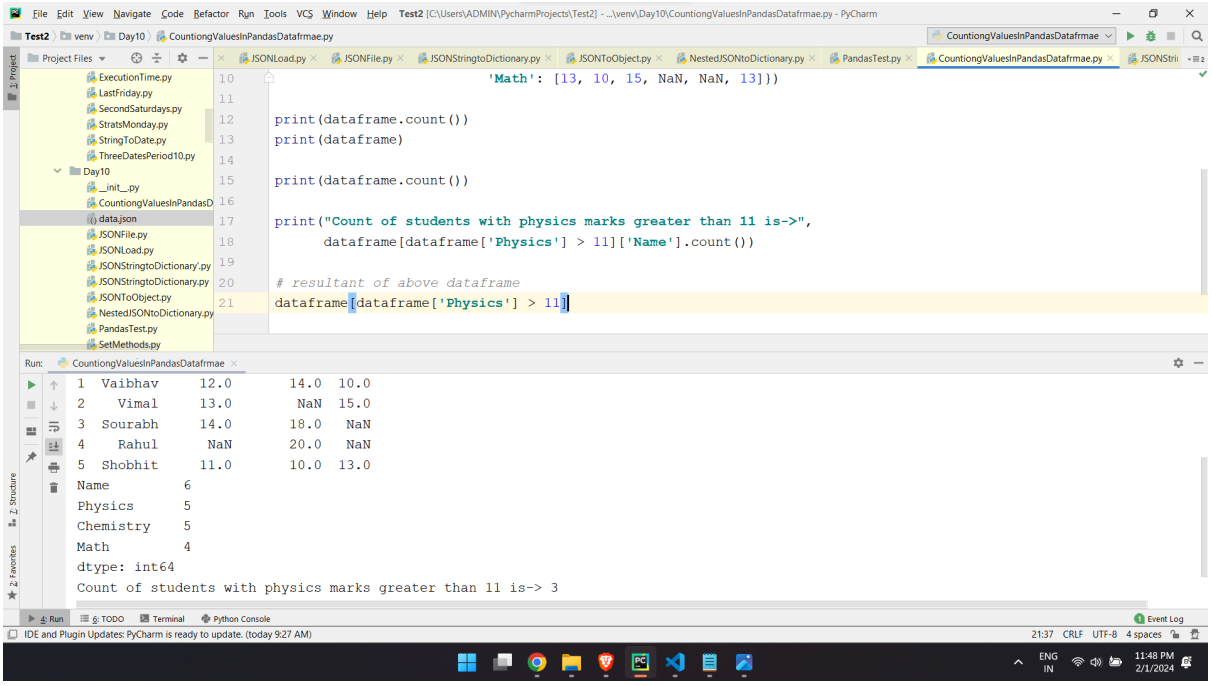
Time Series and Date Functionality:
Pandas provides robust support for time-based data, including date range generation, frequency conversion, and resampling.

Input and Output (I/O):
Pandas supports reading and writing data in various formats, including CSV, Excel, SQL databases, and more.

Statistical Analysis:
Pandas includes a range of statistical functions for descriptive statistics, correlation, and other summary measures.

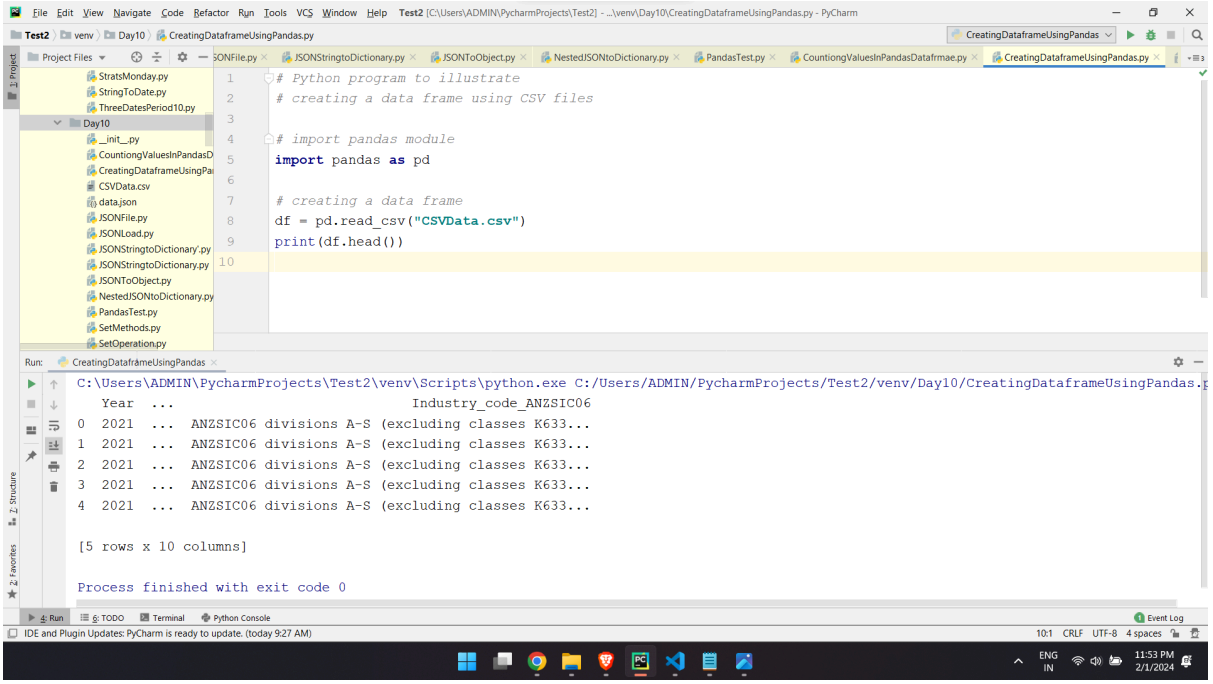
Integration with NumPy:
Pandas is built on top of NumPy, making it seamlessly integrated with other scientific computing libraries in the Python ecosystem.

Count Values in Pandas Dataframe :

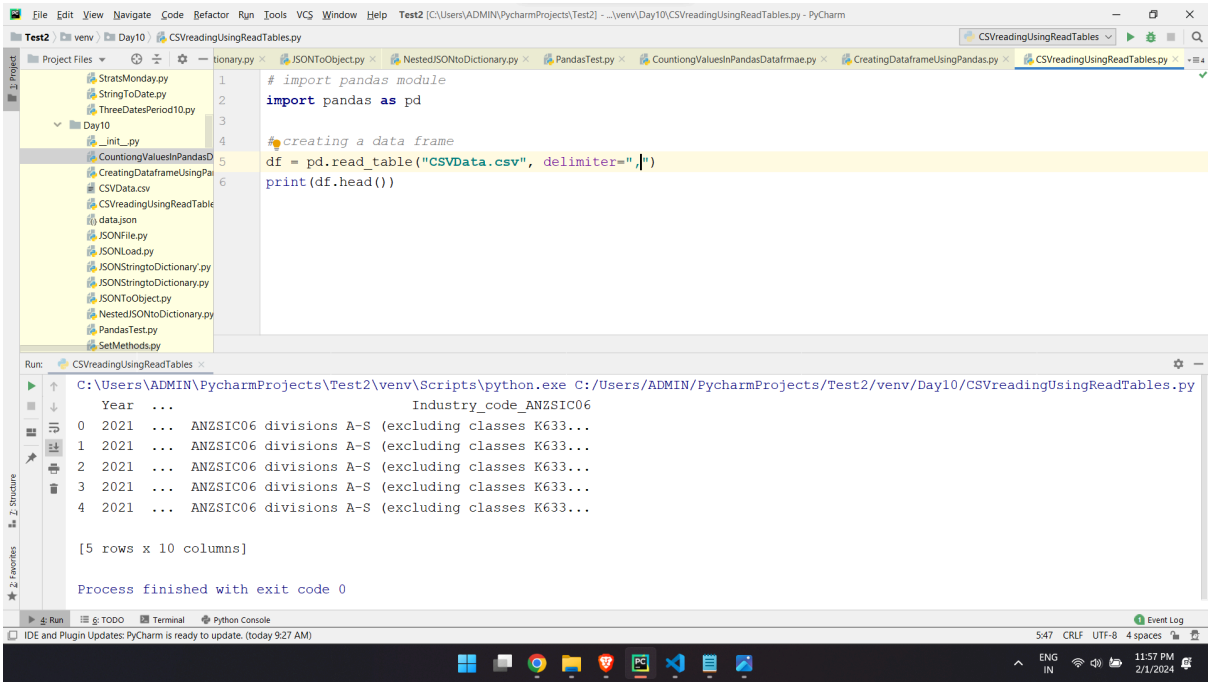


Dataframes using dynamic column list on CSV Data :

Pandas can be used to convert different types of big data into python objects according to need so we can it to convert CSV file into Pandas object :

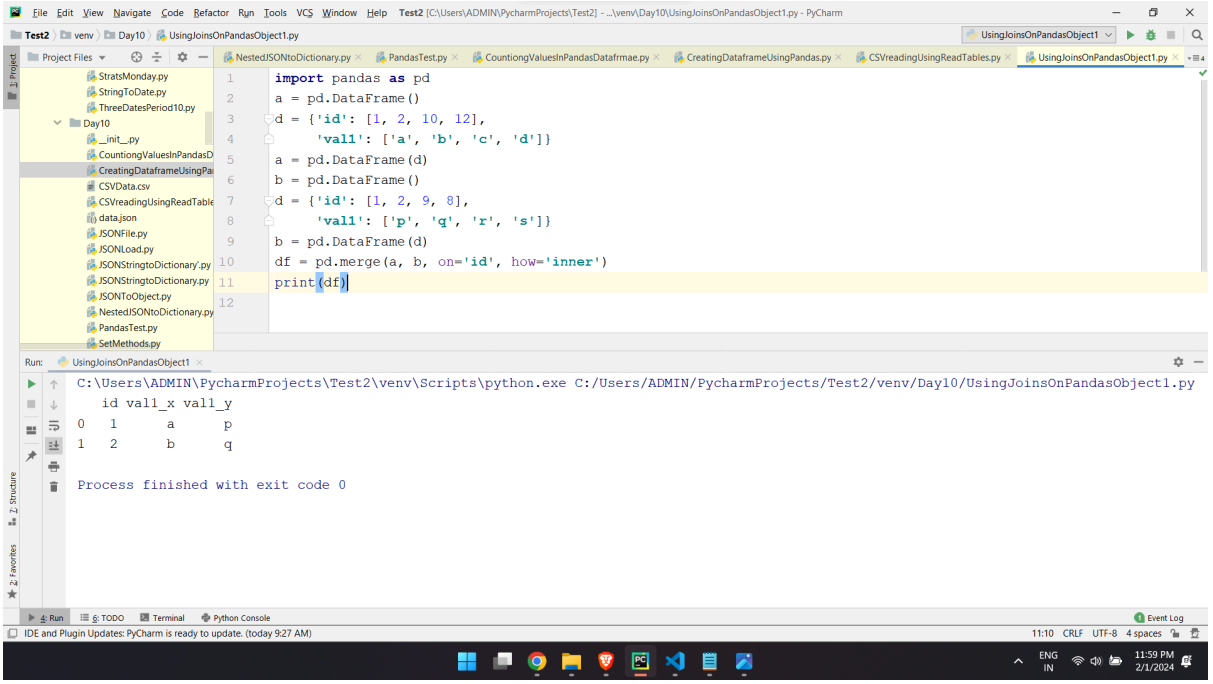


Using Readtable() :

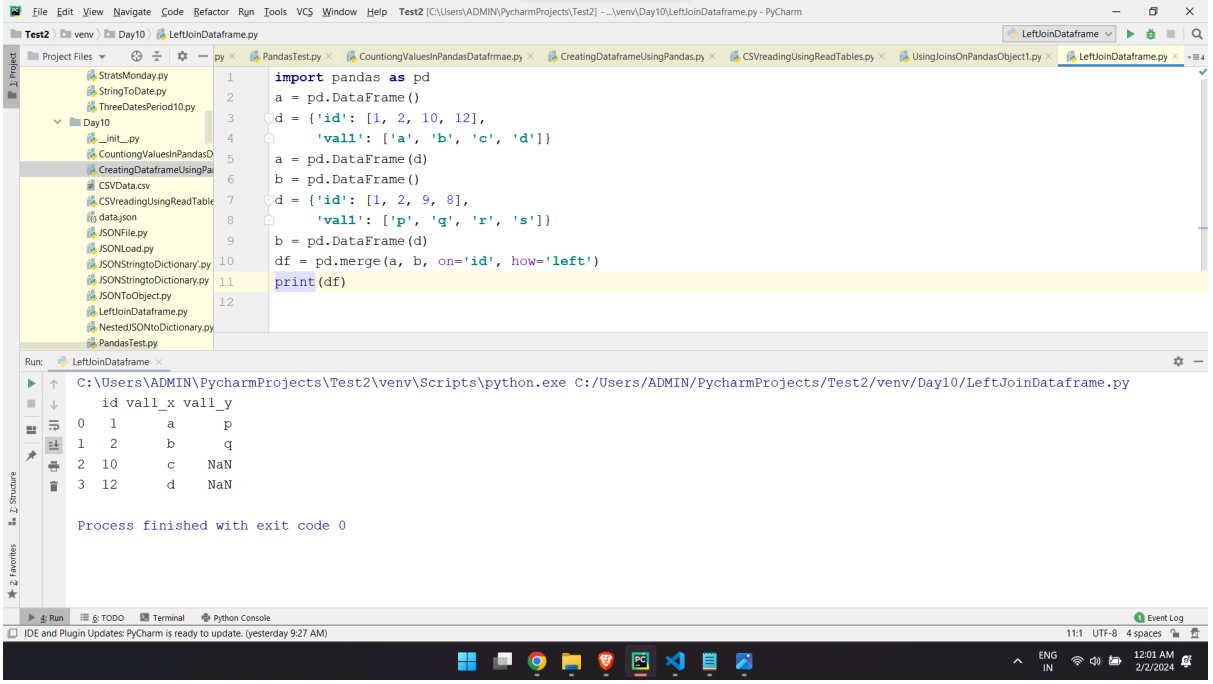


Using JOINS on Dataframes :

INNER JOIN :



LEFT JOIN :



RIGHT JOIN :

```python
import pandas as pd
a = pd.DataFrame()
d = {'id': [1, 2, 10, 12],
     'val1': ['a', 'b', 'c', 'd']}
a = pd.DataFrame(d)
b = pd.DataFrame()
d = {'id': [1, 2, 9, 8],
     'val1': ['p', 'q', 'r', 's']}
b = pd.DataFrame(d)
df = pd.merge(a, b, on='id', how='right')
print(df)
```

```
C:\Users\ADMIN\PycharmProjects\Test2\venv\Scripts\python.exe C:/Users/ADMIN/PycharmProjects/Test2/venv/Day10/RightJoinDataframe.py
   id val1_x val1_y
0   1      a      p
1   2      b      q
2   9    NaN      r
3   8    NaN      s

Process finished with exit code 0
```

FULL JOIN :

```python
import pandas as pd
a = pd.DataFrame()
d = {'id': [1, 2, 10, 12],
     'val1': ['a', 'b', 'c', 'd']}
a = pd.DataFrame(d)
b = pd.DataFrame()
d = {'id': [1, 2, 9, 8],
     'val1': ['p', 'q', 'r', 's']}
b = pd.DataFrame(d)
df = pd.merge(a, b, on='id', how='outer')
print(df)
```

```
C:\Users\ADMIN\PycharmProjects\Test2\venv\Scripts\python.exe C:/Users/ADMIN/PycharmProjects/Test2/venv/Day10/FullJoinDataframe.py
   id val1_x val1_y
0   1      a      p
1   2      b      q
2  10      c    NaN
3  12      d    NaN
4   9    NaN      r
5   8    NaN      s

Process finished with exit code 0
```