Assignment

Q1. Create a Pandas Series that contains the following data: 4, 8, 15, 16, 23, and 42. Then, print the series.

Ans:can create a Pandas Series using the `pd.Series()` constructor and then print the series. Here's an example:

python

Copy code

```
import        as


        4  8  15  16  23  42



print
```

This will output:

go

Copy code

```
0 4
1 8
2 15
3 16
4 23
5 42
       int64
```

In this example, the index values are automatically assigned (0, 1, 2, ...) by default. The data types in the series are inferred based on the provided data (int64 in this case).

Q2. Create a variable of list type containing 10 elements in it, and apply pandas.Series function on the
variable print it.

Ans:can create a Python list containing 10 elements and then use the `pd.Series()` function to convert it into a Pandas Series. Here's an example:

python

Copy code

```
import        as
```

```
            11  22  33  44  55  66  77  88  99  100



print
```

This will output:

go

Copy code

```
0 11
1 22
2 33
3 44
4 55
5 66
6 77
7 88
8 99
9 100
       int64
```

In this example, the list is converted into a Pandas Series with default index values (0 to 9) and the inferred data type (`int64`).

Q3. Create a Pandas DataFrame that contains the following data:
Ans:Name
Alice
Bob
Claire

Age
25
30
27

Gender
Female
Male
Female
Then, print the DataFrame.
Ans:can create a Pandas DataFrame using the `pd.DataFrame()` constructor with the provided data. Here's an example:
python

Copy code

```
import        as



 'Name'    'Alice'   'Bob'   'Claire'
 'Age'    25   30   27
 'Gender'    'Female'   'Male'   'Female'




print
```

This will output:

markdown

Copy code

In this example, the keys of the dictionary represent the column names, and the associated lists represent the column values. The DataFrame is printed with default index values (0, 1, 2).

Q4. What is 'DataFrame' in pandas and how is it different from pandas.series? Explain with an example.
Ans:In Pandas, a `DataFrame` is a two-dimensional, tabular data structure that consists of rows and columns. It can be thought of as an Excel spreadsheet or a SQL table. Each column in a

DataFrame can be of a different data type (e.g., integers, floats, strings) and has a column label. Additionally, a DataFrame has an index, which labels the rows.

On the other hand, a `Series` is a one-dimensional array-like object that can hold any data type. It is essentially a single column of a DataFrame and has an associated index. You can think of a Series as a single column of data with an index.

Here's an example to illustrate the difference between a DataFrame and a Series:

python

Copy code

```
import        as

                1  2  3  4        'MySeries'


 'Column_A'   1  2  3  4
 'Column_B'   'A'  'B'  'C'  'D'


print "Series:"
print
print "\nType of Series:"  type


print "\nDataFrame:"
print
print "\nType of DataFrame:"  type
```

Output:

vbnet

Copy code

```
Series:
0 1
1 2
```

```
2 3
3 4
Name:

     of          class

DataFrame:

0 1
1 2
2 3
3 4

     of             class
```

In the example, `ser` is a Series with values `[1, 2, 3, 4]`, and `df` is a DataFrame with two columns (`Column_A` and `Column_B`). The key difference is that a DataFrame is a 2D structure with columns, while a Series is a 1D structure with just one column.

In summary:

- A DataFrame is a 2D tabular data structure with rows and columns.
- A Series is a 1D array-like structure, representing a single column of data in a DataFrame.

Q5. What are some common functions you can use to manipulate data in a Pandas DataFrame? Can
you give an example of when you might use one of these functions?
Ans:Pandas provides a wide range of functions to manipulate and analyze data in a DataFrame. Some common functions include:

head() and tail():

- Use: Display the first or last n rows of the DataFrame.
- Example:
- python
- Copy code

```
3
```

-

info():

- Use: Display a concise summary of the DataFrame, including data types and missing values.
- Example:
- python
- Copy code
-

describe():

- Use: Generate descriptive statistics of the DataFrame's numerical columns.
- Example:
- python
- Copy code
-

shape:

- Use: Return the number of rows and columns in the DataFrame.
- Example:
- python
- Copy code
-

columns:

- Use: Return the column labels of the DataFrame.
- Example:
- python
- Copy code

-

value_counts():

- Use: Count unique values in a column.
- Example:
- python
- Copy code

```
'Category'
```

-

sort_values():

- Use: Sort the DataFrame by one or more columns.
- Example:
- python
- Copy code

```
        'Age'            False
```

-

drop():

- Use: Remove specified rows or columns from the DataFrame.
- Example:
- python
- Copy code

```
    0  1            'Column_A'
```

-

fillna():

- Use: Fill missing values in the DataFrame with a specified value or a method.
- Example:
- python

- Copy code

```
0
```

-

groupby():

- Use: Group the DataFrame based on a column and perform aggregate operations.
- Example:
- python
- Copy code

```
'Category'  'Value'
```

-

merge():

- Use: Merge two DataFrames based on a common column or index.
- Example:
- python
- Copy code

```
'common_column'
```

-

pivot_table():

- Use: Create a pivot table based on the DataFrame.
- Example:
- python
- Copy code

```
'Category'        'Value'
'mean'
```

-

These are just a few examples of the many functions available in Pandas for data manipulation. The choice of which function to use depends on the specific task or analysis you are performing on your data.

Q6. Which of the following is mutable in nature Series, DataFrame, Panel?

Ans:In Pandas, among Series, DataFrame, and Panel, only the DataFrame is mutable in nature.

- DataFrame: DataFrames are mutable, meaning you can modify the contents of a DataFrame after it has been created. You can add or remove columns, update values, and perform various other operations on the DataFrame.
- Series: Series are immutable. Once a Series is created, you cannot change its elements. However, you can create a new Series with the desired modifications.
- Panel: Panels were deprecated in recent versions of Pandas, and it is recommended to use multi-index DataFrames or Panel4D instead. Similar to DataFrames, multi-index DataFrames are mutable.

It's important to note that while DataFrames are mutable, it's often a good practice to avoid in-place modifications whenever possible and prefer creating new modified DataFrames to maintain data integrity.

Q7. Create a DataFrame using multiple Series. Explain with an example.

Ans:can create a DataFrame using multiple Series by combining them together. Here's an example:

python

Copy code

```
import        as


                'Alice'   'Bob'   'Claire'
               25  30  27
                  'Female'   'Male'   'Female'



 'Name'
 'Age'
 'Gender'



print
```

Output:

markdown

Copy code

In this example:

- Three Series (`names`, `ages`, `genders`) are created independently, each representing a column of the DataFrame.
- These Series are then combined into a DataFrame using the `pd.DataFrame()` constructor, where the keys of the dictionary represent the column names.
- The resulting DataFrame (`df`) has three columns: 'Name', 'Age', and 'Gender'.

This is a simple example, but in practice, you might be working with more complex data, and combining multiple Series into a DataFrame allows you to organize and analyze the data more effectively.