

# PYTHON IN 10 VIDEOS

Assignment - 7

# 1. How to create pandas series?

- a. `pd.series([1,2,3])`
- b. `pd.series({a:1, b:2, c:3})`
- c. `pd.series((1,2,3))`
- d. All the above

**Ans: d**

## 2. Guess output?

- a. [False, False, False]
- b. [True, True, True]
- c. [False, True, False]
- d. [True, True, False]

```
import pandas as pd  
s1 = pd.Series([1, 2, 3])  
s2 = pd.Series([4, 5, 6])  
s3 = s1 > s2  
print(s3)
```

**Ans:a**

### 3. Syntax for accessing a column in dataframe:

- a. `df [column_name]`
- b. `df.column_name`
- c. Both a & b
- d. Neither a nor b

**Ans: c**

## 4. Diff between loc & iloc?

- a. Loc - label indexing, iloc - integer indexing
- b. Loc - integer indexing, iloc - label indexing
- c. Loc & iloc are both same
- d. Both loc & iloc are used for boolean indexing

**Ans:a**

5. Dataframe with A,B,C columns, code to select all rows where col A is greater than col B

- a. `df[df['A'] > df['B']]`
- b. `df[df.A > df.B]`
- c. `df[(df.A - df.B) > 0]`
- d. All the above

**Ans:d**

## 6. Syntax for iloc?

- a. `Df.loc[row_index, column_index]`
- b. `Df.iloc[row_index]`
- c. `Df.iloc[:, colimn_index]`
- d. All the above

**Ans: d**

7. How to select single column from a dataframe?

- a. `Df.iloc[]`
- b. `Df.loc[]`
- c. `Df[]`
- d. `Df.idx[]`

**Ans:c**



## 8. `Dataframe.groupby()` results in?

- a. New dataframe with grouped data
- b. Dictionary with grouped data
- c. List with grouped data
- d. Tuple with grouped data

**Ans:a**

## 9. Which is true about apply function in pandas?

- a. Can be used only with series not with dataframes
- b. Can be only applied with built-in function not with user-defined functions
- c. Always returns new pandas object never modifies actual object
- d. Can be applied over both rows and columns of the dataframe

**And:d**

10. Create a Pandas Series with 5 random text strings. Convert all the strings to uppercase.

```
# Create a Pandas Series with 5 random text strings
text_data = pd.Series(['apple', 'banana', 'cherry', 'date', 'elderberry'])
print("Original Series:")
print(text_data)

# Extract the first three characters of each string
first_three_chars = text_data.str[:3]
print("First three characters:")
print(first_three_chars)
```

11. Create a Pandas DataFrame with 3 columns and 6 rows filled with random integers. Apply a lambda function to create a new column that is the sum of the existing columns.

```
# Create a Pandas DataFrame with 3 columns and 6 rows filled with random integers
df = pd.DataFrame(np.random.randint(1, 100, size=(6, 3)), columns=['A', 'B', 'C'])
print("Original DataFrame:")
print(df)

# Apply a lambda function to create a new column that is the sum of the existing columns
df['Sum'] = df.apply(lambda row: row.sum(), axis=1)
print("DataFrame with Sum column:")
print(df)
```

12. Create a Pandas DataFrame with 3 columns and 5 rows filled with random integers. Introduce some NaN values. Fill the NaN values with the mean of the respective columns.

```
# Create a Pandas DataFrame with 3 columns and 5 rows filled with random integers
df = pd.DataFrame(np.random.randint(1, 100, size=(5, 3)), columns=['A', 'B', 'C'])
print("Original DataFrame:")
print(df)

# Introduce some NaN values
df.iloc[0, 1] = np.nan
df.iloc[2, 2] = np.nan
df.iloc[4, 0] = np.nan
print("DataFrame with NaN values:")
print(df)

# Fill the NaN values with the mean of the respective columns
df.fillna(df.mean(), inplace=True)
print("DataFrame with NaN values filled:")
print(df)
```