PRELIM TERM
SAMPLE QUESTIONS

Engr. Renato R. Maaliw III, *DIT*
Professor 1, College of Engineering
Southern Luzon State University
Lucban, Quezon, Philippines

01. Which of the following libraries is primarily used for data manipulation and analysis in Python?

A. NumPy

B. Statistics

C. Pandas

D. SKLearn

01. Which of the following libraries is primarily used for data manipulation and analysis in Python?

A. NumPy

C. Pandas

B. Statistics

D. SKLearn

02. In Python, which of the following commands will display the first 5 rows of a DataFrame df?

A. df.tail(5)

B. df.head()

C. df.sample(5)

D. df.head(4)

02. In Python, which of the following commands will display the first 5 rows of a DataFrame df?

A. df.tail(5)

B. df.head()

C. df.sample(5)

D. df.head(4)

03. If you want to calculate the mean of a column age in a DataFrame, which of the following would you use?

A. df[‘age’].mean()

B. df.average(‘age’)

C. df.mean()

D. df.mean(df[‘age’])

03. If you want to calculate the mean of a column age in a DataFrame, which of the following would you use?

A. df[‘age’].mean()

B. df.average(‘age’)

C. df.mean()

D. df.mean(df[‘age’])

04. Which of the following commands would you used to check the number of missing values in a DataFrame df?

A. df.isnull()

B. df.hasna()

C. df.isnull().sum()

D. df.missing()

04. Which of the following commands would you used to check the number of missing values in a DataFrame df?

A. df.isnull()

B. df.hasna()

C. df.isnull().sum()

D. df.missing()

05. What is the output of the following code:

```
import numpy as np
data = np.array([1, 2, 3, 4, 5])
print(data[1:4])
```

A. [1, 2, 3, 4]

B. [1, 2, 3]

C. [2, 3, 4]

D. [2, 3, 4, 5]

05. What is the output of the following code:

```
import numpy as np
data = np.array([1, 2, 3, 4, 5])
print(data[1:4])
```

A. [1, 2, 3, 4]

B. [1, 2, 3]

C. [2, 3, 4]

D. [2, 3, 4, 5]

06. Which of the following commands will create a DataFrame with columns ‘A’ and ‘B’ and two rows of values [1, 2] and [3, 4]?

A. pd.DataFrame([[1, 2], [3, 4]], columns = [‘A’, ‘B’])

B. pd.DataFrame({‘A’: [1, 3], ‘B’: [2, 4]})

C. Both A and B

D. None of the above

06. Which of the following commands will create a DataFrame with columns 'A' and 'B' and two rows of values [1, 2] and [3, 4]?

A. pd.DataFrame([[1, 2], [3, 4]], columns = ['A', 'B'])

B. pd.DataFrame({'A': [1, 3], 'B': [2, 4]})

C. Both A and B

D. None of the above

07. Which of the following methods can be used to group a DataFrame df by a column 'category' and then calculate the mean of each group?

A. df.group('category').mean()

B. df.groupby('category').mean()

C. df.mean('category')

D. df.groupby.mean('category')

07. Which of the following methods can be used to group a DataFrame df by a column 'category' and then calculate the mean of each group?

A. df.group('category').mean()

B. df.groupby('category').mean()

C. df.mean('category')

D. df.groupby.mean('category')

08. What is the output of the following code?

```
import pandas as pd
data = {'A': [1, 2, None], 'B': [4, None, 6]}
df = pd.DataFrame(data)
print(df.dropna())
```

A.

```
     A    B
0  1.0  4.0
```

B.

```
     A    B
0  1.0  4.0
1  2.0  NaN
```

C.

```
     A    B
1  2.0  NaN
2  NaN  6.0
```

D.

```
     A    B
2  NaN  6.0
```

08. What is the output of the following code?

```
import pandas as pd
data = {'A': [1, 2, None], 'B': [4, None, 6]}
df = pd.DataFrame(data)
print(df.dropna())
```

A.

```
     A    B
0  1.0  4.0
```

B.

```
     A    B
0  1.0  4.0
1  2.0  NaN
```

C.

```
     A    B
1  2.0  NaN
2  NaN  6.0
```

D.

```
     A    B
2  NaN  6.0
```

09. Which of the following lines of code will produce an array containing the square of each element in the NumPy array arr = np.array([1, 2, 3, 4])??

A. arr ** 2

B. arr * arr

C. np.power(arr, 2)

D. All of the above

09. Which of the following lines of code will produce an array containing the square of each element in the NumPy array arr = np.array([1, 2, 3, 4])??

A. arr ** 2

B. arr * arr

C. np.power(arr, 2)

D. All of the above

09. Given the following DataFrame, which command would you use to select only the 'Age' column as a DataFrame rather than a Series.

```
data = {'Name': ['Alice', 'Bob', 'Charlie'], 'Age': [24, 27, 22], 'Salary': [70000, 80000, 60000]}
df = pd.DataFrame(data)
```

A. df.Age

B. df[['Age']]

C. df.loc['Age']

D. df['Age']

09. Given the following DataFrame, which command would you use to select only the ‘Age’ column as a DataFrame rather than a Series.

```
data = {'Name': ['Alice', 'Bob', 'Charlie'], 'Age': [24, 27, 22], 'Salary': [70000, 80000, 60000]}
df = pd.DataFrame(data)
```

A. df.Age

C. df.loc[‘Age’]

B. df[[‘Age’]]

D. df[‘Age’]

10. What will be the output of the following code:

```
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])
result = np.dot(a, b)
print(result)
```

A. [[4, 10, 18]]

B. 32

C. [32]

D. [[32]]

10. What will be the output of the following code:

```
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])
result = np.dot(a, b)
print(result)
```

A. [[4, 10, 18]]

C. [32]

B. 32

D. [[32]]

11. What will be the shape of the output by the following code?

```
a = np.array([[1, 2, 3], [4, 5, 6]])
b = np.array([[1, 2], [3, 4], [5, 6]])
result = np.dot(a, b)
```

A. (3, 2)

B. (2, 3)

C. (2, 2)

D. (, 2)

11. What will be the shape of the output by the following code?

```
a = np.array([[1, 2, 3], [4, 5, 6]])
b = np.array([[1, 2], [3, 4], [5, 6]])
result = np.dot(a, b)
```

A. (3, 2)

C. (2, 2)

B. (2, 3)

D. (, 2)

12. Which of the following commands would you use to merge two DataFrames, df1 and df2, on a column named 'id', keeping only the rows with matching 'id' values in both DataFrames?

A. pd.merge(df1, df2, on='id', how='outer')

B. pd.merge(df1, df2, on='id', how='left')

C. pd.merge(df1, df2, on='id', how='right')

D. pd.merge(df1, df2, on='id', how='inner')

12. Which of the following commands would you use to merge two DataFrames, df1 and df2, on a column named 'id', keeping only the rows with matching 'id' values in both DataFrames?

A. pd.merge(df1, df2, on='id', how='outer')

B. pd.merge(df1, df2, on='id', how='left')

C. pd.merge(df1, df2, on='id', how='right')

D. pd.merge(df1, df2, on='id', how='inner')

13. Given the following two DataFrames, which command will concatenate df1 and df2 along the rows (vertically)?

A. pd.concat([df1, df2], axis=0)

B. pd.concat([df1, df2], axis=1)

C. df1.concat(df2)

D. pd.concat([df1, df2], join='inner')

13. Given the following two DataFrames, which command will concatenate df1 and df2 along the rows (vertically)?

A. pd.concat([df1, df2], axis=0)

B. pd.concat([df1, df2], axis=1)

C. df1.concat(df2)

D. pd.concat([df1, df2], join='inner')

14. What will be the result of the following code?

```
import pandas as pd
df1 = pd.DataFrame({'id': [1, 2, 3], 'value1': ['A', 'B', 'C']})
df2 = pd.DataFrame({'id': [2, 3, 4], 'value2': ['D', 'E', 'F']})
result = pd.merge(df1, df2, on='id', how='outer')
print(result)
```

A.

	id	value1	value2
0	1	A	NaN
1	2	B	D
2	3	C	E
3	4	NaN	F

B.

	id	value1	value2
0	2	B	D
1	3	C	E

C.

	id	value1	value2
0	1	A	NaN
1	2	B	NaN
2	3	C	NaN
3	4	NaN	NaN

D.

	id	value1	value2
0	2	D	B
1	3	E	C
2	4	F	NaN

14. What will be the result of the following code?

```
import pandas as pd
df1 = pd.DataFrame({'id': [1, 2, 3], 'value1': ['A', 'B', 'C']})
df2 = pd.DataFrame({'id': [2, 3, 4], 'value2': ['D', 'E', 'F']})
result = pd.merge(df1, df2, on='id', how='outer')
print(result)
```

A.

	id	value1	value2
0	1	A	NaN
1	2	B	D
2	3	C	E
3	4	NaN	F

B.

	id	value1	value2
0	2	B	D
1	3	C	E

C.

	id	value1	value2
0	1	A	NaN
1	2	B	NaN
2	3	C	NaN
3	4	NaN	NaN

D.

	id	value1	value2
0	2	D	B
1	3	E	C
2	4	F	NaN

15. What does the .describe() function return by default when called on a DataFrame containing only numerical columns?

A. A summary with only the mean and standard deviation

B. The first 5 rows of the DataFrame

C. Summary statistics including count, mean, std, min, 25%, 50%, 75%, and max

D. The unique values in each column

15. What does the .describe() function return by default when called on a DataFrame containing only numerical columns?

A. A summary with only the mean and standard deviation

B. The first 5 rows of the DataFrame

C. Summary statistics including count, mean, std, min, 25%, 50%, 75%, and max

D. The unique values in each column

16. Which of the following methods could be used to fill missing values in <u>all columns</u> of df with their respective column means?

A. df = df.fillna(df.mean(), axis = 0)

B. df.fillna(mean())

C. df.fillna(df.mean(), inplace=True)

D. df.fillna(value = mean(df))

16. Which of the following methods could be used to fill missing values in <u>all columns</u> of df with their respective column means?

A. df = df.fillna(df.mean(), axis = 0)

B. df.fillna(mean())

C. df.fillna(df.mean(), inplace=True)

D. df.fillna(value = mean(df))

17. What does the thresh parameter in the dropna() function do?

A. It specifies the maximum number of NaN values allowed before dropping a row or column.

B. It removes all rows with any NaN values.

C. It specifies the minimum number of non-NaN values required to keep a row or column.

D. It replaces NaN values with a specified threshold.

17. What does the thresh parameter in the dropna() function do?

A. It specifies the maximum number of NaN values allowed before dropping a row or column.

C. It specifies the minimum number of non-NaN values required to keep a row or column.

B. It removes all rows with any NaN values.

D. It replaces NaN values with a specified threshold.

18. What does the .value_counts() function do when called on a column of a DataFrame?

A. It removes duplicates in the column.

B. It counts the number of unique values in the column.

C. It returns the number of missing values in the column.

D. It finds the mean of numeric values in the column.

18. What does the .value_counts() function do when called on a column of a DataFrame?

A. It removes duplicates in the column.

C. It returns the number of missing values in the column.

B. It counts the number of unique values in the column.

D. It finds the mean of numeric values in the column.

19. If you wanted to retain the original order of x after performing x.sort_values('Billing Amount', ascending=False), what would you do?

A. Set inplace=True in the sort_values() function.

B. Use x.reset_index() after sorting.

C. Store the result in a new variable, like sorted_x.

D. Use ascending=True instead of ascending=False.

19. If you wanted to retain the original order of x after performing x.sort_values('Billing Amount', ascending=False), what would you do?

A. Set inplace=True in the sort_values() function.

B. Use x.reset_index() after sorting.

C. Store the result in a new variable, like sorted_x.

D. Use ascending=True instead of ascending=False.

20. Consider the following code, what will df_reset look like?

```
import pandas as pd

data = {'Name': ['Alice', 'Bob', 'Charlie'],
        'Age': [24, 27, 22]}
df = pd.DataFrame(data, index=['a', 'b', 'c'])
df_reset = df.reset_index()
```

A.

	index	Name	Age
0	0	Alice	24
1	1	Bob	27
2	2	Charlie	22

B.

	Name	Age
0	Alice	24
1	Bob	27
2	Charlie	22

C.

	index	Name	Age
0	a	Alice	24
1	b	Bob	27
2	c	Charlie	22

D.

	Name	Age
a	Alice	24
b	Bob	27
c	Charlie	22

20. Consider the following code, what will df_reset look like?

```
import pandas as pd

data = {'Name': ['Alice', 'Bob', 'Charlie'],
        'Age': [24, 27, 22]}
df = pd.DataFrame(data, index=['a', 'b', 'c'])
df_reset = df.reset_index()
```

A.

	index	Name	Age
0	0	Alice	24
1	1	Bob	27
2	2	Charlie	22

B.

	Name	Age
0	Alice	24
1	Bob	27
2	Charlie	22

C.

	index	Name	Age
0	a	Alice	24
1	b	Bob	27
2	c	Charlie	22

D.

	Name	Age
a	Alice	24
b	Bob	27
c	Charlie	22

Good Luck!