Hello Shiyun Tang! You scored 7 out of 10.

Please click on the feedback to view your response.

Question 1

Feedback: This will correctly extract Alice's data as 'Mathematics' would be a column in df.T and column names can be passed as a key to retrieve the contents of the entire column, i.e. Alice's informaion in this case

Score: 1/1

Question:

Which of the following is the correct way to extract all information related to the student named `Alice` from the DataFrame `df` given below:

(Major)	Name	Age	Gender
Mathematics	Alice	20	F
Sociology	Jack	22	M

- ○ `df['Alice']`
- ○ `df['Mathematics']`
- ○ `df.iloc['Mathematics']`
- ◉ `df.T['Mathematics']`

Question 2

Correct answer:

Score: 0/1

```
(df['toefl score'] > 105) & (df['toefl score'] < 115)
```

Explanation:

This will just return a boolean mask of True's and False's instead of filtering the correct rows.

Question:

Serial No.	gre score	toefl score
1	337	118
2	324	107
3	316	104
4	322	110
5	314	103

For the given DataFrame `df` shown above, we want to get all records with a `toefl score` greater than 105 but smaller than 115. Which of the following expressions is **incorrect** to perform the same?

- (●) `df[df['toefl score'].gt(105) & df['toefl score'].lt(115)]`
- ( ) `df[(df['toefl score'].isin(range(106, 115)))]`
- ( ) `(df['toefl score'] > 105) & (df['toefl score'] < 115)`
- ( ) `df[(df['toefl score'] > 105) & (df['toefl score'] < 115)]`

Question 3

Feedback: loc and iloc are attributes of pandas.Series object, not methods.

Score: 1/1

Question:

Which of the following statements is **incorrect**?

- ○ If `s` is a `pd.Series` object, then we can use `s.loc[label]` to get all data where the index is equal to label.
- ○ We can use `s.iteritems()` on a `pd.Series` object `s` to iterate on it.
- ○ If `s` and `s1` are two pd.Series objects, we cannot use `s.append(s1)` to directly append `s1` to the existing series `s`
- ◉ `loc` and `iloc` are two useful and commonly used Pandas methods.

Question 4

Feedback: There is no index of value 1 in s2, hence this will give an error.

Score: 1/1

Question:

For the Series `s1` and `s2` defined below, which of the following statements **will give an error**?

```
import pandas as pd
s1 = pd.Series({1: 'Alice', 2: 'Jack', 3: 'Molly'})
s2 = pd.Series({'Alice': 1, 'Jack': 2, 'Molly': 3})
```

- ○ `s2.iloc[1]`
- ○ `s2[1]`
- ○ `s1.loc[1]`
- ◉ `s2.loc[1]`

Question 5

Score:

1/1

Feedback: This is an incorrect way to drop values from the column named 'two' because the axis has not been specified as 1 (representing 'columns') and the default value of axis is 0. It would yield the following error: KeyError: '['two'] not found in axis'.

Question:

Which of the following is an **incorrect** way to `drop` entries from the Pandas DataFrame named `df` shown below?

	one	two	three	four
Ohio	0	1	2	3
Colorado	4	5	6	7
Utah	8	9	10	11
New York	12	13	14	15

- ○ `df.drop('Ohio')`
- ○ `df.drop('one', axis = 1)`
- ○ `df.drop(['Utah', 'Colorado'])`
- ◉ `df.drop('two')`

Question 6

Feedback: All of these can be used to create a DataFrame in Pandas

Score: 1/1

Question:

Which of the following can be used to create a DataFrame in Pandas?

- ○ Pandas Series object

- ○ 2D ndarray
- ○ Python dict
- ◉ All of the above

Question 7

Correct answer:

`df.where(df['toefl score'] > 105)`

Score:
0/1

Explanation:

This will not work as `df.where()` will not drop any data we don't want, it will just set their values to `nan`.

Question:

Serial No.	gre score	toefl score
1	337	118
2	324	107
3	316	104
4	322	110
5	314	103

For the given DataFrame `df` we want to keep only the records with a `toefl score` greater than 105. Which of the following will **not** work?

- ◉ `df[df['toefl score'] > 105]`
- ○ `df.where(df['toefl score'] > 105)`
- ○ `df.where(df['toefl score'] > 105).dropna()`
- ○ All of these will work

Question 8

Correct answer:

Score: 0/1

```
df.rename(mapper = lambda x: x.upper(), axis = 1)
```

Explanation:

This is incorrect because the rename method will return a new DataFrame by default. We have to pass the result to our original DataFrame `df` or set the inplace parameter to 'True'.

Question:

Suppose we have a DataFrame named `df`. We want to change the original DataFrame `df` in a way that all the column names are cast to upper case. Which of the following expressions is **incorrect** to perform the same?

- ○ `df.rename(mapper = lambda x: x.upper(), axis = 1, inplace = True)`
- ◉ `df = df.rename(mapper = lambda x: x.upper(), axis = 1)`
- ○ `df.rename(mapper = lambda x: x.upper(), axis = 1)`
- ○ `df = df.rename(mapper = lambda x: x.upper(), axis = 'columns')`

Question 9

Feedback: S.iloc[i:j] can be used to retrieve Series rows from indices i to j-1

Score: 1/1

Question:

```
import pandas as pd
d = {
       '1': 'Alice',
       '2': 'Bob',
       '3': 'Rita',
       '4': 'Molly',
```

```
        '5': 'Ryan'
}
S = pd.Series(d)
```

In the above python code, the keys of the dictionary `d` represent student ranks and the value for each key is a student name. Which of the following can be used to extract rows with student ranks that are lower than or equal to 3?

- ○ `S.loc[0:3]`
- ○ `S.loc[0:2]`
- ○ `S.iloc[0:2]`
- ◉ `S.iloc[0:3]`

Question 10

Feedback: The value of obj2['California'] is nan which is not the same as None, so this will return False

Score: 1/1

Question:

For the following code, which of the following statements will **not** return True?

```
import pandas as pd

sdata = {'Ohio': 35000, 'Texas': 71000, 'Oregon': 16000, 'Utah': 5000}
obj1 = pd.Series(sdata)
states = ['California', 'Ohio', 'Oregon', 'Texas']
obj2 = pd.Series(sdata, index=states)
obj3 = pd.isnull(obj2)
```

- ○ `obj3['California']`
- ○
```
import math
math.isnan(obj2['California'])
```
- ○
```
x = obj2['California']
obj2['California'] != x
```

- `obj2['California'] == None`