

✔ Congratulations! You passed!

Grade received 100%

Latest Submission Grade 100%

To pass 80% or higher

[Go to next item](#)

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

1 / 1 point

```
1 import pandas as pd
2 sdata = {'Ohio': 35000, 'Texas': 71000, 'Oregon': 16000, 'Utah': 5000}
3 obj1 = pd.Series(sdata)
4 states = ['California', 'Ohio', 'Oregon', 'Texas']
5 obj2 = pd.Series(sdata, index=states)
6 obj3 = pd.isnull(obj2)
```

☒ 1 obj2['California'] == None

☐ 1 import math
2 math.isnan(obj2['California'])

☐ 1 obj3['California']

☐ 1 x = obj2['California']
2 obj2['California'] != x

✔ **Correct**

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

2.

```
1 import pandas as pd
2 d = {'1': 'Alice', '2': 'Bob', '3': 'Rita', '4': 'Molly', '5': 'Ryan'}
3 S = pd.Series(d)
```

1 / 1 point

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.iloc[0:2]☐ S.loc[0:3]☐ S.loc[0:2]☒ S.iloc[0:3]☒ **Correct**

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

3. 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?

1 / 1 point

☐ df = df.rename(mapper = lambda x: x.upper(), axis = 'columns')☐ df.rename(mapper = lambda x: x.upper(), axis = 1, inplace = True)☒ df.rename(mapper = lambda x: x.upper(), axis = 1)☐ df = df.rename(mapper = lambda x: x.upper(), axis = 1)☒ **Correct**

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'.

4.

1 / 1 point

	gre score	toefl score
Serial No.		
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.where(df['toefl score'] > 105).dropna()
- ☐ All of these will work
- ☒ df.where(df['toefl score'] > 105)
- ☐ df[df['toefl score'] > 105]

✓ **Correct**

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

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

1 / 1 point

- ☐ 2D ndarray
- ☐ Pandas Series object
- ☐ Python dict
- ☒ All of these work

✓ **Correct**

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

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

1 / 1 point

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

**Correct**

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`.

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

1 / 1 point

```

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

```

- ☐ `s2.iloc[1]`
- ☒ `s2.loc[1]`
- ☐ `s2[1]`
- ☐ `s1.loc[1]`

**Correct**

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

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

1 / 1 point

- ☐ We can use `s.iteritems()` on a `pd.Series` object `s` to iterate on it.
- ☒ `loc` and `iloc` are two useful and commonly used Pandas methods.
- ☐ 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.
- ☐ If `s` and `s1` are two `pd.Series` objects, we cannot use `s.append(s1)` to directly append `s1` to the existing series `s`
- ☒ **Correct**
`loc` and `iloc` are attributes of pandas. Series object, not methods.

9.

1 / 1 point

	gre score	toefl score
Serial No.		
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['toefl score'] > 105) & (df['toefl score'] < 115)`
- ☐ `df[(df['toefl score'].isin(range(106, 115)))]`
- ☐ `df[(df['toefl score'] > 105) & (df['toefl score'] < 115)]`
- ☒ **Correct**
 This will just return a boolean mask of True's and False's instead of filtering the correct rows.

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

1 / 1 point

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

☒ `df.T['Mathematics']`

☐ `df['Mathematics']`

☐ `df['Alice']`

☐ `df.iloc['Mathematics']`

☒ **Correct**

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 information in this case