1分

```
1 import numpy as np
2
3 a = np.arange(8)
4 b = a[4:6]
5 b[:] = 40
6 c = a[4] + a[6]
```

2. Given the string **s** as shown below, which of the following expressions will be **True**?

•

1 import re 2 s = 'ABCAC'

```
1 bool(re.match('A', s)) == True
```

3. Consider a string s. We want to find all characters (other than A) which are followed by triple A, i.e., have AAA to the right. 1分 We don't want to include the triple A in the output and just want the character immediately preceding AAA. Complete the code given below that would output the required result.

```
1 v def result():
2 s = 'ACAABAACAAABACDBADDDFSDDDFFSSSASDAFAAACBAAAFASD'
3
4 result = []
5 # compete the pattern below
6 pattern = '([A-Z]*?)(A{3})'
7 v for item in re.finditer(pattern, s):
8 # identify the group number below.
9 result.append(item.group(1))
10
11 return result
12 result()

['ACAABAAC', 'BACDBADDDFSDDDFFSSSASDAF', 'CB']
```

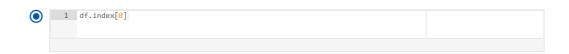
```
def result():
    s = 'ACAABAACAAABACDBADDDFSDDDFFSSSASDAFAAACBAAAFASD'

    result = []
    # compete the pattern below
    pattern = '([A-Z]*?)(A{3})'
    for item in re.finditer(pattern, s):
        # identify the group number below.
        result.append(item.group(1))
```

4.

>>> df d 4 b 7 a -5 c 3 dtype: int64

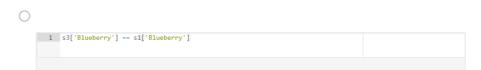
Consider the following 4 expressions regarding the above pandas Series **df**. All of them have the same value except one expression. Can you identify which one it is?



5. 1分

| | | >>> s2 | |
|--------------|----|--------------|----|
| >>> s1 | | Strawberry | 20 |
| Mango | 20 | Vanilla | 30 |
| Strawberry | 15 | Banana | 15 |
| Blueberry | 18 | Mango | 20 |
| Vanilla | 31 | Plain | 20 |
| dtype: int64 | | dtype: int64 | |

Consider the two pandas Series objects shown above, representing the no. of items of different yogurt flavors that were sold in a day from two different stores, **s1 and s2**. Which of the following statements is **True** regarding the Series **s3** defined below?





1 s3['Blueberry'] == s1.add(s2, fill_value = 0)['Blueberry']

| 6. | In the following list of statements regarding a DataFrame df, one or more statements are correct. Can you identify all correct statements? | the 1分 |
|----|--|--------|
| | Every time we call df.set_index(), the old index will be discarded. | |
| | Every time we call df.set_index(), the old index will be set as a new column. | |
| | Every time we call df.reset_index(), the old index will be discarded. | |
| | Every time we call df.reset_index(), the old index will be set as a new column. | |
| 7. | Consider the Series object S defined below. Which of the following is an incorrect way to slice S such that we obtain all data points corresponding to the indices 'b', 'c', and 'd'? | 1分 |
| | 1 S = pd.Series(np.arange(5), index=['a', 'b', 'c', 'd', 'e']) | |
| | | |
| | | |
| | 1 S['b':'e'] | |
| | | |
| | | |
| | \circ | |
| | 1 S[['b', 'c', 'd']] | |
| | | |
| 8. | | 1分 |
| | >>> df a b c | |
| | R1 5 6 20 R2 5 82 28 | |
| | R3 71 31 92 R4 67 37 49 | |
| | | |
| | Consider the DataFrame df shown above with indexes 'R1', 'R2', 'R3', and 'R4'. In the following code, a new DataFrame df_new is created using df. What will be the value of df_new[1] after the below code is executed? | |
| | | |
| | <pre>1 f = lambda x: x.max() + x.min() 2 df_new = df.apply(f)</pre> | |
| | | |
| | | |
| | 88 | |
| | | |

9. 1分

| | mean | | | | | amax | | | | |
|------------|-----------------------------|------------------------|------------------------------|-----------------------------|-----------|-----------------------------|------------------------|------------------------------|-----------------------------|-------|
| Rank_Level | First Tier Top Unversity | Other Top Unversity | Second Tier Top Unversity | Third Tier Top Unversity | All | First Tier Top Unversity | Other Top Unversity | Second Tier Top Unversity | Third Tier Top Unversity | All |
| country | | | | | | | | | | |
| Argentina | NaN | 44.672857 | NaN | NaN | 44.672857 | NaN | 45.66 | NaN | NaN | 45.66 |
| Australia | 47.9425 | 44.645750 | 49.2425 | 47.285000 | 45.825517 | 51.61 | 45.97 | 50.40 | 47.47 | 51.61 |
| Austria | NaN | 44.864286 | NaN | 47.066667 | 45.139583 | NaN | 46.29 | NaN | 47.78 | 47.78 |
| Belgium | 51.8750 | 45.081000 | 49.0840 | 46.746667 | 47.011000 | 52.03 | 46.21 | 49.73 | 47.14 | 52.03 |
| Brazil | NaN | 44.499706 | 49.5650 | NaN | 44.781111 | NaN | 46.08 | 49.82 | NaN | 49.82 |

Consider the DataFrame named new_df shown above. Which of the following expressions will output the result (showing the head of a DataFrame) below?

| | country | Argentina | Australia | Austria | Belgium | Brazil | Bulgaria | Canada | Chile | China | Colombia | Switzerland | Taiwan | Thai |
|------|---------------------------------|-----------|-----------|-----------|-----------|-----------|----------|-----------|---------|-----------|----------|-----------------|-----------|------|
| | Rank_Level | | | | | | | | | | | | | |
| mean | First Tier Top Unversity | NaN | 47.942500 | NaN | 51.875000 | NaN | NaN | 53.633846 | NaN | 53.592500 | NaN | 54.005000 | 54.210000 | |
| | Other Top Unversity | 44.672857 | 44.645750 | 44.864286 | 45.081000 | 44.499706 | 44.335 | 44.760541 | 44.7675 | 44.564267 | 44.4325 | 44.625000 | 44.476667 | 44.8 |
| | Second Tier Top Unversity | NaN | 49.242500 | NaN | 49.084000 | 49.565000 | NaN | 49.218182 | NaN | 47.868000 | NaN | 48.184000 | NaN | |
| | Third Tier Top Unversity | NaN | 47.285000 | 47.066667 | 46.746667 | NaN | NaN | 46.826364 | NaN | 46.926250 | NaN | 47.930000 | 47.065000 | 46.5 |
| | All | 44.672857 | 45.825517 | 45.139583 | 47.011000 | 44.781111 | 44.335 | 47.359306 | 44.7675 | 44.992575 | 44.4325 | 51.208846 | 45.012391 | 45.1 |

new_df.stack()

10.

| | Item | Store | Quantity sold |
|---|--------|-------|---------------|
| 0 | item_1 | Α | 10.0 |
| 1 | item_1 | В | 20.0 |
| 2 | item_1 | С | NaN |
| 3 | item_2 | Α | 5.0 |
| 4 | item_2 | В | 10.0 |
| 5 | item_2 | С | 15.0 |

Consider the DataFrame \mathbf{df} shown above. What will be the output (rounded to the nearest integer) when the following code related to \mathbf{df} is executed:

1 df.groupby('Item').sum().iloc[0]['Quantity sold']

30.0