Week 2_Quiz #2 (Working)

March 7, 2021

1 Week 2 Quiz 2 Working

[1]:
```
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)
```

1.1 Q1

x = obj2['California'], obj2['California']!=x is correct, as two objects with the value of NaN are not equal to each other according to Python. NaN != None hence the 2nd statement is true.

Obj3 is a boolean mask of all the elements in obj2. Hence, obj3 would return true for obj2['California'] is Nan Last option is true.

[2]:
```
obj2['California']  == None
```

[2]:
```
False
```

1.2 Q2

[3]:
```
import pandas as pd
d = {
    '1': 'Alice',
    '2': 'Bob',
    '3': 'Rita',
    '4': 'Molly',
    '5': 'Ryan'
}
S = pd.Series(d)
S.iloc[0:3]
```

[3]:
```
1    Alice
2      Bob
3     Rita
```

```
dtype: object
```

1.3 Q3

df.rename(mapper = lambda x: x.upper(), axis = 1)
This line creates an alias of the dataframe, and it does not change to original dataframe.

1.4 Q4

df.where(df['toefl score']) > 105) will not work
without the .dropna() clause, as all the NaN values will still be present.

1.5 Q5

A 2D array, a panda series object and a python dictionary can all be used to create a DataFrame in Pandas.

1.6 Q6

df.drop('two') will most likely give an error. This is because an extra parameter axis is needed to specify a column.

1.7 Q7

[4]:
```
import pandas as pd
s1 = pd.Series({1: 'Alice', 2: 'Jack', 3: 'Molly'})
s2 = pd.Series({'Alice': 1, 'Jack': 2, 'Molly': 3})
```

[5]:
```
try:
    s2.loc[1]
except:
    print("TypeError")
```

```
TypeError
```

1.8 Q8

loc and iloc are attributes and NOT methods.

You are lucky: take note that the rest is true: - If s is a pd.Series object, then we can use s.loc[label] to get all the data where the index is equal to the 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 CAN use s.append(s1) but this is to create a new series with s1 appended, but not to directly append s1 to the existing series s (and store it in s).

1.9 Q9

(df['toefl score'] > 105) & (df ['toefl score'] < 115) returns a boolean and not the records.

1.10 Q10

[6]:
```
import pandas as pd
record1 = pd.Series({
    'Name': 'Alice',
    'Age': '20',
    'Gender': 'F'
})
record2 = pd.Series({
    'Name': 'Jack',
    'Age': '22',
    'Gender': 'M'
})
df = pd.DataFrame([record1,record2], index = ['Mathematics','Sociology'])
df
```

[6]:

	Name	Age	Gender
Mathematics	Alice	20	F
Sociology	Jack	22	M

[7]:
```
df.iloc['Mathematics'] # Cannot index by location index with non-integer key.
```

```
␣
↪---------------------------------------------------------------------------

TypeError                                 Traceback (most recent call␣
↪last)

<ipython-input-7-d656bae4694e> in <module>
----> 1 df.iloc['Mathematics'] # Cannot index by location index with␣
↪non-integer key.


/opt/conda/lib/python3.7/site-packages/pandas/core/indexing.py in␣
↪__getitem__(self, key)
    1408
    1409             maybe_callable = com.apply_if_callable(key, self.obj)
-> 1410             return self._getitem_axis(maybe_callable, axis=axis)
    1411
    1412     def _is_scalar_access(self, key: Tuple):


/opt/conda/lib/python3.7/site-packages/pandas/core/indexing.py in␣
↪_getitem_axis(self, key, axis)
    2127             key = item_from_zerodim(key)
    2128             if not is_integer(key):
```

```
   -> 2129                 raise TypeError("Cannot index by location index with␣
↪a non-integer key")
      2130
      2131             # validate the location


       TypeError: Cannot index by location index with a non-integer key
```

[8]:
```
df['Alice'] # This is an error because Alice is neither the label or the index.
```

```
      ␣
↪---------------------------------------------------------------------------

       KeyError                                  Traceback (most recent call␣
↪last)

       /opt/conda/lib/python3.7/site-packages/pandas/core/indexes/base.py in␣
↪get_loc(self, key, method, tolerance)
      2889             try:
   -> 2890                 return self._engine.get_loc(key)
      2891             except KeyError:


       pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()


       pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()


       pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.
↪PyObjectHashTable.get_item()


       pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.
↪PyObjectHashTable.get_item()


       KeyError: 'Alice'


   During handling of the above exception, another exception occurred:


       KeyError                                  Traceback (most recent call␣
↪last)
```

```
      <ipython-input-8-2a0af6909fec> in <module>
   ----> 1 df['Alice'] # This is an error because Alice is neither the label or␣
↪the index.


      /opt/conda/lib/python3.7/site-packages/pandas/core/frame.py in␣
↪__getitem__(self, key)
      2973             if self.columns.nlevels > 1:
      2974                 return self._getitem_multilevel(key)
   -> 2975             indexer = self.columns.get_loc(key)
      2976             if is_integer(indexer):
      2977                 indexer = [indexer]


      /opt/conda/lib/python3.7/site-packages/pandas/core/indexes/base.py in␣
↪get_loc(self, key, method, tolerance)
      2890                 return self._engine.get_loc(key)
      2891             except KeyError:
   -> 2892                 return self._engine.get_loc(self.
↪_maybe_cast_indexer(key))
      2893         indexer = self.get_indexer([key], method=method,␣
↪tolerance=tolerance)
      2894         if indexer.ndim > 1 or indexer.size > 1:


      pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()


      pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()


      pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.
↪PyObjectHashTable.get_item()


      pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.
↪PyObjectHashTable.get_item()


      KeyError: 'Alice'
```

[9]:
```
print(df.T)
df.T['Mathematics'] #this indexing works because now Mathematics is the header␣
 ↪of the column. Probably need to revise more of this.
```

```
       Mathematics Sociology
Name         Alice      Jack
Age             20        22
Gender           F         M
```

[9]:
```
Name      Alice
Age          20
Gender        F
Name: Mathematics, dtype: object
```

[10]:
```
df.rename(mapper = lambda x: x.upper(), axis = 1)
```

[10]:
```
              NAME AGE GENDER
Mathematics  Alice  20      F
Sociology     Jack  22      M
```

[11]:
```
df
```

[11]:
```
              Name Age Gender
Mathematics  Alice  20      F
Sociology     Jack  22      M
```

[14]:
```
df.where(df['Name'] == 'Alice').dropna()
```

[14]:
```
              Name Age Gender
Mathematics  Alice  20      F
```

[15]:
```
df
```

[15]:
```
              Name Age Gender
Mathematics  Alice  20      F
Sociology     Jack  22      M
```

[17]:
```
df.drop('Age',axis = 1)
```

[17]:
```
              Name Gender
Mathematics  Alice      F
Sociology     Jack      M
```

[ ]:
```
```