Dealing with missing data

A. Identifying missing values in tabular data

```
import pandas as pd
from io import StringIO
import sys

csv_data = \
'''A,B,C,D
1.0,2.0,3.0,4.0
5.0,6.0,,8.0
10.0,11.0,12.0,'''

# If you are using Python 2.7, you need
# to convert the string to unicode:

if (sys.version_info < (3, 0)):
    csv_data = unicode(csv_data)
```

Step 1: Read the csv file as a pandas dataframe

```
csv_data
```

```
'A,B,C,D\n1.0,2.0,3.0,4.0\n5.0,6.0,,8.0\n10.0,11.0,12.0,'
```

```
StringIO(csv_data)
```

```
<_io.StringIO at 0x7d02c3393b50>
```

```
mydata = pd.read_csv(StringIO(csv_data))
```

```
mydata
```

	A	B	C	D
0	1.0	2.0	3.0	4.0
1	5.0	6.0	NaN	8.0
2	10.0	11.0	12.0	NaN

Step 2: Check the number of missing values for the columns

```
mydata.isnull().sum()
```

```
A    0
B    0
C    1
D    1
dtype: int64
```

Step 3: access the underlying NumPy array via the `values` attribute

```
import numpy as np
df = pd.DataFrame(mydata)

# Access the underlying NumPy array via the values attribute
array = df.values

print(array)
```

```
[[ 1.  2.  3.  4.]
 [ 5.  6. nan  8.]
 [10. 11. 12. nan]]
```

Step 4: Remove rows from df that contain missing values

```
df_cleaned_row = df.dropna(axis=0)
print(df_cleaned_row)
```

```
     A    B    C    D
0  1.0  2.0  3.0  4.0
```

Step 5: Remove columns from df that contain missing values

```
df_cleaned_col = df.dropna(axis=1)
print(df_cleaned_col)
```

```
     A    B    C    D
0  1.0  2.0  3.0  4.0
```

Step 6: Only drop rows where all columns are NaN

```
df_cleaned_row_col = df.dropna(how='all')

print(df_cleaned_row_col)
```

```
     A    B    C    D
0  1.0  2.0  3.0  4.0
```

Step 7: Drop rows that have less than 3 real values

```
df_cleaned_real_values = df.dropna(thresh=3)

print(df_cleaned_real_values)
```

```
     A    B    C    D
0  1.0  2.0  3.0  4.0
```

Step 8: Only drop rows where NaN appear in specific columns (here: 'C')

```
# Drop rows where NaN appears in the 'C' column
df_cleaned_specolumn = df.dropna(subset=['C'])

print(df_cleaned_specolumn)
```

```
      A     B     C    D
0   1.0   2.0   3.0  4.0
2  10.0  11.0  12.0  NaN
```

B. Imputing missing values

```
# again: our original array
df.values
```

```
array([[ 1.,  2.,  3.,  4.],
       [ 5.,  6., nan,  8.],
       [10., 11., 12., nan]])
```

Step 1: impute missing values via the column mean

```
from sklearn.impute import SimpleImputer
```

```
import numpy as np
```

```
from sklearn.impute import SimpleImputer
import numpy as np

# Create your original array with missing values
original_array = np.array([
    [1.0, 2.0, 3.0, 4.0],
    [5.0, 6.0, np.nan, 8.0],
    [10.0, 11.0, 12.0, np.nan]
])

# Initialize the SimpleImputer with 'mean' strategy
imputer = SimpleImputer(strategy='mean')

# Fit the imputer to the data and transform the array
imputed_array = imputer.fit_transform(original_array)

print("Original Array:")
print(original_array)
print("\nImputed Array:")
print(imputed_array)
```

```
Original Array:
[[ 1.  2.  3.  4.]
 [ 5.  6. nan  8.]
 [10. 11. 12. nan]]

Imputed Array:
[[ 1.   2.   3.   4. ]
 [ 5.   6.   7.5  8. ]
 [10.  11.  12.   6. ]]
```