Poston
Harsy
Custard
Schlottman

Sean Poston, Ryan Custard, Timothy Schlottman, Jonathan Harsy

1.What are the two main Pandas data structures? Use examples to explain how to create them

The two main data structures in panda are series and dataframes.

Series would be initialized like this: pd.Series([7, 'Heisenberg', 3.14, -1789710578, 'Happy Eating!'])

and would display like a 1-dimensional array that goes from 0 – N where N is the length -1.

DataFrames would be initialized like this: data = {'year': [2010, 2011, 2012, 2011, 2012, 2010, 2011, 2012],

'team': ['Bears', 'Bears', 'Bears', 'Packers', 'Packers', 'Lions', 'Lions', 'Lions'],

'wins': [11, 8, 10, 15, 11, 6, 10, 4],

'losses': [5, 8, 6, 1, 5, 10, 6, 12]}

where each bracket are separated into columns and can be exported into csv files.

2. Use an example to explain the difference between Pandas DataFrame and Numpy ndarray

Numpy arrays must be made with their shape and size like this “np.ndarray(shape=(2,2), dtype=float, order='F')” while DataFrames can just be filled without needing to be initialized first.

3.Use an example to explain the various ways to do indexing and slicing with DataFrame with brackets, .loc, .iloc, .at, .iat. Please also explain the differences among them

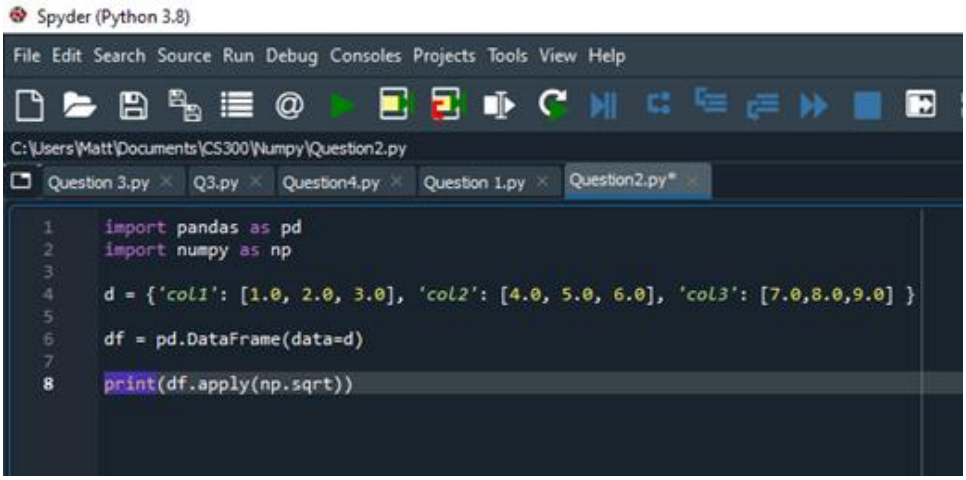
Poston
Harsy
Custard
Schlottman

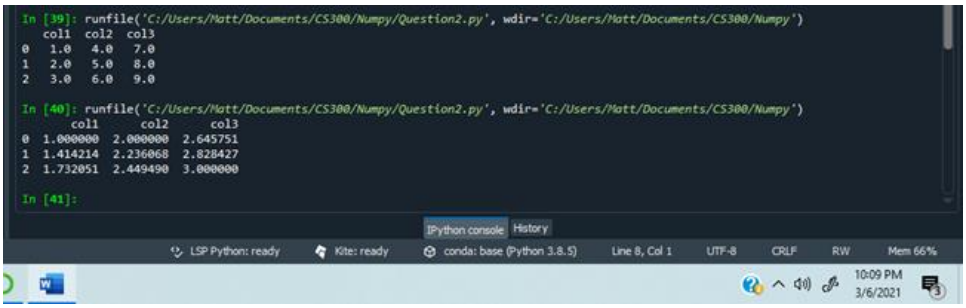
```python
d = {'col1': [1, 2, 3], 'col2': [4, 5, 6], 'col3': [7,8,9] }

df = pd.DataFrame(data=d, index = ["row1", 'row2', 'row3'])

#Here is the most basic selecting using .loc
print(df.loc['row1', 'col2'])
#Here is how indexing is done with .loc
print(df.loc['row1':'row2','col1':'col2'])
#Here is the .iloc basic selecting the 1st row second column
print(df.iloc[0,1])
#Here is the slicing with iloc
print(df.iloc[1:3, 0:3])
#Here is the basic selecting with .at
print(df.at['row1', 'col2'])
#.at and .iat does not allow for slicing
print(df.iat[0,1])
```

```
In [33]: runfile('C:/Users/Matt/Docume
4
      col1  col2
row1     1     4
row2     2     5
4
      col1  col2  col3
row2     2     5     8
row3     3     6     9
4
4

In [34]:
```

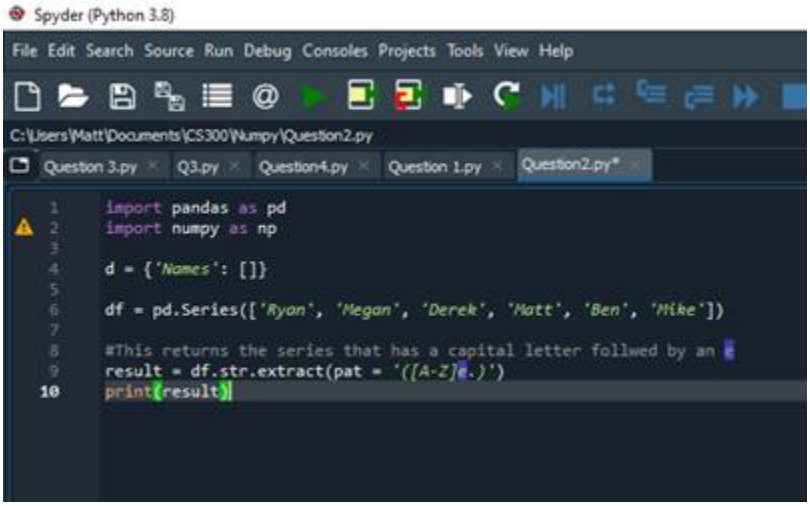
4. Use an example to explain how to apply functions to DataFrame.

Poston
Harsy
Custard
Schlottman

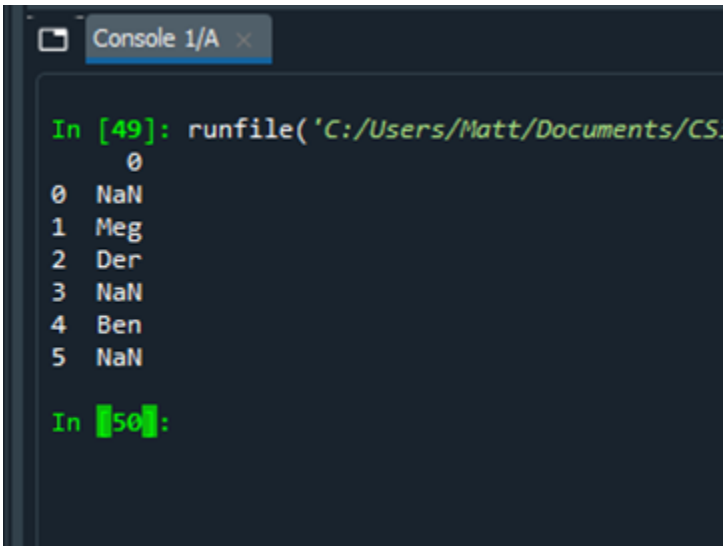




5. Use an example to explainhow .str.extract( ) works with Pandas series



Poston
Harsy
Custard
Schlottman



6. Use an example to explainhow DataFrame.concat()work in the following cases: Taking the union of them all, join='outer'. This is the default option as it results in zero information loss. Taking the intersection, join='inner'. Use a specific index, as passed to the join_axes argument.

For a given array;

First, the default `join='outer'` behavior:

In [8]: df4 = pd.DataFrame({'B': ['B2', 'B3', 'B6', 'B7'], ...: 'D': ['D2', 'D3', 'D6', 'D7'], ...: 'F': ['F2', 'F3', 'F6', 'F7']}, ...: index=[2, 3, 6, 7]) ...: In [9]: result = pd.concat([df1, df4], axis=1, sort=False)

2)

with join='inner':

In [10]: result = pd.concat([df1, df4], axis=1, join='inner')

7. Use an example to explain how DataFrame.merge( ) work in various cases

Able to inner, outer, left and right join.

Left example:

```
import pandas as pd
left_dataframe  = pd.DataFrame({'key':['Key_0','Key_1','Key_4','Key_7'],
'B':[145,2373,415,2946]})
right_dataframe  = pd.DataFrame({'key': ['Key_0', 'Key_1', 'Key_2', 'Key_3', 'Key_4',
```

Poston
Harsy
Custard
Schlottman

```
'Key_5'],
'A': ['113', '2342', '4567', '2563', '2234', '71218'],
'B': ['991.03', '993.13', '983.12', '936.45', '995.44', '999.99']})
print(" THE LEFT DATAFRAME ")
print(left_dataframe )
print("")
print(" THE RIGHT DATAFRAME ")
print(right_dataframe )
print("")
print(" LEFT JOIN ")
print(pd.merge(left_dataframe ,right_dataframe ,on=['key','key'],how='left'))
```

8. Use an example to explain how DataFrame.join( ) work in various cases

```
import pandas as pd
info1 = pd.DataFrame({'Reg_no': ['11', '12', '13', '14', '15', '16'],
'Result1': ['77', '79', '96', '38', '54', '69']})
print(info1)
info2 = pd.DataFrame({'Reg_no': ['11', '12', '13'],
'Result2': ['72', '82', '92']})
print(info2)\
final_info = info1.join(info2.set_index('Reg_no'), on="Reg_no")
print(final_info)
```

9. Use an example to explain how DataFrame.groupby( ) work

This example groups the two ‘Fries’ keys and takes the average of their values to output 140.

Poston
Harsy
Custard
Schlottman

```python
import pandas as pd
food = {'Food': ['Spaghetti', 'Tacos',
                 'Spinach', 'Fries', 'Fries'],
        'Calories': [200, 150, 40, 150, 130]}

frame = pd.DataFrame(food)
print(frame)

x = frame.groupby(['Food']).mean()
print(x)
```

```
       Food  Calories
0  Spaghetti       200
1     Tacos       150
2   Spinach        40
3    Fries       150
4    Fries       130
           Calories
Food
Fries           140
Spaghetti       200
Spinach          40
Tacos           150
```

10. Use an example to explain how DataFrame stacking and unstacking work with hierarchical index.

Stack and unstack will pivot the columns to add or remove to the index axis. stack() will add, unstack() will remove.

```python
import pandas as pd
food = {'Food': ['Spaghetti', 'Tacos',
                 'Spinach', 'Fries', 'Fries'],
        'Calories': [200, 150, 40, 150, 130]}

frame = pd.DataFrame(food)
print(frame)
print()
print(frame.stack())
print()
print(frame.unstack())
```

Poston
Harsy
Custard
Schlottman

```
        Food  Calories
0  Spaghetti       200
1      Tacos       150
2    Spinach        40
3     Fries       150
4     Fries       130

0  Food        Spaghetti
   Calories          200
1  Food            Tacos
   Calories          150
2  Food          Spinach
   Calories           40
3  Food            Fries
   Calories          150
4  Food            Fries
   Calories          130
dtype: object

Food      0    Spaghetti
          1        Tacos
          2      Spinach
          3       Fries
          4       Fries
Calories  0         200
          1         150
          2          40
          3         150
          4         130
```

11. Use an example to explain how DataFrame pivot_table( ) works.

This pivot table consists of the calories of a food. It's indexed by the type of food and split into columns of the number of servings.

```
import pandas as pd
food = {'Food': ['Spaghetti', 'Tacos',
                 'Spinach', 'Fries', 'Fries'],
        'Servings': [1, 1, 2, 1, 1],
        'Calories': [200, 150, 40, 150, 130]}

frame = pd.DataFrame(food)
print(frame)
print()
print(pd.pivot_table(frame, values='Calories', index=['Food'],
columns='Servings'))
```

Poston
Harsy
Custard
Schlottman

```
        Food  Servings  Calories
0  Spaghetti         1       200
1      Tacos         1       150
2    Spinach         2        40
3      Fries         1       150
4      Fries         1       130

Servings       1     2
Food
Fries      140.0   NaN
Spaghetti  200.0   NaN
Spinach      NaN  40.0
Tacos      150.0   NaN
```

12. Use an example to explain how categorical data in a DataFrame works.

In this example, the pd.cut() function is used to sort data into bins and turns it into a categorical variable.

```
import pandas as pd
import numpy as np
frame = pd.DataFrame({'value' : np.random.randint(0, 100, 20)})

labels = ['{0} - {1}'.format(i, i + 9) for i in range(0, 100, 10)]

frame['group'] = pd.cut(frame.value, range(0, 105, 10), labels=labels)

print(frame.tail(5))
```

```
    value    group
15     12  10 - 19
16     58  50 - 59
17     33  30 - 39
18      6    0 - 9
19     69  60 - 69
```

If we print frame.dtype, you can see that it’s a catergory

```
value       int32
group    category
dtype: object
```