Consider the following Python dictionary `data` and Python list `labels`:

```
data = {'animal': ['cat', 'cat', 'snake', 'dog', 'dog', 'cat', 'snake',
'cat', 'dog', 'dog'],
        'age': [2.5, 3, 0.5, np.nan, 5, 2, 4.5, np.nan, 7, 3],
        'visits': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'priority': ['yes', 'yes', 'no', 'yes', 'no', 'no', 'no',
'yes', 'no', 'no']}

labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
```

1. Create a DataFrame `df` from this dictionary `data` which has the index `labels`.

In [219…

```
import pandas as pd
import numpy as np
data = {'animal': ['cat', 'cat', 'snake', 'dog', 'dog', 'cat', 'snake', 'cat', '
        'age': [2.5, 3, 0.5, np.nan, 5, 2, 4.5, np.nan, 7, 3],
        'visits': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'priority': ['yes', 'yes', 'no', 'yes', 'no', 'no', 'no', 'yes', 'no', '

labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
df=pd.DataFrame(data,\
                  columns = ['animal','age','visits','priority'],
                  index = ['a','b','c','d','e','f','g','h','i','j'] )
df
```

Out[219…

	animal	age	visits	priority
a	cat	2.5	1	yes
b	cat	3.0	3	yes
c	snake	0.5	2	no
d	dog	NaN	3	yes
e	dog	5.0	2	no
f	cat	2.0	3	no
g	snake	4.5	1	no
h	cat	NaN	1	yes
i	dog	7.0	2	no
j	dog	3.0	1	no

2. Display a summary of the basic information about this DataFrame and its data (*hint: there is a single method that can be called on the DataFrame*).

In [221…

```
df.describe()
```

Out[221…

	age	visits
count	8.000000	10.000000
mean	3.437500	1.900000
std	2.007797	0.875595
min	0.500000	1.000000
25%	2.375000	1.000000
50%	3.000000	2.000000
75%	4.625000	2.750000
max	7.000000	3.000000

3. Return the first 3 rows of the DataFrame `df`.

In [223…

```
df.head(3)
```

Out[223…

	animal	age	visits	priority
a	cat	2.5	1	yes
b	cat	3.0	3	yes
c	snake	0.5	2	no

4. Display the 'animal' and 'age' columns from the DataFrame `df`

In [225…

```
df[["animal", "age"]]
```

Out[225…

	animal	age
a	cat	2.5
b	cat	3.0
c	snake	0.5
d	dog	NaN
e	dog	5.0
f	cat	2.0
g	snake	4.5
h	cat	NaN
i	dog	7.0
j	dog	3.0

5. Display the data in rows `[3, 4, 8]` *and* in columns `['animal', 'age']'

In [227…

```
df.iloc[[3,4,8],[0,1]]
```

Out[227...

	animal	age
d	dog	NaN
e	dog	5.0
i	dog	7.0

6. Select only the rows where the number of visits is greater than 3.

In [229...

```
df[df["visits"]>3]
```

Out[229...

animal	age	visits	priority

In [231...

```
df[df["visits"]>=3]
```

Out[231...

	animal	age	visits	priority
b	cat	3.0	3	yes
d	dog	NaN	3	yes
f	cat	2.0	3	no

7. Select the rows where the age is missing, i.e. it is `NaN`.

In [233...

```
null=pd.isna(data1)
null
df[null["age"]==True]
```

Out[233...

	animal	age	visits	priority
d	dog	NaN	3	yes
h	cat	NaN	1	yes

In [138...

```
null=pd.isna(data1)
null
null[null["age"]==True]
```

Out[138...

	animal	age	visits	priority
d	False	True	False	False
h	False	True	False	False

8. Select the rows where the animal is a cat *and* the age is less than 3.

In [235...

```
df[(df["animal"]=="cat") & (df["age"]<3)]
```

Out[235…

	animal	age	visits	priority
a	cat	2.5	1	yes
f	cat	2.0	3	no

9. Select the rows where the age is between 2 and 4 (inclusive)

In [237…

```
age=df[(df["age"]>=2)&(df["age"]<=4)]
age
```

Out[237…

	animal	age	visits	priority
a	cat	2.5	1	yes
b	cat	3.0	3	yes
f	cat	2.0	3	no
j	dog	3.0	1	no

10. Change the age in row 'f' to 1.5.

In [239…

```
df.loc[df.age==2.0, 'age'] = 1.5
df
```

Out[239…

	animal	age	visits	priority
a	cat	2.5	1	yes
b	cat	3.0	3	yes
c	snake	0.5	2	no
d	dog	NaN	3	yes
e	dog	5.0	2	no
f	cat	1.5	3	no
g	snake	4.5	1	no
h	cat	NaN	1	yes
i	dog	7.0	2	no
j	dog	3.0	1	no

11. Calculate the sum of all visits in `df` (i.e. the total number of visits).

In [241…

```
df["visits"].sum()
```

Out[241…

```
19
```

12. Calculate the mean age for each different animal in `df` .

In [247…

```
cat=df[(df["animal"]=="cat")]
age_of_cat=cat["age"]
```

```
age_of_cat.mean()
```

Out[247…

```
2.3333333333333335
```

In [245…

```
dog=df[(df["animal"]=="dog")]
age_of_dog=dog["age"]
print(age_of_dog.mean())

snake=df[(df["animal"]=="snake")]
age_of_snake=snake["age"]
print(age_of_snake.mean())
```

```
5.0
2.5
```

13. Append a new row 'k' to `df` with your choice of values for each column. Then delete that row to return the original DataFrame.

In [249…

```
df.loc['k'] = ["raccoon", 2, 3, "yes"]
print(df)

print("\nDeleting new row")
df=df.drop("k")
print(df)
```

```
    animal  age  visits priority
a      cat  2.5       1      yes
b      cat  3.0       3      yes
c    snake  0.5       2       no
d      dog  NaN       3      yes
e      dog  5.0       2       no
f      cat  1.5       3       no
g    snake  4.5       1       no
h      cat  NaN       1      yes
i      dog  7.0       2       no
j      dog  3.0       1       no
k  raccoon  2.0       3      yes

Deleting new row
  animal  age  visits priority
a    cat  2.5       1      yes
b    cat  3.0       3      yes
c  snake  0.5       2       no
d    dog  NaN       3      yes
e    dog  5.0       2       no
f    cat  1.5       3       no
g  snake  4.5       1       no
h    cat  NaN       1      yes
i    dog  7.0       2       no
j    dog  3.0       1       no
```

14. Count the number of each type of animal in `df`.

In [251…

```
df['animal'].value_counts()
```

Out[251…

```
animal
cat      4
dog      4
snake    2
Name: count, dtype: int64
```

15. Sort `df` first by the values in the 'age' in *decending* order, then by the value in the 'visits' column in *ascending* order (so row `i` should be first, and row `d` should be last).

In [259…

```
# age_desc=df.sort_values(by=["age"], ascending=False)
# age_desc

age_visit=df.sort_values(['age', 'visits'], ascending=[False, True])
age_visit
```

Out[259…

	animal	age	visits	priority
i	dog	7.0	2	no
e	dog	5.0	2	no
g	snake	4.5	1	no
j	dog	3.0	1	no
b	cat	3.0	3	yes
a	cat	2.5	1	yes
f	cat	1.5	3	no
c	snake	0.5	2	no
h	cat	NaN	1	yes
d	dog	NaN	3	yes

16. The 'priority' column contains the values 'yes' and 'no'. Replace this column with a column of boolean values: 'yes' should be `True` and 'no' should be `False`.

In [263…

```
df.loc[df.priority=='yes', 'priority'] = True
df.loc[df.priority=='no', 'priority'] = False
df
```

Out[263…

	animal	age	visits	priority
a	cat	2.5	1	True
b	cat	3.0	3	True
c	snake	0.5	2	False
d	dog	NaN	3	True
e	dog	5.0	2	False
f	cat	1.5	3	False
g	snake	4.5	1	False
h	cat	NaN	1	True
i	dog	7.0	2	False
j	dog	3.0	1	False

17. In the 'animal' column, change the 'snake' entries to 'python'.

In [265…

```
df.loc[df.animal=='snake', 'animal'] = 'python'
df
```

Out[265…

	animal	age	visits	priority
a	cat	2.5	1	True
b	cat	3.0	3	True
c	python	0.5	2	False
d	dog	NaN	3	True
e	dog	5.0	2	False
f	cat	1.5	3	False
g	python	4.5	1	False
h	cat	NaN	1	True
i	dog	7.0	2	False
j	dog	3.0	1	False

18. Load the ny-flights dataset to Python

In [267…

```
flight=pd.read_csv("C:\\Users\\Gouri\\Downloads\\ny-flights.csv")
flight
```

Out[267…

	fl_date	unique_carrier	airline_id	tail_num	fl_num	origin	dest	dep_time	de
0	2014-01-01 00:00:00	AA	19805	N338AA	1	JFK	LAX	914.0	
1	2014-01-01 00:00:00	AA	19805	N335AA	3	JFK	LAX	1157.0	
2	2014-01-01 00:00:00	AA	19805	N327AA	21	JFK	LAX	1902.0	
3	2014-01-01 00:00:00	AA	19805	N3EHAA	29	LGA	PBI	722.0	
4	2014-01-01 00:00:00	AA	19805	N319AA	117	JFK	LAX	1347.0	
...	...	...	...	...	...	...	...	...	
20812	2014-01-31 00:00:00	UA	19977	N54711	1253	ROC	ORD	801.0	
20813	2014-01-31 00:00:00	UA	19977	N77525	1429	LGA	CLE	1522.0	
20814	2014-01-31 00:00:00	UA	19977	N37293	1456	LGA	IAH	719.0	
20815	2014-01-31 00:00:00	UA	19977	N24729	1457	LGA	IAH	852.0	
20816	2014-01-31 00:00:00	MQ	20398	N609MQ	3699	BUF	ORD	1208.0	

20817 rows × 14 columns

19. Which airline ID is present maximum times in the dataset

In [289…

```
flight[['airline_id']].count().max()
```

Out[289…

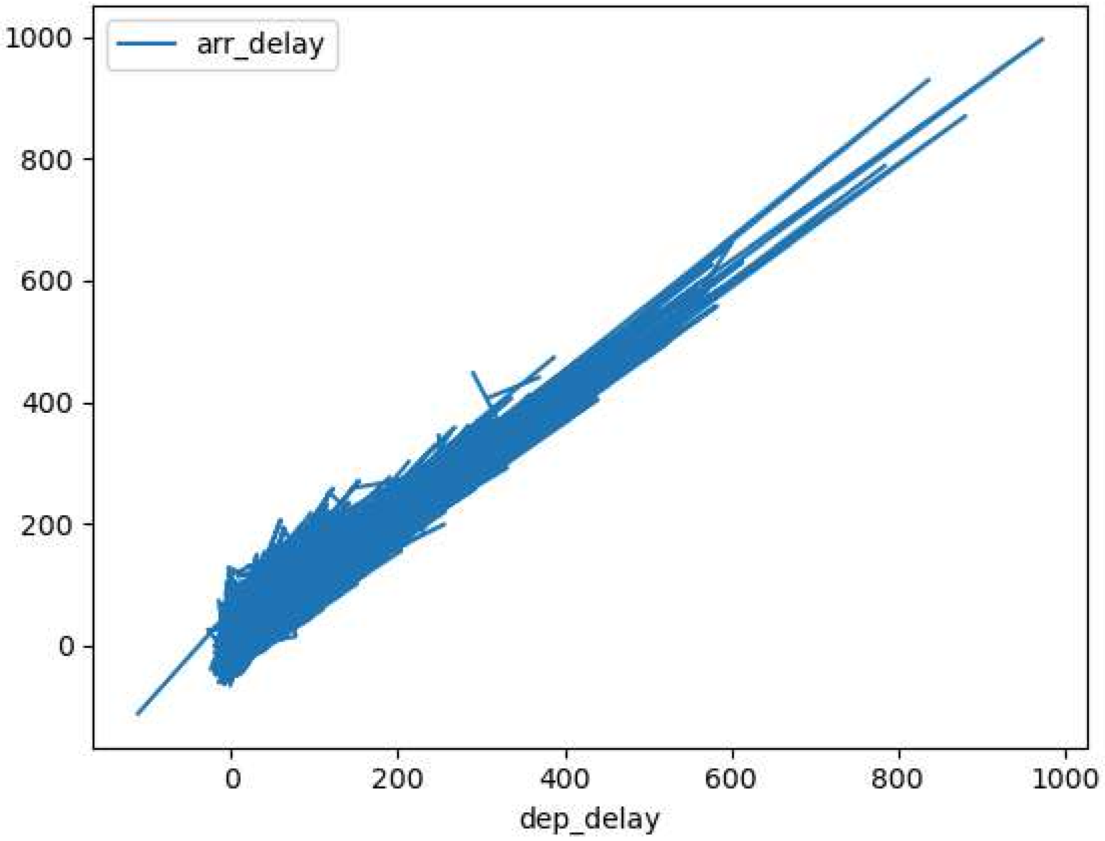
```
20817
```

20. Draw a plot between dep_delay and arr_delay

In [293…

```
import matplotlib.pyplot as plt
x='dep_delay'
y=['arr_delay']
flight.plot(x,y)
```

```
Out[293…  <Axes: xlabel='dep_delay'>
```


arr_delay
1000
800
600
400
200
0
0
200
400
600
800
1000
dep_delay

```
In [ ]:
```