Cont. Pandas for data analysis

1- Implementation for exercise #1

2- Calculating some statistics: mean max, min mode median corelation

3-More pandas functions

1. Solving Exercise (2_1) from Lab02

In [ ]:

```
'''
exercise 2_1

1. Read the file "dataFile" using pandas.
2. Create dataFrame for the data file.
3. Print the first 10 rows using head.
4. Print the last 5 rows.
5. Create a new dataFrame by removing rows with empty cells (how many?).
6. Create a new dataFrame by replacing empty valuse with '111'
7. Find the mean and median for "Pulse" column

'''
```

In [1]:

```
import pandas as pd

df1 = pd.read_csv('dataFile.csv')
type(df1)
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 169 entries, 0 to 168
Data columns (total 4 columns):
Duration    169 non-null int64
Pulse       169 non-null int64
Maxpulse    169 non-null int64
Calories    164 non-null float64
dtypes: float64(1), int64(3)
memory usage: 5.4 KB
```

In [5]:

```
#3
df1.head(10)
print(df1[0:10])
```

Out[5]:

	Duration	Pulse	Maxpulse	Calories
0	60	110	130	409.1
1	60	117	145	479.0
2	60	103	135	340.0
3	45	109	175	282.4
4	45	117	148	406.0
5	60	102	127	300.0
6	60	110	136	374.0
7	45	104	134	253.3
8	30	109	133	195.1
9	60	98	124	269.0
10	60	103	147	329.3
11	60	100	120	250.7
12	60	106	128	345.3
13	60	104	132	379.3
14	60	98	123	275.0
15	60	98	120	215.2
16	60	100	120	300.0
17	45	90	112	NaN
18	60	103	123	323.0
19	45	97	125	243.0
20	60	108	131	364.2
21	45	100	119	282.0
22	60	130	101	300.0
23	45	105	132	246.0
24	60	102	126	334.5
25	60	100	120	250.0
26	60	92	118	241.0
27	60	103	132	NaN
28	60	100	132	280.0
29	60	102	129	380.3
30	60	92	115	243.0
31	45	90	112	180.1

	Duration	Pulse	Maxpulse	Calories
32	60	101	124	299.0
33	60	93	113	223.0
34	60	107	136	361.0
35	60	114	140	415.0
36	60	102	127	300.0
37	60	100	120	300.0
38	60	100	120	300.0
39	45	104	129	266.0
40	45	90	112	180.1
41	60	98	126	286.0
42	60	100	122	329.4
43	60	111	138	400.0
44	60	111	131	397.0
45	60	99	119	273.0
46	60	109	153	387.6
47	45	111	136	300.0
48	45	108	129	298.0
49	60	111	139	397.6

In [8]:

```
#4
print(df1.tail().to_string())
```

```
     Duration  Pulse  Maxpulse  Calories
164        60    105       140     290.8
165        60    110       145     300.0
166        60    115       145     310.2
167        75    120       150     320.4
168        75    125       150     330.4
```

In [9]:

```
print(df1[-5:])
```

```
     Duration  Pulse  Maxpulse  Calories
164        60    105       140     290.8
165        60    110       145     300.0
166        60    115       145     310.2
167        75    120       150     320.4
168        75    125       150     330.4
```

In [11]:

```
#5
print(df1.info())

df2 = df1.dropna()
print(df2.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 169 entries, 0 to 168
Data columns (total 4 columns):
Duration    169 non-null int64
Pulse       169 non-null int64
Maxpulse    169 non-null int64
Calories    164 non-null float64
dtypes: float64(1), int64(3)
memory usage: 5.4 KB
None
<class 'pandas.core.frame.DataFrame'>
Int64Index: 164 entries, 0 to 168
Data columns (total 4 columns):
Duration    164 non-null int64
Pulse       164 non-null int64
Maxpulse    164 non-null int64
Calories    164 non-null float64
dtypes: float64(1), int64(3)
memory usage: 6.4 KB
None
```

In [12]:

```
#6
df3 = df1.fillna(111)
print(df3[:20])
```

```
    Duration  Pulse  Maxpulse  Calories
0         60    110       130     409.1
1         60    117       145     479.0
2         60    103       135     340.0
3         45    109       175     282.4
4         45    117       148     406.0
5         60    102       127     300.0
6         60    110       136     374.0
7         45    104       134     253.3
8         30    109       133     195.1
9         60     98       124     269.0
10        60    103       147     329.3
11        60    100       120     250.7
12        60    106       128     345.3
13        60    104       132     379.3
14        60     98       123     275.0
15        60     98       120     215.2
16        60    100       120     300.0
17        45     90       112     111.0
18        60    103       123     323.0
19        45     97       125     243.0
```

In [13]:

```
#7
x = df1["Pulse"].mean()
y = df1["Pulse"].median()

print(x)
print(y)
```

```
107.46153846153847
105.0
```

In [ ]:

In [ ]:

In [ ]:

In [14]:

```
import pandas as pd

df = pd.read_csv('dataFile.csv')
print(df[0:20].to_string())
```

```
    Duration  Pulse  Maxpulse  Calories
0         60    110       130     409.1
1         60    117       145     479.0
2         60    103       135     340.0
3         45    109       175     282.4
4         45    117       148     406.0
5         60    102       127     300.0
6         60    110       136     374.0
7         45    104       134     253.3
8         30    109       133     195.1
9         60     98       124     269.0
10        60    103       147     329.3
11        60    100       120     250.7
12        60    106       128     345.3
13        60    104       132     379.3
14        60     98       123     275.0
15        60     98       120     215.2
16        60    100       120     300.0
17        45     90       112       NaN
18        60    103       123     323.0
19        45     97       125     243.0
```

In [15]:

```
#find the data Correlations
print(df.corr())
```

```
          Duration     Pulse  Maxpulse  Calories
Duration  1.000000 -0.155408  0.009403  0.922717
Pulse    -0.155408  1.000000  0.786535  0.025121
Maxpulse  0.009403  0.786535  1.000000  0.203813
Calories  0.922717  0.025121  0.203813  1.000000
```

In [16]:

```
# correlation for a range of rows
print(df[0:50].corr())
```

```
          Duration     Pulse  Maxpulse  Calories
Duration  1.000000  0.026263 -0.107569  0.432934
Pulse     0.026263  1.000000  0.462226  0.653835
Maxpulse -0.107569  0.462226  1.000000  0.515176
Calories  0.432934  0.653835  0.515176  1.000000
```

In [17]:

```
#print(df[Duration, Pulse , Maxpulse].corr())

print(df.corr().loc['Pulse','Maxpulse'])
```

```
0.7865346759989718
```

3. More pandas function

In [21]:
```
import numpy as np
import pandas as pd
data1 = pd.DataFrame(np.arange(16).reshape((4, 4)), index=['r1', 'r2', 'r3', 'r
                     columns=['c1', 'c2', 'c3', 'c4'])
```

In [19]:
```
print(data1)
```

```
    c1  c2  c3  c4
r1   0   1   2   3
r2   4   5   6   7
r3   8   9  10  11
r4  12  13  14  15
```

In [ ]:
```
# dropping rows and columns
```

In [23]:
```
#drop rows
data2 = data1.drop(['r1', 'r2'])
print(data2)
```

```
    c1  c2  c3  c4
r3   8   9  10  11
r4  12  13  14  15
```

In [25]:
```
#drop columns
#data2 = data1.drop('c2', axis=1)
data3 = data1.drop('c2', axis='columns')
print(data3)
```

```
    c1  c3  c4
r1   0   2   3
r2   4   6   7
r3   8  10  11
r4  12  14  15
```

In [26]:
```
print(data1)
```

```
    c1  c2  c3  c4
r1   0   1   2   3
r2   4   5   6   7
r3   8   9  10  11
r4  12  13  14  15
```

In [31]:

```
data1[data1['c3'] > 10]
```

Out[31]:

	c1	c2	c3	c4
r4	12	13	14	15

In [32]:

```
#selecting with loc and iloc
#select a single row and multiple columns by label
#loc for colums and row names
data1.loc['r3', ['c1', 'c2']]
```

Out[32]:

```
c1    8
c2    9
Name: r3, dtype: int32
```

In [33]:

```
print(data1)
#iloc for numeric selection
data1.iloc[2, [3, 0, 1]]
```

```
    c1  c2  c3  c4
r1   0   1   2   3
r2   4   5   6   7
r3   8   9  10  11
r4  12  13  14  15
```

Out[33]:

```
c4    11
c1     8
c2     9
Name: r3, dtype: int32
```

In [34]:

```
# for scalar access, at & iat
#Select a single scalar value by row and column label
data1.at['r2', 'c1']
```

Out[34]:

```
4
```

In [35]:

```
#Select a single scalar value by row and column position (integers)
x = data1.iat[1, 0]
print(x)
```

```
4
```

In [39]:

```
#assign new value to a position
data1.iat[1, 0] = 222
print(data1)
```

```
     c1  c2  c3  c4
r1    0   1   2   3
r2  222   5   6   7
r3    8   9  10  11
r4   12  13  14  15
```

In [42]:

```
#joining two data frames
df2 = pd.DataFrame(np.arange(25.).reshape((5, 5)), columns=list('abcde'),
                   index=list('01234'))
print(df2)
```

```
      a     b     c     d     e
0   0.0   1.0   2.0   3.0   4.0
1   5.0   6.0   7.0   8.0   9.0
2  10.0  11.0  12.0  13.0  14.0
3  15.0  16.0  17.0  18.0  19.0
4  20.0  21.0  22.0  23.0  24.0
```

In [52]:

```
df3 = pd.DataFrame(np.arange(12.).reshape((4, 3)), columns=list('acb'),
                   index=list('0123'))
print(df3)
```

```
     a     c     b
0  0.0   1.0   2.0
1  3.0   4.0   5.0
2  6.0   7.0   8.0
3  9.0  10.0  11.0
```

In [44]:

```
print(df2)
print(df3)
```

```
      a     b     c     d     e
0   0.0   1.0   2.0   3.0   4.0
1   5.0   6.0   7.0   8.0   9.0
2  10.0  11.0  12.0  13.0  14.0
3  15.0  16.0  17.0  18.0  19.0
4  20.0  21.0  22.0  23.0  24.0
     a     b     c
0  0.0   1.0   2.0
1  3.0   4.0   5.0
2  6.0   7.0   8.0
3  9.0  10.0  11.0
```

In [49]:

```
df4 = df2 + df3
print(df4)
```

```
      a     b     c   d   e
0   0.0   3.0   3.0 NaN NaN
1   8.0  11.0  11.0 NaN NaN
2  16.0  19.0  19.0 NaN NaN
3  24.0  27.0  27.0 NaN NaN
4   NaN   NaN   NaN NaN NaN
```

In [50]:

```
df2 - df3
```

Out[50]:

	a	b	c	d	e
0	0.0	-1.0	1.0	NaN	NaN
1	2.0	1.0	3.0	NaN	NaN
2	4.0	3.0	5.0	NaN	NaN
3	6.0	5.0	7.0	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN

In [53]:

```
#sorting values and indeces
df6 = df3.sort_index(axis = 1)
print(df6)
```

```
     a     b     c
0  0.0   2.0   1.0
1  3.0   5.0   4.0
2  6.0   8.0   7.0
3  9.0  11.0  10.0
```

In [55]:

```
df6 = df3.sort_values(by='b', ascending=False)
print(df6)
```

```
     a     c     b
0  0.0   1.0   2.0
1  3.0   4.0   5.0
2  6.0   7.0   8.0
3  9.0  10.0  11.0
```

In [59]:

```
#statistics description summary
df6.iat[1,0] = np.nan
df6
df6.describe()
```

Out[59]:

	a	c	b
count	3.000000	4.000000	4.000000
mean	5.000000	5.500000	6.500000
std	4.582576	3.872983	3.872983
min	0.000000	1.000000	2.000000
25%	3.000000	3.250000	4.250000
50%	6.000000	5.500000	6.500000
75%	7.500000	7.750000	8.750000
max	9.000000	10.000000	11.000000

In [60]:

```
# creating random 20 integer list between 100-200  using numpy
import numpy as np

x = np.random.randint(100, 200, 20)
print(x)
```

```
[127 101 106 178 132 171 133 193 183 119 198 107 187 100 115 192 160 110
 133 185]
```

In [61]:

```
#reshape list into matrix
x = x.reshape((4,5))
print(x)
```

```
[[127 101 106 178 132]
 [171 133 193 183 119]
 [198 107 187 100 115]
 [192 160 110 133 185]]
```

```
In [62]:
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-62-4e901b422fc9> in <module>
----> 1 df10 = pd()

TypeError: 'module' object is not callable
```

```
In [ ]:
```