In [2]:

```
import pandas
food_info = pandas.read_csv("food_info.csv")
print (type(food_info))
print (food_info.dtypes)
print (help(pandas.read_csv))
```

```
<class 'pandas.core.frame.DataFrame'>
NDB_No                 int64
Shrt_Desc             object
Water_(g)            float64
Energ_Kcal             int64
Protein_(g)          float64
Lipid_Tot_(g)        float64
Ash_(g)              float64
Carbohydrt_(g)       float64
Fiber_TD_(g)         float64
Sugar_Tot_(g)        float64
Calcium_(mg)         float64
Iron_(mg)            float64
Magnesium_(mg)       float64
Phosphorus_(mg)      float64
Potassium_(mg)       float64
Sodium_(mg)          float64
Zinc_(mg)            float64
Copper_(mg)          float64
```

In [17]:

```
food_info.head()
#first_rows = food_info.head()
#first_rows
#food_info.tail(4)
#print (food_info.tali(3))
#print (food_info.columns)
#print (food_info.shape)
```

Out[17]:

	NDB_No	Shrt_Desc	Water_(g)	Energ_Kcal	Protein_(g)	Lipid_Tot_(g)	Ash_(g)	Carbohyd
0	1001	BUTTER WITH SALT	15.87	717	0.85	81.11	2.11	
1	1002	BUTTER WHIPPED WITH SALT	15.87	717	0.85	81.11	2.11	
2	1003	BUTTER OIL ANHYDROUS	0.24	876	0.28	99.48	0.00	
3	1004	CHEESE BLUE	42.41	353	21.40	28.74	5.11	
4	1005	CHEESE BRICK	41.11	371	23.24	29.68	3.18	

5 rows × 36 columns

In [20]:

```
#print (food_info.loc[0])

food_info.loc[6]

#food_info.loc[8620]
```

Out[20]:

```
NDB_No                     1007
Shrt_Desc        CHEESE CAMEMBERT
Water_(g)                  51.8
Energ_Kcal                  300
Protein_(g)                19.8
Lipid_Tot_(g)             24.26
Ash_(g)                    3.68
Carbohydrt_(g)             0.46
Fiber_TD_(g)                  0
Sugar_Tot_(g)              0.46
Calcium_(mg)                388
Iron_(mg)                  0.33
Magnesium_(mg)               20
Phosphorus_(mg)             347
Potassium_(mg)              187
Sodium_(mg)                 842
Zinc_(mg)                  2.38
Copper_(mg)               0.021
```

In [22]:

```
#food_info.loc[3:6]

#two_five_ten = [2, 5, 10]
#food_info.loc[two_five_ten]

food_info.loc[[2,5,10]]
```

Out[22]:

	NDB_No	Shrt_Desc	Water_(g)	Energ_Kcal	Protein_(g)	Lipid_Tot_(g)	Ash_(g)	Carbohy
2	1003	BUTTER OIL ANHYDROUS	0.24	876	0.28	99.48	0.00	
5	1006	CHEESE BRIE	48.42	334	20.75	27.68	2.70	
10	1011	CHEESE COLBY	38.20	394	23.76	32.11	3.36	

3 rows × 36 columns

In [26]:

```
ndb_col = food_info["NDB_No"]
print (ndb_col)
#col_name = "NDB_No"
#ndb_col = food_info[aol_name]
```

...

In [33]:

```
columns = ["Zinc_(mg)", "Iron_(mg)"]
zinc_copper = food_info[columns]
print (zinc_copper)
```

```
      Zinc_(mg)  Iron_(mg)
0          0.09       0.02
1          0.05       0.16
2          0.01       0.00
3          2.66       0.31
4          2.60       0.43
5          2.38       0.50
6          2.38       0.33
7          2.94       0.64
8          3.43       0.16
9          2.79       0.21
10         3.07       0.76
11         0.40       0.07
12         0.33       0.16
13         0.47       0.15
14         0.51       0.13
15         0.38       0.14
16         0.51       0.38
17         3.75       0.44
18         2.88       0.65
```

In [34]:

```
col_names = food_info.columns.tolist()
print (col_names)
gram_columns = []

for c in col_names:
    if c.endswith("(g)"):
        gram_columns.append(c)
gram_df = food_info[gram_columns]
print (gram_df.head(3))
```

```
['NDB_No', 'Shrt_Desc', 'Water_(g)', 'Energ_Kcal', 'Protein_(g)', 'Lipid_Tot_(g)',
'Ash_(g)', 'Carbohydrt_(g)', 'Fiber_TD_(g)', 'Sugar_Tot_(g)', 'Calcium_(mg)', 'Iron_
(mg)', 'Magnesium_(mg)', 'Phosphorus_(mg)', 'Potassium_(mg)', 'Sodium_(mg)', 'Zinc_
(mg)', 'Copper_(mg)', 'Manganese_(mg)', 'Selenium_(mcg)', 'Vit_C_(mg)', 'Thiamin_(m
g)', 'Riboflavin_(mg)', 'Niacin_(mg)', 'Vit_B6_(mg)', 'Vit_B12_(mcg)', 'Vit_A_IU',
'Vit_A_RAE', 'Vit_E_(mg)', 'Vit_D_mcg', 'Vit_D_IU', 'Vit_K_(mcg)', 'FA_Sat_(g)', 'FA
_Mono_(g)', 'FA_Poly_(g)', 'Cholestrl_(mg)']
   Water_(g)  Protein_(g)  Lipid_Tot_(g)  Ash_(g)  Carbohydrt_(g)  \
0      15.87         0.85          81.11     2.11            0.06
1      15.87         0.85          81.11     2.11            0.06
2       0.24         0.28          99.48     0.00            0.00

   Fiber_TD_(g)  Sugar_Tot_(g)  FA_Sat_(g)  FA_Mono_(g)  FA_Poly_(g)
0           0.0           0.06      51.368       21.021        3.043
1           0.0           0.06      50.489       23.426        3.012
2           0.0           0.00      61.924       28.732        3.694
```

In [36]:

```
print (food_info["Iron_(mg)"])
div_1000 = food_info["Iron_(mg)"] / 10000
print (div_1000)

add_100 = food_info["Iron_(mg)"] + 100
```

```
29        0.41
        ...
8588      9.00
8589      0.30
8590      0.10
8591      1.63
8592     34.82
8593      2.28
8594      0.17
8595      0.17
8596      4.86
8597      0.25
8598      0.23
8599      0.13
8600      0.11
8601      0.68
8602      7.83
8603      3.11
8604      0.30
8605      0.18
```

In [37]:

```
water_energy = food_info["Water_(g)"] * food_info["Energ_Kcal"]
water_energy = food_info["Water_(g)"] * food_info["Energ_Kcal"]
iron_grams = food_info["Iron_(mg)"] / 1000
print (food_info.shape)
food_info["Iron_(g)"] = iron_grams
print (food_info.shape)
```

```
(8618, 36)
(8618, 37)
```

In [40]:

```
food_info.sort_values("Sodium_(mg)", inplace=True)
print (food_info["Sodium_(mg)"])

food_info.sort_values("Sodium_(mg)", inplace=True, ascending=False)
print (food_info["Sodium_(mg)"])
```

```
2231     0.0
407      0.0
6827     0.0
758      0.0
748      0.0
747      0.0
701      0.0
704      0.0
705      0.0
706      0.0
707      0.0
730      0.0
738      0.0
        ...
8153     NaN
8155     NaN
8156     NaN
8157     NaN
8158     NaN
8159     NaN
```

In [69]:

```
import pandas as pd
import numpy as np
titanic_surivival = pd.read_csv("titanic_train.csv")
titanic_surivival.head()
```

Out[69]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	C
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	

In [44]:

```
age = titanic_surivival["Age"]
print (age.loc[0:10])
age_is_null = pd.isnull(age)
#print (ageage_is_null)
age_null_true = age[age_is_null]
#print (ageage_null_true)
age_null_count = len(age_null_true)
print (age_null_count)
```

```
0     22.0
1     38.0
2     26.0
3     35.0
4     35.0
5      NaN
6     54.0
7      2.0
8     27.0
9     14.0
10     4.0
Name: Age, dtype: float64
177
```

In [45]:

```
mean_age = sum(titanic_surivival["Age"]) / len(titanic_surivival["Age"])
print (mean_age)
```

```
nan
```

In [46]:

```
good_ages = titanic_surivival["Age"][age_is_null == False]
#print good_ages
correct_mean_age = sum(good_ages) / len(good_ages)
print (correct_mean_age)
```

```
29.69911764705882
```

In [47]:

```
correct_mean_age = titanic_surivival["Age"].mean()
print (correct_mean_age)
```

```
29.69911764705882
```

In [70]:

```
passenger_classes = [1, 2, 3]
fares_by_class = {}
for this_class in passenger_classes:
    pclsaa_rows = titanic_surivival[titanic_surivival["Pclass"] == this_class]
    pclass_fares = pclass_rows["Fare"]
    fare_for_class = pclass_fares.mean()
    fares_by_class[this_class] = fare_for_class
    print (fares_by_class)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-70-a3eddfa52d74> in <module>
      3 for this_class in passenger_classes:
      4     pclsaa_rows = titanic_surivival[titanic_surivival["Pclass"] == this_clas
s]
----> 5     pclass_fares = pclass_rows["Fare"]
      6     fare_for_class = pclass_fares.mean()
      7     fares_by_class[this_class] = fare_for_class

NameError: name 'pclass_rows' is not defined
```

In [51]:

```
passenger_surival = titanic_surivival.pivot_table(index="Pclass",values="Survived",aggfunc=np.mean)
print (passenger_surival)
```

```
        Survived
Pclass
1       0.629630
2       0.472826
3       0.242363
```

In [52]:

```
passenger_age = titanic_surivival.pivot_table(index="Pclass", values="Age")
print (passenger_age)
```

```
              Age
Pclass
1       38.233441
2       29.877630
3       25.140620
```

In [55]:

```
port_stats = titanic_surivival.pivot_table(index="Embarked", values=["Fare","Survived"], aggfunc=np.
print (port_stats)
```

```
                Fare  Survived
Embarked
C         10072.2962        93
Q          1022.2543        30
S         17439.3988       217
```

In [57]:

```
drop_na_columns = titanic_surivival.dropna(axis=1)
new_titanic_survival = titanic_surivival.dropna(axis=0,subset=["Age", "Sex"])
print (new_titanic_survival)
```

```
870       0              349248    7.8958          NaN        S
871       1               11751   52.5542          D35        S
872       0                 695    5.0000  B51 B53 B55        S
873       0              345765    9.0000          NaN        S
874       0          P/PP 3381   24.0000          NaN        C
875       0                2667    7.2250          NaN        C
876       0                7534    9.8458          NaN        S
877       0              349212    7.8958          NaN        S
879       1               11767   83.1583          C50        C
880       1              230433   26.0000          NaN        S
881       0              349257    7.8958          NaN        S
882       0                7552   10.5167          NaN        S
883       0  C.A./SOTON 34068   10.5000          NaN        S
884       0   SOTON/OQ 392076    7.0500          NaN        S
885       5              382652   29.1250          NaN        Q
886       0              211536   13.0000          NaN        S
887       0              112053   30.0000          B42        S
889       0              111369   30.0000         C148        C
890       0              370376    7.7500          NaN        Q
```

In [63]:

```
row_index_83_age = titanic_surivival.loc[83, "Age"]
row_index_100_pclass = titanic_surivival.loc[766,"Pclass"]
print (row_index_83_age)
print (row_index_100_pclass)
```

```
28.0
1
```

In [64]:

```
new_titanic_survival = titanic_surivival.sort_values("Age",ascending=False)
print (new_titanic_survival[0:10])
titanic_reindexed = new_titanic_survival.reset_index(drop=True)
print ("------------------")
print (titanic_reindexed.loc[0:10])
```

```
     PassengerId  Survived  Pclass                                Name  \
630          631         1       1  Barkworth, Mr. Algernon Henry Wilson
851          852         0       3                 Svensson, Mr. Johan
493          494         0       1              Artagaveytia, Mr. Ramon
96            97         0       1            Goldschmidt, Mr. George B
116          117         0       3               Connors, Mr. Patrick
672          673         0       2         Mitchell, Mr. Henry Michael
745          746         0       1        Crosby, Capt. Edward Gifford
33            34         0       2              Wheadon, Mr. Edward H
54            55         0       1      Ostby, Mr. Engelhart Cornelius
280          281         0       3                   Duane, Mr. Frank

      Sex   Age  SibSp  Parch      Ticket     Fare Cabin Embarked
630  male  80.0      0      0       27042  30.0000   A23        S
851  male  74.0      0      0      347060   7.7750   NaN        S
493  male  71.0      0      0    PC 17609  49.5042   NaN        C
96   male  71.0      0      0    PC 17754  34.6542    A5        C
116  male  70.5      0      0      370369   7.7500   NaN        Q
672  male  70.0      0      0  C.A. 24580  10.5000   NaN        S
```

In [65]:

```
def hundreth_row(column):
    hundredth_item = column.loc[99]
    return hundredth_item

hundreth_row = titanic_surivival.apply(hundreth_row)
print (hundreth_row)
```

```
PassengerId                  100
Survived                       0
Pclass                         2
Name          Kantor, Mr. Sinai
Sex                         male
Age                           34
SibSp                          1
Parch                          0
Ticket                    244367
Fare                          26
Cabin                        NaN
Embarked                       S
dtype: object
```

In [67]:

```
def is_minor(row):
    if row["Age"] < 18:
        return True
    else:
        return False

def generate_age_label(row):
    age = row["Age"]
    if pd.isnull(age):
        return "unknowa"
    elif age < 18:
        return "minor"
    else:
        return "adult"
age_labels = titanic_surivival.apply(generate_age_label, axis=1)
print (age_labels)
```

```
0        adult
1        adult
2        adult
3        adult
4        adult
5      unknowa
6        adult
7        minor
8        adult
9        minor
10       minor
11       adult
12       adult
13       adult
14       minor
15       adult
16       minor
17     unknowa
18       adult
```

In [68]:

```
titanic_surivival['age_labels'] = age_labels
age_group_survival = titanic_surivival.pivot_table(index="age_labels", values="Survived")
print (age_group_survival)
```

```
            Survived
age_labels
adult       0.381032
minor       0.539823
unknowa     0.293785
```

In [78]:

```
import pandas as pd
fandango = pd.read_csv('fandango_score_comparison.csv')
serier_film = fandango['FILM']
print (type(serier_film))
print (serier_film[0:5])
series_rt = fandango['RottenTomatoes']
print (series_rt[0:5])
```

```
<class 'pandas.core.series.Series'>
0    Avengers: Age of Ultron (2015)
1                 Cinderella (2015)
2                    Ant-Man (2015)
3              Do You Believe? (2015)
4     Hot Tub Time Machine 2 (2015)
Name: FILM, dtype: object
0    74
1    85
2    80
3    18
4    14
Name: RottenTomatoes, dtype: int64
```

In [85]:

```
from pandas import Series
film_names = serier_film.values
print (type(film_names))

#print (film_names)
rt_scores = series_rt.values
#orint (rt_scores)

series_custom = Series(rt_scores , index=film_names)
series_custom[['Minions (2015)', 'Leviathan (2014)']]
```

```
<class 'numpy.ndarray'>
```

Out[85]:

```
Minions (2015)      54
Leviathan (2014)    99
dtype: int64
```

In [86]:

```
series_custom = Series(rt_scores , index=film_names)
series_custom[['Minions (2015)', 'Leviathan (2014)']]
fiveten = series_custom[5:10]
print (fiveten)
```

```
The Water Diviner (2015)        63
Irrational Man (2015)           42
Top Five (2014)                 86
Shaun the Sheep Movie (2015)    99
Love & Mercy (2015)             89
dtype: int64
```

In [87]:

```
original_index = series_custom.index.tolist()
sorted_index = sorted(original_index)
sorted_by_index = series_custom.reindex(sorted_index)
print (sorted_by_index)
```

```
'71 (2015)                                97
5 Flights Up (2015)                       52
A Little Chaos (2015)                     40
A Most Violent Year (2014)                90
About Elly (2015)                         97
Aloha (2015)                              19
American Sniper (2015)                    72
American Ultra (2015)                     46
Amy (2015)                                97
Annie (2014)                              27
Ant-Man (2015)                            80
Avengers: Age of Ultron (2015)            74
Big Eyes (2014)                           72
Birdman (2014)                            92
Black Sea (2015)                          82
Black or White (2015)                     39
Blackhat (2015)                           34
Cake (2015)                               49
Chappie (2015)                            30
```

In [88]:

```
sc2 = series_custom.sort_index()
sc3 = series_custom.sort_values()
print (sc3[0:10])
```

```
Paul Blart: Mall Cop 2 (2015)     5
Hitman: Agent 47 (2015)           7
Hot Pursuit (2015)                8
Fantastic Four (2015)             9
Taken 3 (2015)                    9
The Boy Next Door (2015)         10
The Loft (2015)                  11
Unfinished Business (2015)       11
Mortdecai (2015)                 12
Seventh Son (2015)               12
dtype: int64
```

In [128]:

```
import numpy as np
print (np.add(series_custom,series_custom))
np.sin(series_custom)
np.max(series_custom)
```

```
Avengers: Age of Ultron (2015)                 148
Cinderella (2015)                              170
Ant-Man (2015)                                 160
Do You Believe? (2015)                          36
Hot Tub Time Machine 2 (2015)                   28
The Water Diviner (2015)                       126
Irrational Man (2015)                           84
Top Five (2014)                                172
Shaun the Sheep Movie (2015)                   198
Love & Mercy (2015)                            178
Far From The Madding Crowd (2015)              168
Black Sea (2015)                               164
Leviathan (2014)                               198
Unbroken (2014)                                102
The Imitation Game (2014)                      180
Taken 3 (2015)                                  18
Ted 2 (2015)                                    92
Southpaw (2015)                                118
Night at the Museum: Secret of the Tomb (2014) 100
```

In [92]:

```
series_custom > 50
```

Out[92]:

```
Avengers: Age of Ultron (2015)                 True
Cinderella (2015)                              True
Ant-Man (2015)                                 True
Do You Believe? (2015)                        False
Hot Tub Time Machine 2 (2015)                 False
The Water Diviner (2015)                       True
Irrational Man (2015)                         False
Top Five (2014)                                True
Shaun the Sheep Movie (2015)                   True
Love & Mercy (2015)                            True
Far From The Madding Crowd (2015)              True
Black Sea (2015)                               True
Leviathan (2014)                               True
Unbroken (2014)                                True
The Imitation Game (2014)                      True
Taken 3 (2015)                                False
Ted 2 (2015)                                  False
Southpaw (2015)                                True
```

In [94]:

```
import pandas as pd
fandango = pd.read_csv('fandango_score_comparison.csv')
print (type(fandango))
fandango_films = fandango.set_index('FILM', drop=False)
print (fandango_films.index)
```

```
<class 'pandas.core.frame.DataFrame'>
Index(['Avengers: Age of Ultron (2015)', 'Cinderella (2015)', 'Ant-Man (2015)',
       'Do You Believe? (2015)', 'Hot Tub Time Machine 2 (2015)',
       'The Water Diviner (2015)', 'Irrational Man (2015)', 'Top Five (2014)',
       'Shaun the Sheep Movie (2015)', 'Love & Mercy (2015)',
       ...
       'The Woman In Black 2 Angel of Death (2015)', 'Danny Collins (2015)',
       'Spare Parts (2015)', 'Serena (2015)', 'Inside Out (2015)',
       'Mr. Holmes (2015)', ''71 (2015)', 'Two Days, One Night (2014)',
       'Gett: The Trial of Viviane Amsalem (2015)',
       'Kumiko, The Treasure Hunter (2015)'],
      dtype='object', name='FILM', length=146)
```

In [95]:

```
import numpy as np
types = fandango_films.dtypes
print (types)
float_columns = types[types.values == 'float64'].index
float_df = fandango_films[float_columns]
print (float_df)
deviations = float_df.apply(lambda x: np.std(x))
```

```
FILM                         object
RottenTomatoes                int64
RottenTomatoes_User           int64
Metacritic                    int64
Metacritic_User             float64
IMDB                        float64
Fandango_Stars              float64
Fandango_Ratingvalue        float64
RT_norm                     float64
RT_user_norm                float64
Metacritic_norm             float64
Metacritic_user_nom         float64
IMDB_norm                   float64
RT_norm_round               float64
RT_user_norm_round          float64
Metacritic_norm_round       float64
Metacritic_user_norm_round  float64
IMDB_norm_round             float64
Metacritic_user_vote_count    int64
```

In [96]:

```
fandango_films["Avengers: Age of Ultron (2015)": "Hot Tub Time Machine 2 (2015)"]
fandango_films.loc["Avengers: Age of Ultron (2015)": "Hot Tub Time Machine 2 (2015)"]
fandango_films.loc['Kingsman: The Secret Service (2015)']
movies = ['Kingsman: The Secret Service (2015)', 'Do You Believe? (2015)', 'The Water Diviner (2015)
fandango_films.loc[movies]
```

Out[96]:

FILM	FILM	RottenTomatoes	RottenTomatoes_User	Metacritic	Metacritic_User	IMDB
Kingsman: The Secret Service (2015)	Kingsman: The Secret Service (2015)	75	84	58	7.9	7.8
Do You Believe? (2015)	Do You Believe? (2015)	18	84	22	4.7	5.4
The Water Diviner (2015)	The Water Diviner (2015)	63	62	50	6.8	7.2

3 rows × 22 columns

In [98]:

```
fandango = pd.read_csv('fandango_score_comparison.csv')
print (type(fandango))
fandango_films = fandango.set_index('FILM', drop=False)
print (fandango_films.index)
```

```
<class 'pandas.core.frame.DataFrame'>
Index(['Avengers: Age of Ultron (2015)', 'Cinderella (2015)', 'Ant-Man (2015)',
       'Do You Believe? (2015)', 'Hot Tub Time Machine 2 (2015)',
       'The Water Diviner (2015)', 'Irrational Man (2015)', 'Top Five (2014)',
       'Shaun the Sheep Movie (2015)', 'Love & Mercy (2015)',
       ...
       'The Woman In Black 2 Angel of Death (2015)', 'Danny Collins (2015)',
       'Spare Parts (2015)', 'Serena (2015)', 'Inside Out (2015)',
       'Mr. Holmes (2015)', ''71 (2015)', 'Two Days, One Night (2014)',
       'Gett: The Trial of Viviane Amsalem (2015)',
       'Kumiko, The Treasure Hunter (2015)'],
      dtype='object', name='FILM', length=146)
```

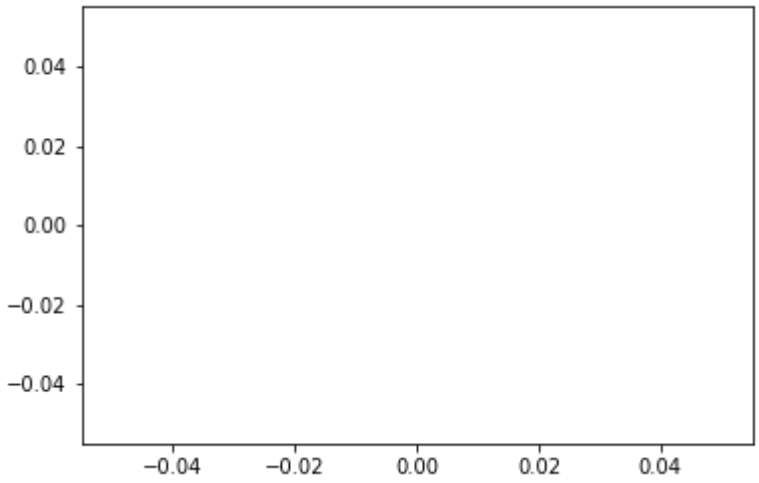
In [100]:

```
import pandas as pd
unrate = pd.read_csv('unrate.csv')
unrate['DATE'] = pd.to_datetime(unrate['DATE'])
print (unrate.head())
```

```
        DATE  VALUE
0 1948-01-01    3.4
1 1948-02-01    3.8
2 1948-03-01    4.0
3 1948-04-01    3.9
4 1948-05-01    3.5
```

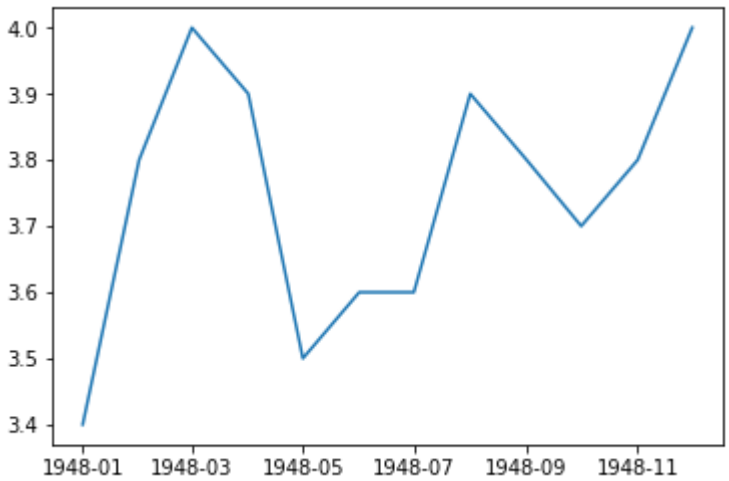
In [102]:

```
import matplotlib.pyplot as plt
plt.plot()
plt.show()
```



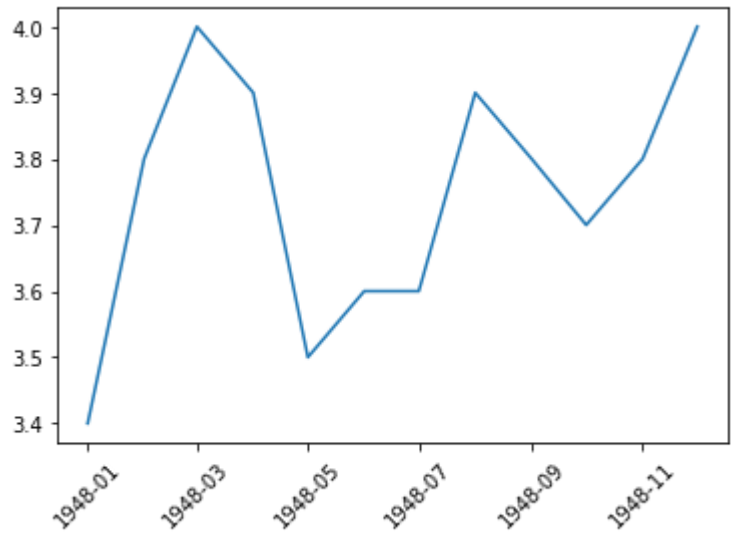
In [103]:

```
first_twelve = unrate[0:12]
plt.plot(first_twelve['DATE'], first_twelve['VALUE'])
plt.show()
```



In [104]:

```
plt.plot(first_twelve['DATE'], first_twelve['VALUE'])
plt.xticks(rotation=45)
plt.show()
```

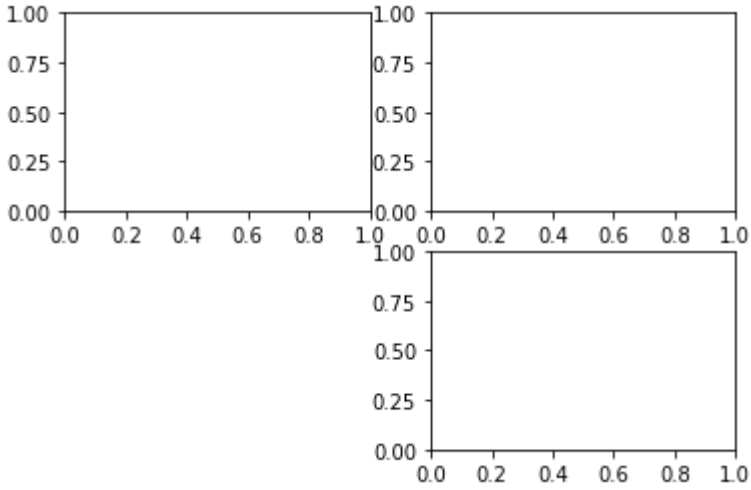


In [106]:

```
plt.plot(first_twelve['DATE'], first_twelve['VALUE'])
plt.xticks(rotation=45)
plt.xlabel('Month')
plt.ylabel('Unemployment Rate')
plt.title('Monthly Unemployment Trends, 1948')
plt.show()
```

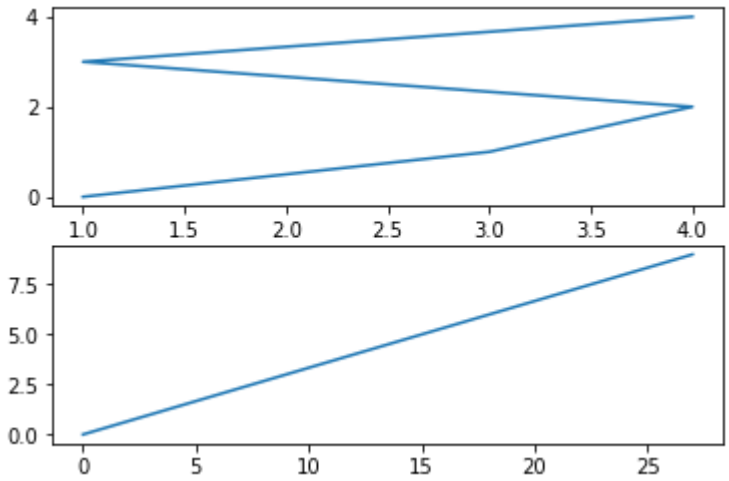
In [108]:

```
import matplotlib.pylab as plt
fig = plt.figure()
ax1 = fig.add_subplot(2,2,1)
ax2 = fig.add_subplot(2,2,2)
ax2 = fig.add_subplot(2,2,4)
plt.show()
```



In [110]:

```
import numpy as np
fig = plt.figure()
ax1 = fig.add_subplot(2,1,1)
ax2 = fig.add_subplot(2,1,2)

ax1.plot(np.random.randint(1,5,5), np.arange(5))
ax2.plot(np.arange(10)*3, np.arange(10))
plt.show()
```

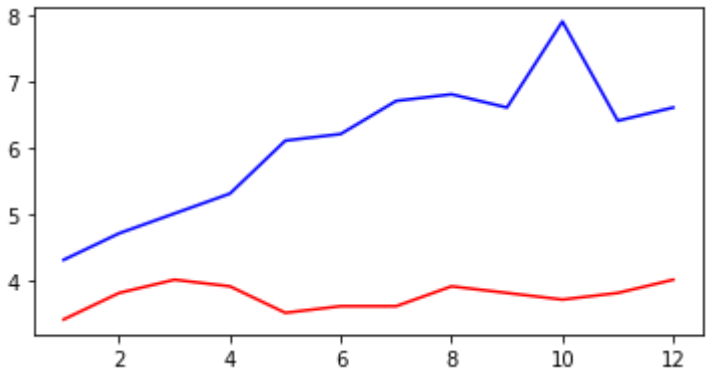


In [112]:

```
unrate['MONTH'] = unrate['DATE'].dt.month
unrate['MONTH'] = unrate['DATE'].dt.month
fig = plt.figure(figsize=(6,3))

plt.plot(unrate[0:12]['MONTH'], unrate[0:12]['VALUE'], c='red')
plt.plot(unrate[12:24]['MONTH'], unrate[12:24]['VALUE'], c='blue')
```

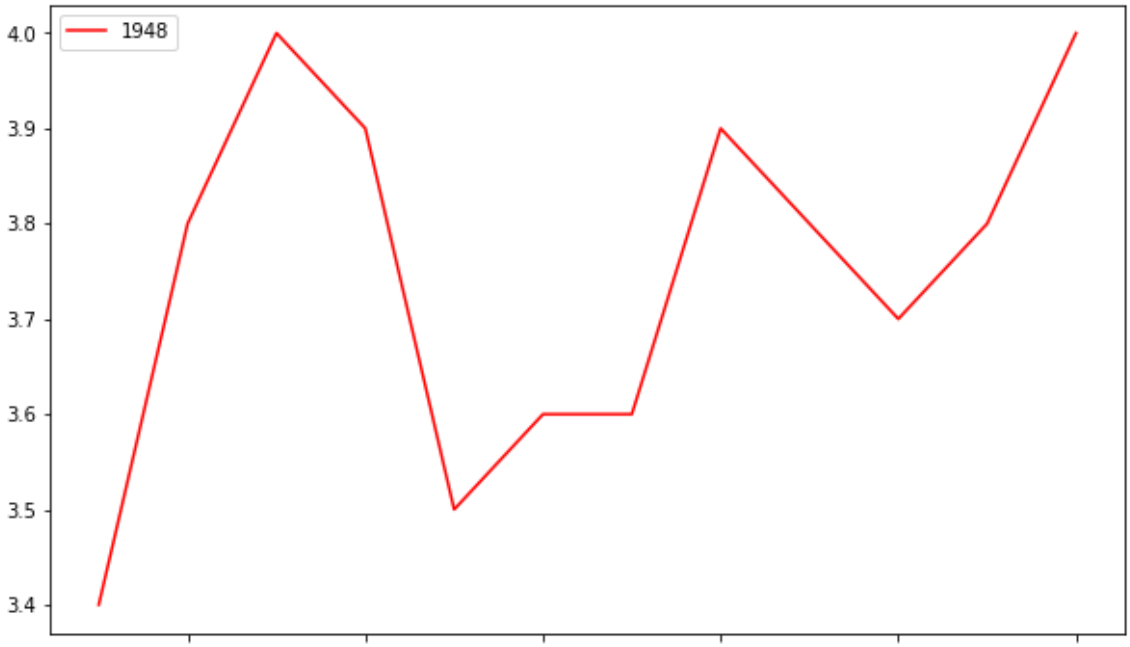
Out[112]:

```
[<matplotlib.lines.Line2D at 0x21c1bee4c18>]
```



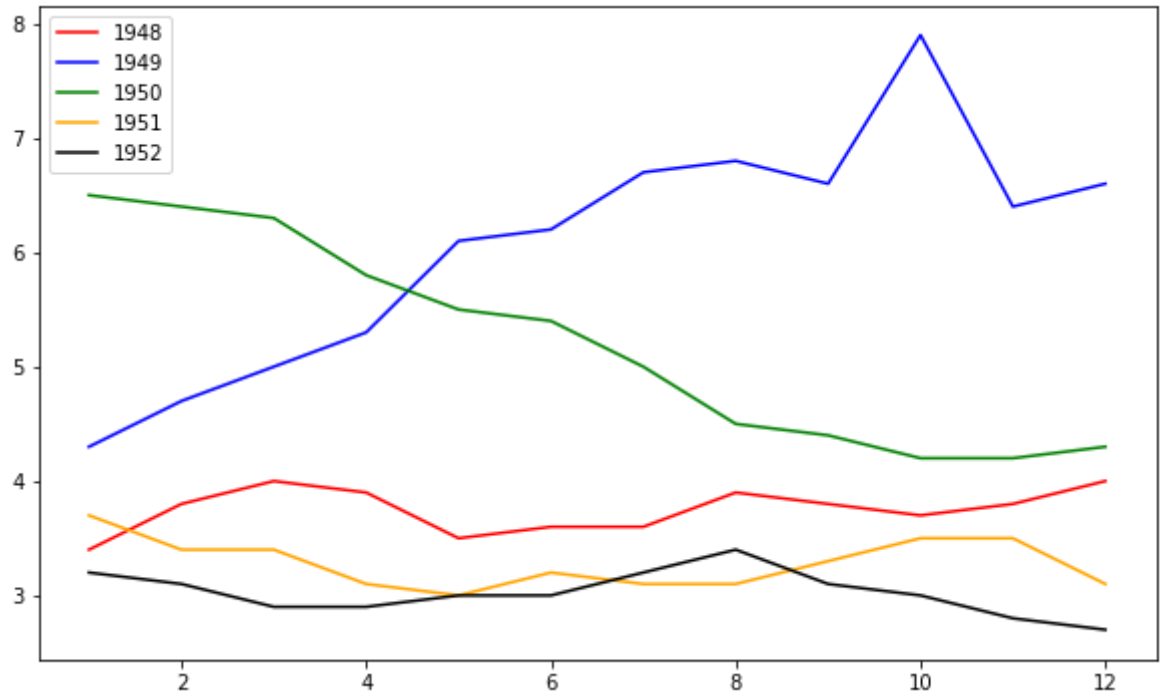
In [119]:

```
fig = plt.figure(figsize=(10,6))
colors = ['red', 'blue', 'green', 'orange', 'black']
for i in range(5):
    start_index = i*12
    end_index = (i+1)*12
    subset = unrate[start_index:end_index]
    label = str(1948 + i)
    plt.plot(subset['MONTH'], subset['VALUE'], c=colors[i], label=label)
    plt.legend(loc='best')
    plt.show()
```



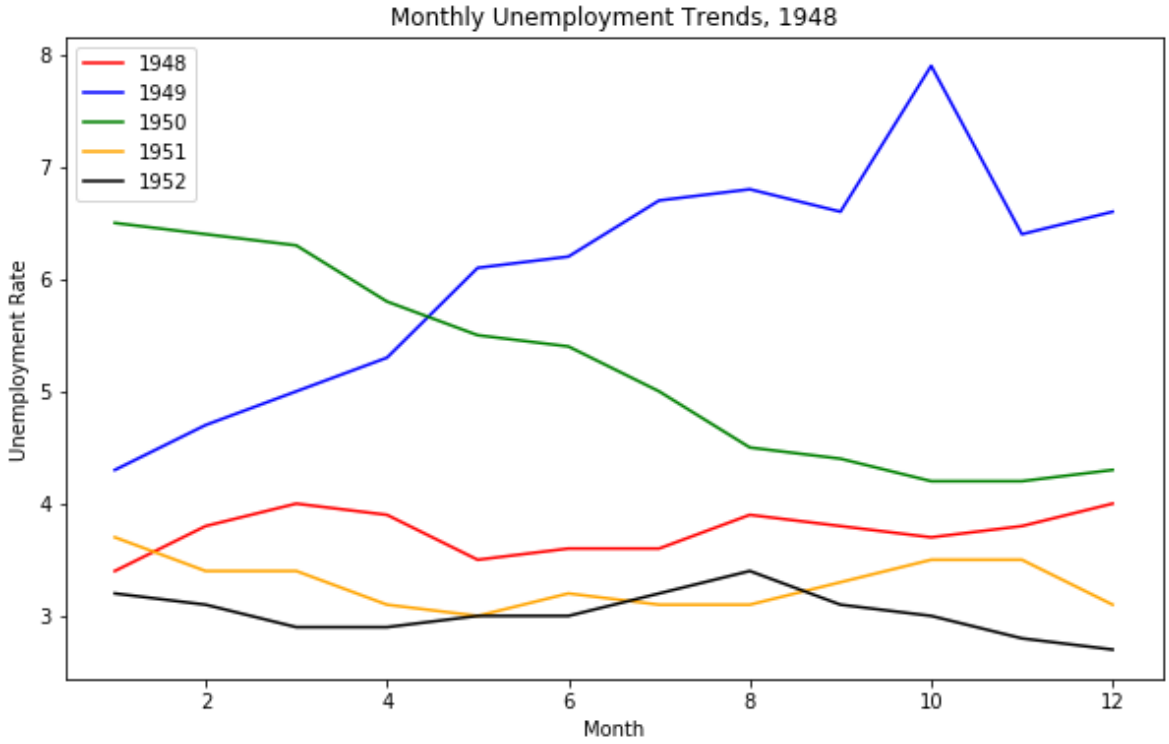
In [116]:

```
fig = plt.figure(figsize=(10,6))
colors = ['red', 'blue', 'green', 'orange', 'black']
for i in range(5):
    start_index = i*12
    end_index = (i+1)*12
    subset = unrate[start_index:end_index]
    label = str(1948 + i)
    plt.plot(subset['MONTH'], subset['VALUE'], c=colors[i], label=label)
plt.legend(loc='upper left')
plt.show()
```



In [120]:

```
fig = plt.figure(figsize=(10,6))
colors = ['red', 'blue', 'green', 'orange', 'black']
for i in range(5):
    start_index = i*12
    end_index = (i+1)*12
    subset = unrate[start_index:end_index]
    label = str(1948 + i)
    plt.plot(subset['MONTH'], subset['VALUE'], c=colors[i], label=label)
plt.legend(loc='upper left')
plt.xlabel('Month')
plt.ylabel('Unemployment Rate')
plt.title('Monthly Unemployment Trends, 1948')
plt.show()
```



In [149]:

```
import pandas as pd
reviews = pd.read_csv('fandango_score_comparison.csv')
cols = ['FILM', 'RT_user_norm', 'Metacritic', 'IMDB_norm', 'Fandango_Ratingvalue', 'Fandango_Stars']
norm_reviews = reviews[cols]
print(norm_reviews[:1])
```

```
                           FILM  RT_user_norm  Metacritic  IMDB_norm  \
0  Avengers: Age of Ultron (2015)           4.3          66        3.9

   Fandango_Ratingvalue  Fandango_Stars
0                   4.5             5.0
```

In [147]:

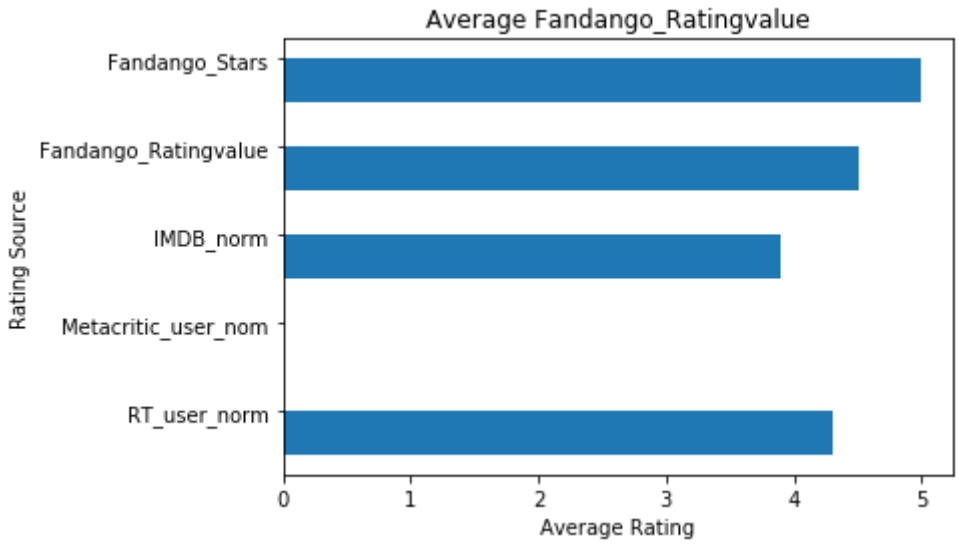
```
import matplotlib.pyplot as plt
from numpy import arange
num_cols = ['FILM', 'RT_user_norm', 'Metacritic', 'IMDB_norm', 'Fandango_Ratingvalue', 'Fandango_Sta
bar_heights = norm_reviews.ix[0, num_cols].values
print (bar_heights)
bar_positions = arange(5) + 0.75
print (bar_positions)
fig, ax = plt.subplots()
#ax.bar(bar_heights, bar_positions, 0.5 )
plt.show()
```

```
['Avengers: Age of Ultron (2015)' 4.3 66 3.9 4.5 5.0]
[0.75 1.75 2.75 3.75 4.75]

D:\Anaconda\ancaonda\lib\site-packages\ipykernel_launcher.py:4: DeprecationWarnin
g:
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexing.html#ix-indexer-is-deprecated
(http://pandas.pydata.org/pandas-docs/stable/indexing.html#ix-indexer-is-deprecate
d)
  after removing the cwd from sys.path.
```

In [150]:

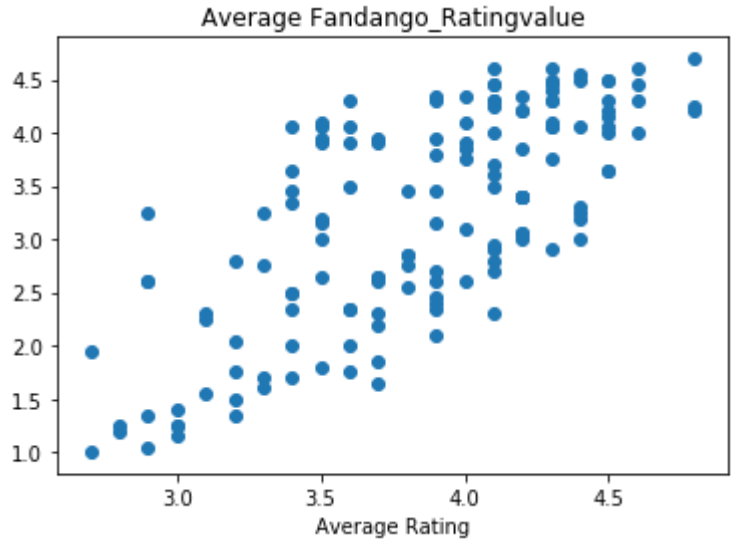
```
import matplotlib.pyplot as plt
from numpy import arange
num_cols = ['RT_user_norm', 'Metacritic_user_nom', 'IMDB_norm', 'Fandango_Ratingvalue', 'Fandango_St

bar_widths = norm_reviews.ix[0, num_cols].values
bar_positions = arange(5) + 0.75
tick_positions = range(1,6)
fig, ax = plt.subplots()
ax.barh(bar_positions, bar_widths, 0.5)

ax.set_yticks(tick_positions)
ax.set_yticklabels(num_cols)
ax.set_ylabel('Rating Source')
ax.set_xlabel('Average Rating')
ax.set_title('Average Fandango_Ratingvalue')
plt.show()
```

```
D:\Anaconda\ancaonda\lib\site-packages\ipykernel_launcher.py:5: DeprecationWarning:
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexing.html#ix-indexer-is-deprecated
 (http://pandas.pydata.org/pandas-docs/stable/indexing.html#ix-indexer-is-deprecate
d)
  """
```



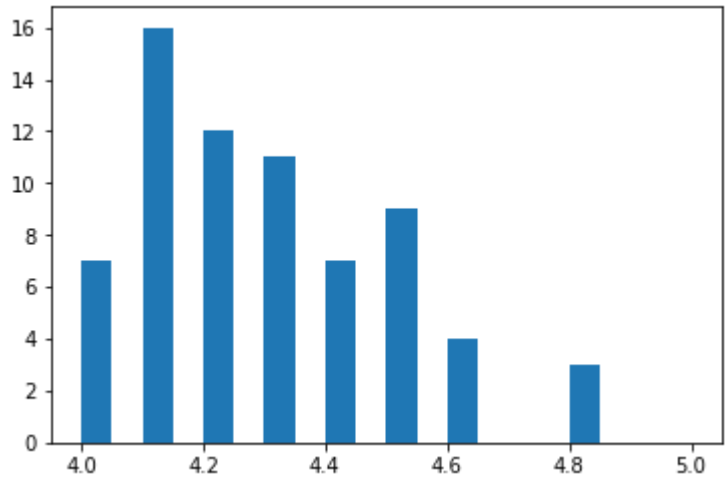
In [151]:

```
fix, ax = plt.subplots()
ax.scatter(norm_reviews['Fandango_Ratingvalue'], norm_reviews['RT_user_norm'])
ax.set_xlabel('Average Rating')
ax.set_title('Average Fandango_Ratingvalue')
plt.show()
```



In [152]:

```
fig, ax = plt.subplots()
ax.hist(norm_reviews['Fandango_Ratingvalue'], range=(4, 5),bins=20)
plt.show()
```



In [153]:

```
import pandas as pd
import matplotlib.pyplot as plt
reviews = pd.read_csv('fandango_score_comparison.csv')
cols = ['FILM', 'RT_user_norm', 'Metacritic', 'IMDB_norm', 'Fandango_Ratingvalue', 'Fandango_Stars']
norm_reviews = reviews[cols]
print(norm_reviews[ :5])
```

```
                           FILM  RT_user_norm  Metacritic  IMDB_norm  \
0  Avengers: Age of Ultron (2015)           4.3          66       3.90
1               Cinderella (2015)           4.0          67       3.55
2                  Ant-Man (2015)           4.5          64       3.90
3          Do You Believe? (2015)           4.2          22       2.70
4   Hot Tub Time Machine 2 (2015)           1.4          29       2.55

   Fandango_Ratingvalue  Fandango_Stars
0                   4.5             5.0
1                   4.5             5.0
2                   4.5             5.0
3                   4.5             5.0
4                   3.0             3.5
```

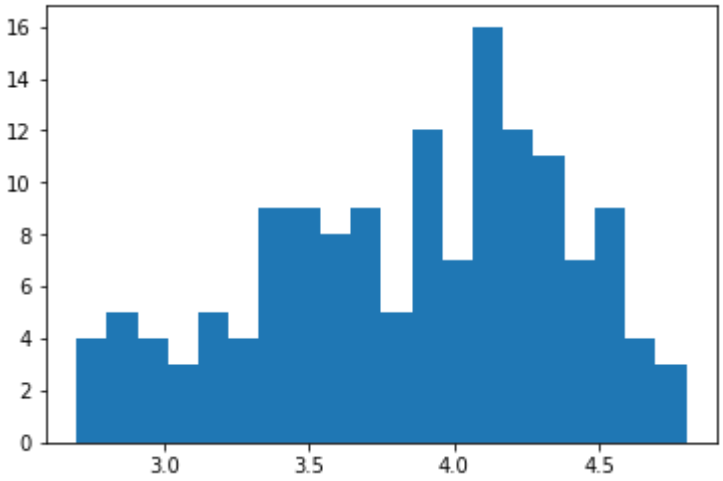
In [156]:

```
fandango_distribution = norm_reviews['Fandango_Ratingvalue'].value_counts()
fandango_distribution = fandango_distribution.sort_index()

imdb_distribution = norm_reviews['IMDB_norm'].value_counts()
imdb_distribution = imdb_distribution.sort_index()

print(fandango_distribution)
print(imdb_distribution)
```

```
3.8     5
3.9    12
4.0     7
4.1    16
4.2    12
4.3    11
4.4     7
4.5     9
4.6     4
4.8     3
Name: Fandango_Ratingvalue, dtype: int64
2.00    1
2.10    1
2.15    1
2.20    1
2.30    2
2.45    2
2.50    1
2.55    1
2.60    2
```

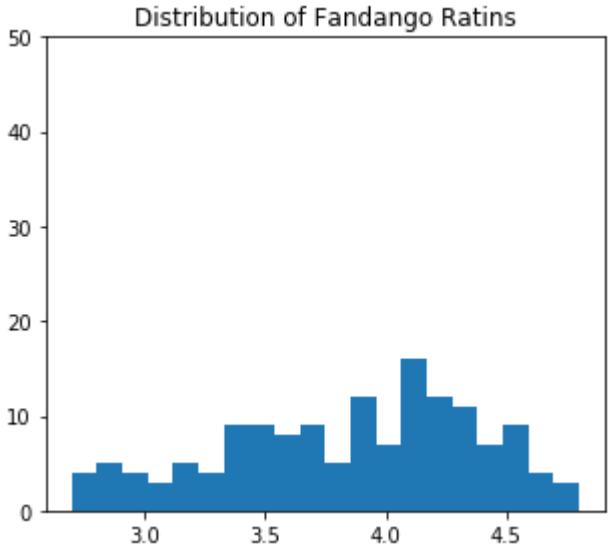
In [202]:

```
fig, ax = plt.subplots()
#ax.hist(norm_reviews['Fandango_Ratingvalue'])
#ax.hist(norm_reviews['Fandango_Ratingvalue'],bins=20)

ax.hist(norm_reviews['Fandango_Ratingvalue'] ,bins=20)
plt.show()
```



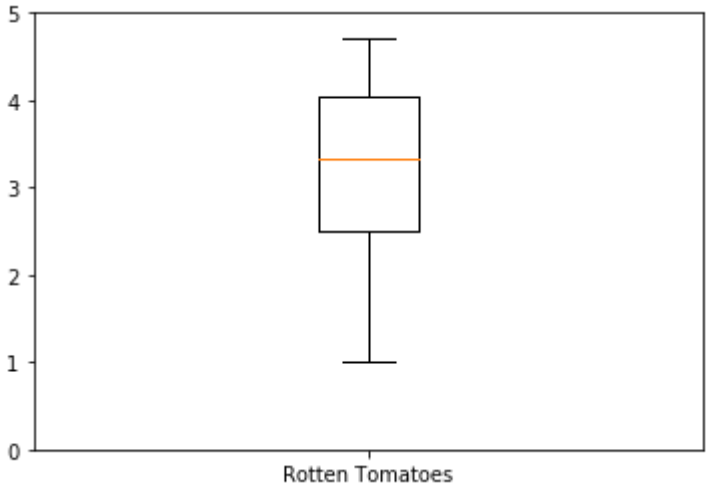
In [185]:

```
fig = plt.figure(figsize=(5, 20))
ax1 = fig.add_subplot(4, 1, 1)
#ax2 = fig.add_subplot(4, 1, 2)
#ax3 = fig.add_subplot(4, 1, 3)
#ax4 = fig.add_subplot(4, 1, 4)
ax1.hist(norm_reviews['Fandango_Ratingvalue'], bins=20)
ax1.set_title('Distribution of Fandango Ratins')
ax1.set_ylim(0,50)

plt.show()
```



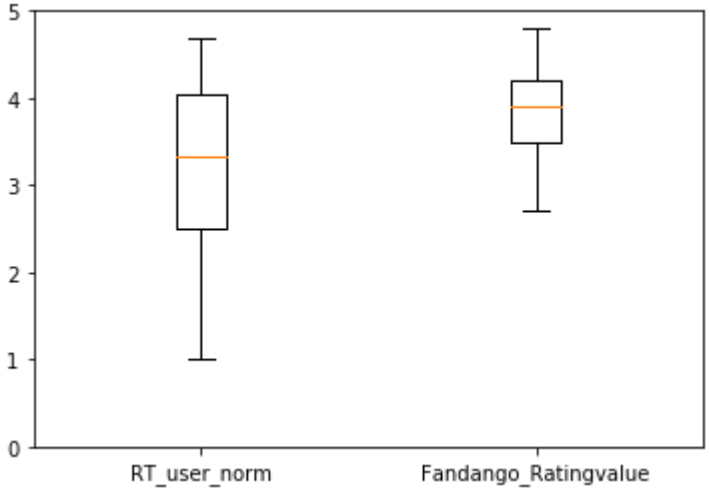
In [187]:

```
fig, ax = plt.subplots()
ax.boxplot(norm_reviews['RT_user_norm'])
ax.set_xticklabels(['Rotten Tomatoes'])
ax.set_ylim(0,5)
plt.show()
```



In [201]:

```
num_cols = ['RT_user_norm', 'Fandango_Ratingvalue']
fig, ax = plt.subplots()
ax.boxplot(norm_reviews[num_cols].values)
ax.set_xticklabels(num_cols, rotation=0)
ax.set_ylim(0,5)
plt.show()
```



In [206]:

```
import pandas as pd
import matplotlib.pyplot as plt

women_degrees = pd.read_csv('percent-bachelors-degrees-women-usa.csv')
major_cats = ['Biology', 'Computer Science', 'Engineering', 'Math and Statistics']

cb_dark_blue = (0/255, 107/255, 164/255)
cb_orange = (255/255, 128/255, 14/255)

fig = plt.figure(figsize=(12, 12))

for sp in range(0,4):
    ax = fig.add_subplot(2,2,sp+1)

    ax.plot(women_degrees['Year'], women_degrees[major_cats[sp]], c=cb_dark_blue, label='Women')
    ax.plot(women_degrees['Year'], 100-women_degrees[major_cats[sp]],c=cb_orange, label='Men')
    for key,spine in ax.spines.items():
        spine.set_visible(False)
    ax.set_xlim(1968, 2011)
    ax.set_ylim(0, 100)
    ax.set_title(major_cats[sp])
    #ax.tick_params(bottom="off", top="off", left="off", ringht="off")

plt.legend(loc='upper right')
plt.show()
```