```
#Section 1 for printing "Hello World"
print("Hello World!")
```

```
 Hello World!
```

```
#Section 2.1 cardinality of unique sets
import csv
def cardinality_items(file_name):
    itemSet = set()
    with open(file_name) as csvFile:
        items = csv.reader(csvFile)
        for itemRow in items:
            for item in itemRow:
                #I want to strip all items before finding them in the set to aviod any repetition like ' okra' and 'okra'
                item = item.strip()
                itemSet.add(item)
    return len(itemSet)


    return len(itemSet)


cardinality_items("/content/sample_data/basket_data.csv")
```

```
 21
```

```
#Section 2.3
import csv
def all_itemsets(filename):
    def backtrack(start = 0, cur = []):
        if len(cur) == k:
            output.append(cur[:])
            return
        for i in range(start, n):
            cur.append(itemList[i])
            backtrack(i + 1, cur)
            cur.pop()
    itemSet = set()
    with open(filename) as csvFile:
        items = csv.reader(csvFile)
        for itemRow in items:
            for item in itemRow:
              #I want to strip all items before finding them in the set to aviod any repetition like ' okra' and 'okra'
                item = item.strip()
                itemSet.add(item)

    itemList = list(itemSet)
    output = []
    n = len(itemList)
    for k in range(1, n + 1):
        backtrack()

    return output


#the test data I use is created by myself, the csv.file has itemSets looks like['', 'bread', 'basket_data', 'ketchup', 'diapers',
output = all_itemsets("/content/sample_data/basket_data_test.csv")
for itemSet in output:
  print(itemSet)
```

```
 ['butter']
 ['', 'bread']
 ['', 'basket_data']
 ['', 'ketchup']
 ['', 'diapers']
 ['', 'butter']
 ['bread', 'basket_data']
 ['bread', 'ketchup']
 ['bread', 'diapers']
 ['bread', 'butter']
 ['basket_data', 'ketchup']
```

```
['', 'bread', 'ketchup']
['', 'bread', 'diapers']
['', 'bread', 'butter']
['', 'basket_data', 'ketchup']
['', 'basket_data', 'diapers']
['', 'basket_data', 'butter']
['', 'ketchup', 'diapers']
['', 'ketchup', 'butter']
['', 'diapers', 'butter']
['bread', 'basket_data', 'ketchup']
['bread', 'basket_data', 'diapers']
['bread', 'basket_data', 'butter']
['bread', 'ketchup', 'diapers']
['bread', 'ketchup', 'butter']
['bread', 'diapers', 'butter']
['basket_data', 'ketchup', 'diapers']
['basket_data', 'ketchup', 'butter']
['basket_data', 'diapers', 'butter']
['ketchup', 'diapers', 'butter']
['', 'bread', 'basket_data', 'ketchup']
['', 'bread', 'basket_data', 'diapers']
['', 'bread', 'basket_data', 'butter']
['', 'bread', 'ketchup', 'diapers']
['', 'bread', 'ketchup', 'butter']
['', 'bread', 'diapers', 'butter']
['', 'basket_data', 'ketchup', 'diapers']
['', 'basket_data', 'ketchup', 'butter']
['', 'basket_data', 'diapers', 'butter']
['', 'ketchup', 'diapers', 'butter']
['bread', 'basket_data', 'ketchup', 'diapers']
['bread', 'basket_data', 'ketchup', 'butter']
['bread', 'basket_data', 'diapers', 'butter']
['bread', 'ketchup', 'diapers', 'butter']
['basket_data', 'ketchup', 'diapers', 'butter']
['', 'bread', 'basket_data', 'ketchup', 'diapers']
['', 'bread', 'basket_data', 'ketchup', 'butter']
['', 'bread', 'basket_data', 'diapers', 'butter']
['', 'bread', 'ketchup', 'diapers', 'butter']
['', 'basket_data', 'ketchup', 'diapers', 'butter']
['bread', 'basket_data', 'ketchup', 'diapers', 'butter']
['', 'bread', 'basket_data', 'ketchup', 'diapers', 'butter']
```

```
#Section 2.4 assume input s is a set and input d is a list of set
def prob_S(S, D):
    totalCount = len(D)
    count = 0
    for set in D:
        if set == S:
            count += 1
    return count / totalCount



s_set = {1, 2, 5}
d_set = [{1, 2, 5}, {1, 2, 4}, {1, 2, 3}, {1, 2, 5}, {2, 5}]
print(prob_S(s_set, d_set))
```

```
0.4
```

```
#Read data from file
import pandas as pd
movie_title = pd.read_csv('/content/drive/MyDrive/NetflixChallenge/movie_titles.csv', names=['Movie_id', 'YearOfRelease', 'Title']
qualifying_data = pd.read_csv('/content/drive/MyDrive/NetflixChallenge/qualifying.txt', header=None, names=['User_id', 'Date'], us
probe_data = pd.read_csv('/content/drive/MyDrive/NetflixChallenge/probe.txt', header=None, names=['User_id'], usecols=[0])
df1 = pd.read_csv('/content/drive/MyDrive/NetflixChallenge/combined_data_1.txt', header=None, names=['User_id', 'Rating', 'Date'],
df2 = pd.read_csv('/content/drive/MyDrive/NetflixChallenge/combined_data_2.txt', header=None, names=['User_id', 'Rating', 'Date'],
df3 = pd.read_csv('/content/drive/MyDrive/NetflixChallenge/combined_data_3.txt', header=None, names=['User_id', 'Rating', 'Date'],
df4 = pd.read_csv('/content/drive/MyDrive/NetflixChallenge/combined_data_4.txt', header=None, names=['User_id', 'Rating', 'Date'],
```

```
#Data Verification
print(f'file: movie_title.csv, There are {len(movie_title)} data items, a total of {len(movie_title.columns)} cols\n')
print(f'file: combined_data_1.txt, There are {len(df1)} data items, a total of {len(df1.columns)} cols\n')
print(f'file: combined_data_2.txt, There are {len(df2)} data items, a total of {len(df2.columns)} cols\n')
print(f'file: combined_data_3.txt, There are {len(df3)} data items, a total of {len(df3.columns)} cols\n')
print(f'file: combined_data_4.txt, There are {len(df4)} data items, a total of {len(df4.columns)} cols\n')
print(f'file: qualifying.txt, There are {len(qualifying_data)} data items, a total of {len(qualifying_data.columns)} cols\n')
print(f'file: probe.txt, There are {len(probe_data)} data items, a total of {len(probe_data.columns)} cols\n')
```

```
file: movie_title.csv, There are 17770 data items, a total of 3 cols
```

```
file: combined_data_1.txt, There are 24058263 data items, a total of 3 cols

file: combined_data_2.txt, There are 26982302 data items, a total of 3 cols

file: combined_data_3.txt, There are 22605786 data items, a total of 3 cols

file: combined_data_4.txt, There are 26851926 data items, a total of 3 cols

file: qualifying.txt, There are 2834601 data items, a total of 2 cols

file: probe.txt, There are 1425333 data items, a total of 1 cols
```

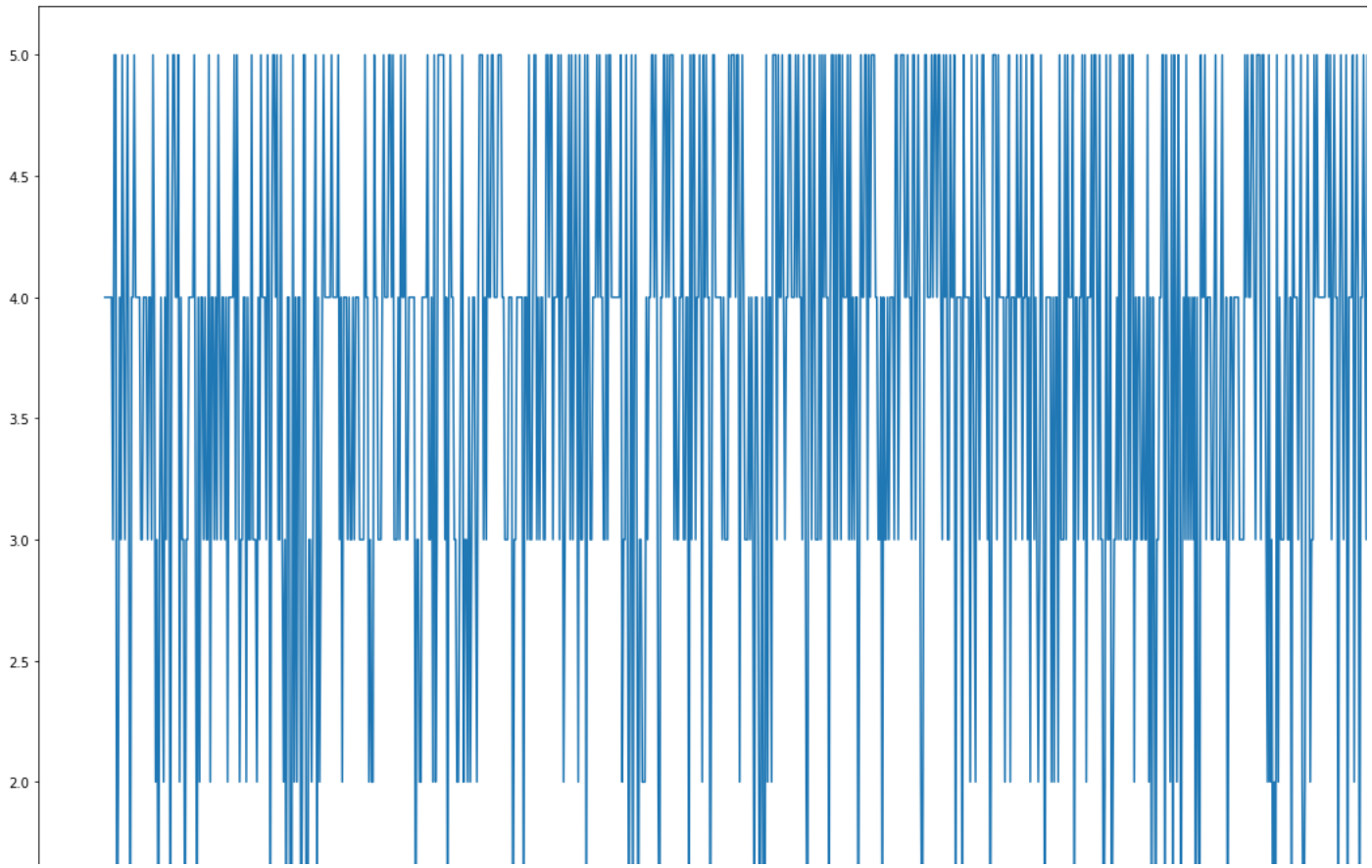
```
#Count lines in txt file without pandas (if use pandas, it will run out of memory and cannot run full dataset in colab)
def count_valid_rating(fileName):
    row_cnt = 0
    for row in open(fileName):
        if ':' in row:
            continue
        else:
            row_cnt += 1
    return row_cnt
```

```
#Section 3.2.1 Count how many total records for each dataset
#find how many ratings in training dataset
valid_ratings = count_valid_rating('/content/drive/MyDrive/NetflixChallenge/combined_data_1.txt')\
                + count_valid_rating('/content/drive/MyDrive/NetflixChallenge/combined_data_2.txt')\
                + count_valid_rating('/content/drive/MyDrive/NetflixChallenge/combined_data_3.txt')\
                + count_valid_rating('/content/drive/MyDrive/NetflixChallenge/combined_data_4.txt')
print(f'There are {valid_ratings} rating records in training dataset')

#find how many rows in movie_title dataset
rowCount = 0
for row in open('/content/drive/MyDrive/NetflixChallenge/movie_titles.csv', encoding='ISO-8859-1'):
  rowCount += 1
print(f'There are {rowCount} records in movie_title dataset')

#find how many records in probe.txt dataset
probe_records = count_valid_rating('/content/drive/MyDrive/NetflixChallenge/probe.txt')
print(f'There are {probe_records} records in probe.txt dataset')
#find how many records in qualifying.txt dataset
qualifying_records = count_valid_rating('/content/drive/MyDrive/NetflixChallenge/qualifying.txt')
print(f'There are {qualifying_records} records in qualifying.txt dataset')
```

```
There are 100480507 rating records in training dataset
There are 17770 records in movie_title dataset
There are 1408395 records in probe.txt dataset
There are 2817131 records in qualifying.txt dataset
```

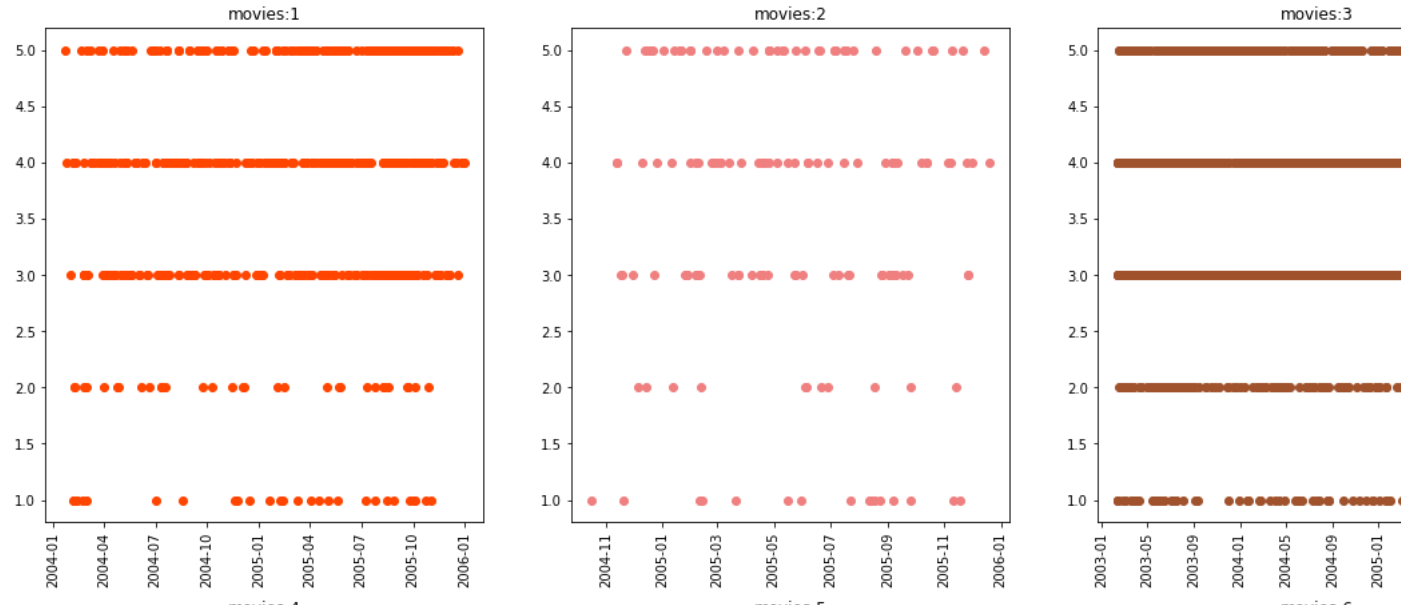
```
# Section 3.2.2 Plot rating and time graph to detect any trend
df1 = pd.read_csv('/content/drive/MyDrive/NetflixChallenge/combined_data_1.txt', header=None, names=['User_id', 'Rating', 'Date'],
df2 = pd.read_csv('/content/drive/MyDrive/NetflixChallenge/combined_data_2.txt', header=None, names=['User_id', 'Rating', 'Date'],
combined_data = df1.append(df2)
movies_index = combined_data[combined_data['Rating'].isnull()].index
sample_test_data = combined_data[:1000]
```

```
#convert date to year so we can see the trend of rating and years
# We first check the how all the movie ratings changed over time
sorted_data = sample_test_data.sort_values('Date')
import matplotlib.pyplot as plt
sorted_data.plot(kind='line', x='Date', y='Rating', figsize=(20, 15))
plt.show()
```




```
#Then we take a small sample to see how each movie's rating changed over time
sample_test_data = combined_data[:10000]
sample_test_data['Date'] = pd.to_datetime(sample_test_data['Date'])

index_sample = movies_index[0:7]
print(index_sample)
palette = ['orangered', 'lightcoral', 'sienna', 'olive', 'cyan', 'gold', 'pink', 'orchid', 'black', 'brown', 'red', 'blue', 'purpl
plt.figure(figsize=(20, 15))
for id, sample in enumerate(index_sample[:-1]):
    plt.subplot(2, 3, id+1)
    movies = sample_test_data.iloc[index_sample[id]]['User_id'][0 : -1]
    draw_data = sample_test_data.iloc[index_sample[id] + 1 : index_sample[id + 1]].sort_values(by='Date')
    plt.plot(draw_data['Date'], draw_data['Rating'], 'o', c = palette[id])
    plt.title(f'movies:{movies}')
    plt.xticks(rotation=90)

plt.show()
```

```
<ipython-input-10-45e0ac0f6762>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-
  sample_test_data['Date'] = pd.to_datetime(sample_test_data['Date'])
Int64Index([0, 548, 694, 2707, 2850, 3991, 5011], dtype='int64')
```




```
#Section 3.2.3
#Sort each comment based on rating date, count first half's average rating and second half's average rating then compare, if the s
#It might indicate that the movie become more popular over time
import numpy as np
popular_cnt = 0
count = 0
sample_index = sample_test_data[sample_test_data['Rating'].isnull()].index
for ind, values in enumerate(sample_index[:-1]):
    movies_id = sample_test_data.iloc[values]['User_id'][0:-1]
    sample_data_ind = sample_test_data.iloc[sample_index[ind]+1:sample_index[ind+1]].sort_values(by='Date')
    rating = sample_test_data.iloc[sample_index[ind]+1:sample_index[ind+1]]['Rating']
    before_rating = np.mean(rating[0:len(rating//2)])
    after_rating = np.mean(rating[len(rating)//2:])

    if after_rating > before_rating:
      popular_cnt += 1

    count += 1

percentage = popular_cnt / count
print(f'The percentage of the films more popularover time is {percentage}')
```

```
The percentage of the films more popularover time is 0.42857142857142855
```

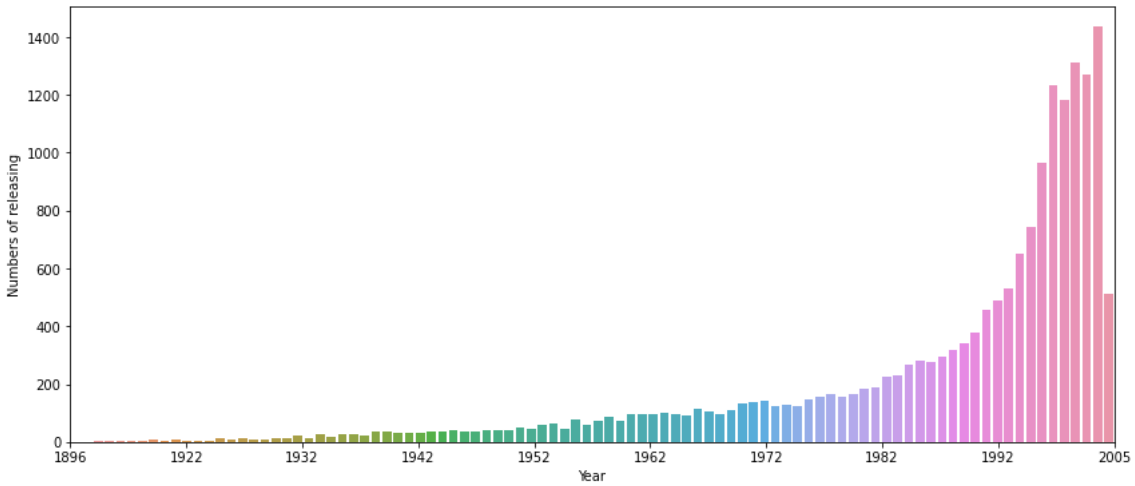
```
#3.2.4
re_count = 0
movies_index_df1 = df1[df1['Rating'].isnull()].index
for ind, values in enumerate(movies_index_df1[:-1]):
    movies_id_df1 = df1.iloc[values]['User_id'][0:-1]
    min_data = min(df1.iloc[movies_index_df1[ind]+1:movies_index_df1[ind+1]]['Date'])
    real_data = movie_title.iloc[ind]['YearOfRelease']
    if min_data[0:4] < str(real_data)[0:4] and str(real_data) != 'nan':
        print(f'The earliest comment was {min_data} that the movie was released on {real_data}')
        re_count += 1

print(f'In combined_data_1, there are a total of {re_count} movies re-released, \
      because the comment time is earlier than the movie release time, indicating that the movie has been re-released')
```

```
The earliest comment was 2003-01-09 that the movie was released on 2004.0
The earliest comment was 2004-12-25 that the movie was released on 2005.0
The earliest comment was 2003-12-26 that the movie was released on 2004.0
The earliest comment was 2002-11-03 that the movie was released on 2003.0
The earliest comment was 2004-12-08 that the movie was released on 2005.0
The earliest comment was 2003-11-20 that the movie was released on 2004.0
The earliest comment was 2004-12-31 that the movie was released on 2005.0
The earliest comment was 2001-04-05 that the movie was released on 2002.0
```

```
The earliest comment was 2000-12-20 that the movie was released on 2001.0
The earliest comment was 2004-12-28 that the movie was released on 2005.0
The earliest comment was 2001-06-12 that the movie was released on 2002.0
The earliest comment was 2004-12-28 that the movie was released on 2005.0
The earliest comment was 2003-11-12 that the movie was released on 2004.0
The earliest comment was 2004-12-31 that the movie was released on 2005.0
The earliest comment was 2001-12-01 that the movie was released on 2002.0
The earliest comment was 2003-12-22 that the movie was released on 2004.0
The earliest comment was 2000-03-01 that the movie was released on 2003.0
The earliest comment was 2004-12-22 that the movie was released on 2005.0
The earliest comment was 2000-12-08 that the movie was released on 2001.0
The earliest comment was 2003-09-22 that the movie was released on 2004.0
The earliest comment was 2003-03-27 that the movie was released on 2004.0
In combined_data_1, there are a total of 21 movies re-released,       because the comment time is earlier than the movie rele
```

```
#Section 3.2.5
#Movie release year
import seaborn as sns

fig, ax = plt.subplots(1, 1, figsize=(14, 6))

movie_title = pd.read_csv('/content/sample_data/movie_titles.csv', names=['Movie_id', 'YearOfRelease', 'Title'], encoding='ISO-885
data = movie_title['YearOfRelease'].value_counts().sort_index()
x = data.index.map(int)
y = data.values

sns.barplot(x, y)
xmin, xmax = plt.xlim()
xtick_labels = [x[0]] + list(x[10:-10:10]) + [x[-1]]
plt.xticks(ticks=np.linspace(xmin, xmax, 10), labels=xtick_labels)
plt.xlabel('Year')
plt.ylabel('Numbers of releasing')
plt.show()
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variables as keyword args
  warnings.warn(
```




```
#Rating distribution
fig, ax = plt.subplots(1, 1, figsize=(10, 5))
data = combined_data['Rating'].value_counts().sort_index()
x = data.index.map(int)
y = data.values

sns.barplot(x, y)
xmin, xmax = plt.xlim()
plt.xticks(ticks=[0, 1, 2, 3, 4, 5])
plt.xlabel('Rating')
plt.ylabel('Numbers')
plt.show()
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning:
  warnings.warn(
```