MDA term project report　　108062533　　洪翎恩

實作內容：
對於 ilms 上給的 data 資料 針對推薦給某 user 做 user-user similarity
在針對與該 user 相似性高於 0.8 且投過相同電影數大於 1 的 高 user 群做運算後推薦給一開始所設定的 user。

步驟：

1. 用 map reduce 的方式先對資料做處理，算出每個 user 的評分平均數，和各 user 的每筆評分數-評分平均數。

```
conf = SparkConf().setMaster("local").setAppName("Term_project")
sc = SparkContext(conf = conf)

print('preprocessing...')
data = sc.textFile("data.txt").map(read_data)

#calculate each avg rating of user ( sum of rat )/( #_items rated by user)
num_item_user = data.map(lambda x: (x[0],1)).reduceByKey(lambda x,y :x+y)
sum_user_rate = data.map(lambda x: (x[0],x[2])).reduceByKey(lambda x,y:x+y)
avg_user_rate = sum_user_rate.join(num_item_user).mapValues(lambda x: x[0]/x[1])

#calculate (r.xs - avg.x)  r.xs : item s rated by user x , avg.x : avg rating of user x
user_item_list=data.map(lambda x : [x[0],(x[1],x[2])]).join(avg_user_rate).mapValues(lambda x: (x[0][0],x[0][1]-x[1])).groupByKey().mapValues
user_item_map=user_item_list.collectAsMap()
```

2. 輸入想針對哪個 user 做推薦電影後，計算該 user 跟其他 user 的 similarity，方法是用講義提到的 pearson correlation coefficient。

```
def pearson_correlation_coefficient(user):
    global user_item_map
    user_x=user
    user_x_item=user_item_map[user_x]
    similarity=[]
    for user_y in range(1,Num_User+1):
        if user_x == str(user_y):
            continue
        user_y_item=user_item_map[str(user_y)]
        XY=0
        YY=0
        XX=0
        divisor=0
        same_item=0
        for item_x in user_x_item:
            for item_y in user_y_item:
                if item_x[0] == item_y[0]:
                    same_item+=1
                    XY+=(item_x[1]*item_y[1])
                    XX+=item_x[1]**2
                    YY+=item_y[1]**2
        divisor=math.sqrt(XX)*math.sqrt(YY)
        if divisor==0:
            continue
        simXY=XY/divisor
        similarity.append((user_x,user_y,simXY,same_item))

    return similarity
```

```
user=input('target user:')

#pearson correlation coefficient
print('user-user similarity by pearson correlation coefficient')
user_similarity_list=sorted(pearson_correlation_coefficient(user),key=lambda x: x[2],reverse=True)
```

3. 之後再選擇前 10 個與該 user 相似度高且投過多於一個相同電影的進行下方運算然後將分數高的 movie 推薦給步驟 2 設定的 user

$$r_{xi} = \frac{\sum_{y \in N} s_{xy} \cdot r_{yi}}{\sum_{y \in N} s_{xy}}$$

```
high_similarity=[]
for element in user_similarity_list:
    if element[2]>=0.8 and element[3]>1:
        high_similarity.append(element)

high_similarity=sorted(high_similarity,key=lambda x: x[2]*x[3],reverse=True)

num_top_similarity_user=10
if len(high_similarity)< num_top_similarity_user:
    num_top_similarity_user=len(high_similarity)
movie_recommand=[]
for i in range(0,num_top_similarity_user):
    similar_user=high_similarity[i][1]
    for j in user_item_map[str(similar_user)]:
        movie_recommand.append(j)

target_user_item=sc.parallelize(user_item_map[str(user)])
movie_recommand=sc.parallelize(movie_recommand).subtract(target_user_item).r
educeByKey(lambda x,y:x+y).mapValues(lambda x:
x/num_top_similarity_user).sortBy(lambda x : x[1],ascending=False)
recommand_list=movie_recommand.collect()[0:3]
```

輸出結果：

user 600 的 user-user similarity

```
1   600     175     0.9749503289171089
2   600     547     0.8570499429953169
3   600     251     0.8223659026996988
4   600     296     0.771883444657502
5   600     531     0.7690638722339562
6   600     106     0.7574647870471609
7   600     505     0.7344786591153836
8   600     209     0.7300948717671961
9   600     124     0.7255205706794807
10  600     258     0.7132308120761193
11  600     609     0.7029235045336888
12  600     50      0.6947821177265846
```

推薦 user 600 三部電影：

```
1   3791
2   38388
3   44613
```