实训3：协同过滤推荐系统

· ***目标***：掌握字典操作和相似性计算。

· ***任务***：读取JSON评分数据，计算相似度，推荐物品。

· ***详细要求***：

o 读取JSON文件（ratings.json），格式为用户-物品评分字典（如{"user1": {"item1": 5, "item2": 3}}）。

o 计算用户间余弦相似度，处理无共同评分情况（返回0）。

o 为目标用户推荐3个未评分的物品，基于其他用户评分和相似度加权。

o 输出推荐结果到recommendations.txt，格式为“物品: 预测评分”。

o 验证输入数据：评分在0-5之间，JSON格式正确。

o 添加日志记录，记录推荐过程的每次计算。

o 提供用户选择功能（如通过命令行输入目标用户）。

· ***技能***：余弦相似度计算、JSON处理、推荐算法、日志记录。

```
# -*- coding: utf-8 -*-
import json
import math
import logging

# 配置日志
logging.basicConfig(filename='recommend.log', level=logging.INFO, format='%(message)s')

# 读取JSON文件
def load_ratings(file_path):
    try:
        with open(file_path, 'r', encoding='utf-8') as f:
            data = json.load(f)
            # 验证评分合法性
            for user, items in data.items():
                for item, score in items.items():
                    if not (0 <= score <= 5):
                        raise ValueError(f"非法评分：用户 {user}, 物品 {item}, 评分 {score}")
            return data
    except Exception as e:
        print(f"读取文件失败：{e}")
        return {}

# 计算用户间余弦相似度
def cosine_similarity(user_ratings1, user_ratings2):
    common_items = set(user_ratings1.keys()) & set(user_ratings2.keys())
    if not common_items:
        return 0  # 没有共同评分

    sum1 = sum(user_ratings1[item] * user_ratings2[item] for item in common_items)
```

```
    sum1_sq = sum(user_ratings1[item] ** 2 for item in common_items)
    sum2_sq = sum(user_ratings2[item] ** 2 for item in common_items)

    denominator = math.sqrt(sum1_sq) * math.sqrt(sum2_sq)
    return sum1 / denominator if denominator != 0 else 0

# 推荐物品
def recommend(user_id, ratings, top_n=3):
    target_ratings = ratings.get(user_id)
    if not target_ratings:
        print(f"用户 {user_id} 不存在。")
        return []

    scores = {}
    sim_sums = {}

    for other_user, other_ratings in ratings.items():
        if other_user == user_id:
            continue
        sim = cosine_similarity(target_ratings, other_ratings)
        logging.info(f"相似度({user_id}, {other_user}) = {sim:.4f}")
        for item, rating in other_ratings.items():
            if item not in target_ratings:
                scores.setdefault(item, 0)
                sim_sums.setdefault(item, 0)
                scores[item] += sim * rating
                sim_sums[item] += sim

    # 计算预测评分
    predictions = []
    for item in scores:
        if sim_sums[item] != 0:
            predicted_score = scores[item] / sim_sums[item]
            predictions.append((item, round(predicted_score, 2)))

    predictions.sort(key=lambda x: x[1], reverse=True)
    return predictions[:top_n]

# 写入推荐结果
def save_recommendations(recommendations, file_path='recommendations.txt'):
    try:
        with open(file_path, 'w', encoding='utf-8') as f:
            for item, score in recommendations:
                f.write(f"{item}: {score}\n")
    except Exception as e:
        print(f"写入推荐结果失败：{e}")

# 主函数
def main():
    ratings = load_ratings('ratings.json')
    if not ratings:
        return

    user_id = input("请输入目标用户ID（如 user1）: ").strip()
    recommendations = recommend(user_id, ratings)
    if recommendations:
        print("推荐结果：")
        for item, score in recommendations:
```

```
            print(f"{item}: {score}")
        save_recommendations(recommendations)
    else:
        print("无推荐结果。")

if __name__ == "__main__":
    main()

```