توضیحات مربوط به کد تمرین گروهی تیم‌بندی دونده‌ها

```
runners_list=input('Enter your name: ').split(',')
teams_list=input('Enter your team :').split('*')
```

لیست دونده ها (که با , وارد می‌شوند) از کاربر گرفته می‌شود و در runners_list ذخیره می‌شود.

لیست دونده ها به همراه تیم هایشان از کاربر گرفته میشود (فرض میشود کاربر بین اعضای هر تیم از , و بین تیم‌ها از * استفاده می‌کند) و در teams_list ذخیره می‌شود.

//

```
def strip(lst):
    lst=[i.strip() for i in lst]
    return lst
```

تابعی برای حذف فاصله از رشته، ورودی تابع یک لیست است که هر عضو آن یک رشته است.

//

```
runners_list=strip(runners_list)
```

حذف فاصله از اسامی دونده ها

//

```
runners_list=set(runners_list)
runners_list=list(runners_list)
runners_list.sort()
```

حذف اسامی تکراری و مرتب کردن آنها بر اساس حروف الفبا

//

```
print('Name of runners:')
for i in range(len(runners_list)):
    print(i ,runners_list[i])
```

چاپ اسامی دونده ها به ترتیب حروف الفبا

//

```
for i in range(len(runners_list)):
    info=dict()
    info['name']=runners_list[i]
    info['distance']=float(input('\n%s Enter your distance:'%runners_list[i]))
    info['duration']=int(input('%s Enter your Duration :'%runners_list[i]))
    info['speed']=round(float(info['distance']/(info['duration']/60)),4)
    info['pace']=round(float(info['duration']/info['distance']),4)
    runners_list[i]=info
```

دریافت اطلاعات از هر دونده (فاصله پیموده شده و مدت زمان)، محاسبه سرعت و pace و ذخیره این اطلاعات در دیکشنری info.

//

```
def sorting(lst,metric):
    lst=sorted(lst,key=lambda i:i[metric],reverse=True)
    return lst
```

تابعی برای مرتب سازی نزولی یک لیست که حاوی تعدادی دیکشنری است. Metric معیاری است که بر اساس آن میخواهیم مرتب‌سازی انجام دهیم.

//

```
print('sorted runners_list based on speed:',sorting(runners_list,'speed'))
```

مرتب سازی runners_list بر اساس سرعت و چاپ خروجی مربوطه.

runners_list یک لیست است که شامل چندین دیکشنری است. هر دیکشنری حاوی اطلاعات هر دونده است (نام، مسافت پیموه شده، مدت زمان، سرعت و pace).

//

```
def find_fastest(lst):#lst is a list of dictionaries
    return sorting(lst,'speed')[0]
```

تابعی برای پیدا کردن سریعترین دونده که این تابع از تابع sorting که قبلا تعریف کردیم استفاده میکند و عنصر اول آن را به عنوان سریعترین دونده برمی‌گرداند.

//

```
print('\nFastest runner:\n',find_fastest(runners_list))
print('runner with most_distance:\n',max(runners_list,key=lambda i:i['distance']))
```

چاپ سریعترین دونده و همچنین دونده با بیشترین مسافت طی شده.

//

```
for name in runners_list:
   if 10<=name['distance']<21.09775:
      k_10.append(name)
   elif 21.09775<=name['distance']<42.195:
      halfmaraton.append(name)
   elif 42.195<=name['distance']<60:
      maraton.append(name)
   else:ultra.append(name)
```

دسته‌بندی دونده‌ها در گروهای 10K، maraton، halfmaraton و یا ultra.

//

```
print('\nThe 3 fastest of 10k:')
[print(k_10[i]) for i in range(min(3,len(k_10)))]
print ('\nThe 3 fastest of Half_maraton:')
[print(halfmaraton[i]) for i in range(min(3,len(halfmaraton)))]
print ('\nThe 3 fastest of maraton:')
[print(maraton[i]) for i in range(min(3,len(maraton)))]
print ('\nThe 3 fastest of Ultra:')
[print(ultra[i]) for i in range(min(3,len(ultra)))]
```

چاپ سه سریعترین دونده در هر یک از گروه‌های گفته شده.

//

```
for i in range(len(teams_list)):
    team_members=teams_list[i].split(',')
    team_members=[i.strip() for i in team_members]
    team=dict()
    team['name']=teams_list[i]
    sum_speed=0
    total_distance=0
    for j in range(len(team_members)):
        info=next(item for item in runners_list if item["name"]==team_members[j])
        sum_speed=info['speed']+sum_speed
        total_distance=info['distance']+total_distance
        team_members[j]=info
    team['average_speed']=sum_speed/len(team_members)
    team['total_distance']=total_distance
    max_team=find_fastest(team_members)
    team['leader']=max_team['name']
    teams_list[i]=team
```

به طور کلی در این بخش از برنامه یک لیست (teams_list) که طول آن به تعداد تیم ها است ایجاد می‌شود. هر عضو این لیست یک دیکشنری است که هر دیکشنری حاوی اطلاعات مربوط به تیم مورد نظر است (میانگین سرعت، مجموع مسافت طی شده و رهبر آن تیم؛ که رهبر در هر تیم دونده‌ی با بیشترین سرعت است).

team: به ازای هر تیم یک دیکشنری با نام team ایجاد کرده‌ایم. این دیکشنری حاوی اطلاعات آن تیم است(total_distance, average_speed, lrader).

team_members: به ازای هر تیم یک team_members ایجاد کرده‌ایم. این لیست حاوی چند دیکشنری است که هر یک از دیکشنری‌ها حاوی اطلاعات هر عضو آن تیم است(distance, duration, pace, speed).

تابع next برای سرچ کردن یک دیکشنری مورد نظر در لیستی از دیکشنری‌ها استفاده می‌شود. در واقع نام هر دونده را در لیست runners_list جستجو کرده و در صورت وجود، اطلاعات آن دونده را (که یک دیکشنری است)

در info ذخیره می‌کنیم. سپس مقدار سرعت و مسافت آن دونده به ترتیب به sum_speed و total_distance افزوده می شود و در نهایت 'average_speed' و 'total_distance' و 'leader' برای آن تیم محاسبه می شود.

//

```
teams_list=sorting(teams_list,'total_distance')
print('\nAnnounce top 3 teams base on total distance:')
[print(teams_list[i]) for i in range(min(3,len(teams_list)))]
```

مرتب‌سازی teams_list بر اساس total_distance و چاپ سه تیم دارای بیشترین مسافت طی شده.