```
# EJERCICIO 1 [1 punto]

def lee_trending_videos(fichero):
    videos = []
    with open(fichero, encoding='utf-8') as f:
        lector = csv.reader(f, delimiter=";")
        next(lector)
        for id_video, fecha_trending, titulo, canal, categoria, visitas, likes, dislikes
                in lector:
            tupla = Video(id_video,
                        datetime.strptime(fecha_trending, '%d/%M/%Y').date(),
                        titulo, canal, categoria,
                        int(visitas), int(likes), int(dislikes))
            videos.append(tupla)
    return videos


# EJERCICIO 2 [1 punto]

def media_visitas(videos, fecha):
    visitas = [video.visitas for video in videos if fecha == video.fecha_trending]
    media = 0
    if len(visitas) > 0:
        media = sum(visitas) / len(visitas)
    return media


# EJERCICIO 3 [1,5 puntos]

def video_mayor_ratio_likes_dislikes(videos, categoria=None):
    videos = [video for video in videos
        if (categoria == None or categoria == video.categoria) and video.dislikes > 0]

return max(videos, key = lambda video:video.likes / video.dislikes)


# EJERCICIO 4 [1,5 puntos]

#Versión 1: Creando diccionario y lista ordenada
def canales_top(videos, n=3):
    dicc = videos_por_canal(videos)
    return sorted(dicc.items(), key=lambda item:item[1], reverse=True)[:n]

#Versión 2: Creando Counter y usando most_common
def canales_top(videos, n=3):
    dicc = videos_por_canal(videos)
    return dicc.most_common(n)

#Función auxiliar - Versión 1 - Esquema contador con diccionario
def videos_por_canal(videos):
    res = dict()
    for video in videos:
        if video.canal in res:
            res[video.canal] += 1
        else:
            res[video.canal] = 1
    return res
```

```
#Función auxiliar - Versión 2 – Esquema contador con get y valor por defecto
def videos_por_canal(videos):
    res = dict()
    for video in videos:
        res[video.canal] = res.get(video.canal, 0) + 1
    return res

#Función auxiliar - Versión 3 – Esquema con Counter
def videos_por_canal(videos):
    res = Counter(video.canal for video in videos)
    return res


# EJERCICIO 5 [2 puntos]

def video_mas_likeability_por_categoria(videos, k):
    dicc = videos_por_categoria(videos)
    res = dict()
    for categoria, lista_videos in dicc.items():
        tupla_max = max(lista_videos, key=lambda video:likeability(video, k))
        res[categoria] = tupla_max.id_video
    return res

#versión por compresión
def video_mas_likeability_por_categoria(videos, k):
    dicc = videos_por_categoria(videos)
    res = {categoría: max(lista_videos, key=lambda video:likeability(video, k)).id_video
            for categoria, lista_videos in dicc.items()}
   return res

#Función auxiliar – Cálculo del índice likeability
def likeability (video, k):
    return (k * video.likes - video.dislikes) / (k * video.visitas)

#Función auxiliar – Generación de diccionario con las categorías como claves y la lista
#de videos de esa categoría como valores.
def videos_por_categoria(videos):
    res = dict()
    for video in videos:
        if video.categoria in res:
           res[video.categoria].append(video)
        else:
            res[video.categoria] = [video]
    return res


# EJERCICIO 6 [2 puntos]

def incrementos_visitas(videos, canal):
    dias = dias_trending(videos)
    dicc = visitas_por_dia(videos, canal)
    incrementos = []
    for index in range(len(dias) - 1):
        incremento = dicc.get(dias[index+1], 0) - dicc.get(dias[index], 0)
        incrementos.append(incremento)
    return incrementos
```

```
#Función auxiliar - obtención diccionario con claves la fecha
#y con valores el total de visitas de esa fecha.
def visitas_por_dia(videos, canal):
    res = dict()
    for video in videos:
        if canal == video.canal:
            if video.fecha_trending in res:
                res[video.fecha_trending] += video.visitas
            else:
                res[video.fecha_trending] = video.visitas
    return res

#Función auxiliar
def dias_trending(videos):
    conj = {video.fecha_trending for video in videos}
    return sorted(conj)

#Versión con zip y listas con comprensión
def incrementos_visitas(videos, canal):
    dicc = visitas_por_dia(videos, canal)
    fechas = dias_trending(videos)
    return [dicc.get(f2,0) - dicc.get(f1, 0) for f1, f2 in zip(fechas, fechas[1:])]
```

 # EJERCICIO 7 [1 punto]

```
def test_lee_trending_videos(videos):
    print ("Leídos ...", len(VIDEOS), "vídeos")
    print("Los tres primeros vídeos son:", videos[:3])
    print("Los tres últimos vídeos son:", videos[-2:])

def test_media_visitas(videos):
    fecha_str = '11/1/2017'
    fecha = datetime.strptime(fecha_str, '%d/%M/%Y').date()
    print ("La media de visitas del día", fecha_str, "es", media_visitas(videos, fecha))
    fecha_str = '11/1/2000'
    fecha = datetime.strptime(fecha_str, '%d/%M/%Y').date()
    print ("La media de visitas del día", fecha_str, "es", media_visitas(videos, fecha))

def test_video_mayor_ratio_likes_dislikes(videos):
    print ("El vídeo con mayor ratio de todos es:")
    print(video_mayor_ratio_likes_dislikes(videos))
    print ("El vídeo con mayor ratio de la categoría Education es:")
    print(video_mayor_ratio_likes_dislikes(videos, 'Education'))

def test_canales_top(videos):
    print ("El top-3 de canales es:")
    print(canales_top(videos))
    print ("El top-5 de canales es:")
    print(canales_top(videos, 5))

def test_video_mas_likeability_por_categoria(videos):
    k = 20
```

```
    print("Vídeo con más likeability por categoría con constante", k)
    dicc = video_mas_likeability_por_categoria(videos, k)
    for categoria, id_video in dicc.items():
        print(categoria, '-->', id_video)

def test_incrementos_visitas(videos):
    canal = 'Exatlón'
    print("Incrementos de visitas para el canal", canal)
    print(incrementos_visitas(videos, canal))
    canal = 'Mr. Tops'
    print("Incrementos de visitas para el canal", canal)
    print(incrementos_visitas(videos, canal))
```