Ejercicio 1

```
DELIM = ','
DELIM_AUX = ':'

def leer_diccionario_generos(nombre_fichero_auxiliar):
    result = dict()
    with open(nombre_fichero_auxiliar, encoding='utf-8') as archivo:
        lector_csv = csv.reader(archivo, delimiter=DELIM_AUX)
        next(lector_csv)
        for id, data in lector_csv:
            key = id.strip()
            value = genera_conjunto(data.strip())
            if not key in result:
                result[key] = value
            else:
                result[key].update(value)
    return result

def genera_conjunto(datos):
    return {name.strip() for name in datos.split(DELIM)}


def leer_peliculas(nombre_archivo_peliculas, nombre_archivo_generos):
    lista_peliculas = []
    generos = leer_diccionario_generos(nombre_archivo_generos)
    with open(nombre_archivo_peliculas, encoding='utf-8') as archivo:
        lector_csv = csv.reader(archivo)
        next(lector_csv)

        for linea in lector_csv:
            id, title, original_language, release_date, vote_average, \
                    popularity, adult = linea
            release_date = datetime.strptime(release_date, '%Y-%m-%d').date()
            vote_average = float(vote_average)
            popularity = int(popularity)
            adult = parse_boolean(adult)
            genres = generos[id.strip()]

            pelicula = Pelicula(id.strip(), title.strip(), original_language.strip(),
                    release_date, vote_average, popularity, adult, genres)
            lista_peliculas.append(pelicula)

    return lista_peliculas
```

```python
def parse_boolean(str_logico):
    res = None
    str_logico = str_logico.strip().lower()
    if str_logico == "true":
        res = True
    elif str_logico == "false":
        res = False
    return res
```

Ejercicio 2

```python
def genero_mas_frecuente(lista_peliculas):
    result = []
    for pelicula in lista_peliculas:
        result += list(pelicula.genres)

    return Counter(result).most_common(1)[0]
```

Alternativa:

```python
def genero_mas_frecuente(lista_peliculas):
    dicc = contador_por_genero(lista_peliculas)
    return max(dicc.items(), key=lambda it:it[1])

def contador_por_genero(lista_peliculas):
    #Creando un diccionario
    res = dict()
    for pelicula in lista_peliculas:
        for genero in pelicula.genres:
            if genero in res:
                res[genero] += 1
            else:
                res[genero] = 1
    return res

def contador_por_genero(lista_peliculas):
    # Alternativa creando un diccionario con defaultdic
    res = defaultdict(int)
    for pelicula in lista_peliculas:
        for genero in pelicula.genres:
            res[genero] += 1
    return res

def contador_por_genero(lista_peliculas):
    # Alternativa creando un diccionario con Counter
    return Counter(genero for pelicula in lista_peliculas \
        for genero in pelicula.genres)
```

Ejercicio 3

```
def mejor_valorada_por_idioma(lista_peliculas):
    dicc = agrupar_por_idioma(lista_peliculas)
    res = dict()
    return {idioma:max(lista_peliculas, \
                       key=lambda pel:(pel.popularity, pel.vote_average)) \
                      for idioma, lista_peliculas in dicc.items()}

def agrupar_por_idioma(lista_peliculas):
    # Versión con dict
    res = dict()
    for pelicula in lista_peliculas:
        idioma = pelicula.original_language
        if idioma in res:
            res[idioma].append(pelicula)
        else:
            res[idioma]=[pelicula]
    return res

def agrupar_por_idioma(lista_peliculas):
    # Versión con defaultdict
    res = defaultdict(list)
    for pelicula in lista_peliculas:
        idioma = pelicula.original_language
        res[idioma].append(pelicula)
    return res
```

Alternativa:

```
def mejor_valorada_por_idioma(lista_peliculas):
    result = defaultdict(lambda: None)

    for pelicula in lista_peliculas:
        key = pelicula.original_language

        if result[key] is None:
            result[key] = pelicula
        else:
            pelicula_actual = result[key]
            if pelicula.popularity > pelicula_actual.popularity
                  or pelicula.popularity == pelicula_actual.popularity
                  and pelicula.vote_average > pelicula_actual.vote_average:
                result[key] = pelicula

    return result
```

Ejercicio 4

```
def media_calificaciones(lista_peliculas, generos):
    result = 0.0
    num_pel = 0
    total_calificaciones = 0.0
    for pelicula in lista_peliculas:
        if generos.issubset(pelicula.genres):
            num_pel += 1
            total_calificaciones += pelicula.vote_average

    if num_pel >= 0:
        result = total_calificaciones / num_pel

    return result
```

Alternativa 1:

```
def media_calificaciones(lista_peliculas, generos):
    result = 0.0
    return statistics.mean(pelicula.vote_average for pelicula \
        in lista_peliculas if generos.issubset(pelicula.genres))
```

Alternativa 2:

```
def media_calificaciones(lista_peliculas, generos):
    result = 0.0
    votos_peliculas_filtradas = [pelicula.vote_average for pelicula \
        in lista_peliculas if generos.issubset(pelicula.genres)]

    if len(votos_peliculas_filtradas):
        total_calificaciones = sum(votos_peliculas_filtradas)
        result = total_calificaciones / len(votos_peliculas_filtradas)

    return result
```

Ejercicio 5

```
def top_n_por_genero(lista_peliculas, n):

    peliculas_por_genero = agrupar_peliculas_por_genero(lista_peliculas)

    return {genero: sorted(peliculas, key=lambda pelicula: pelicula.vote_average, \
      reverse=True)[:n] for genero, peliculas in peliculas_por_genero.items()}
```

```python
def agrupar_peliculas_por_genero(lista_peliculas):
    # Versión con dict
    res = dict()

    for pelicula in lista_peliculas:
        for genero in pelicula.genres:
            if genero in res:
                res[genero].append(pelicula)
            else:
                res[genero] = [pelicula]
    return res

def agrupar_peliculas_por_genero(lista_peliculas):
    # Versión con defaultdict
    res = defaultdict(list)

    for pelicula in lista_peliculas:
        for genero in pelicula.genres:
            res[genero].append(pelicula)
    return res
```

Ejercicio 6

```python
def test_lectura(lista):
    print(len(lista))
    print('Primera:', lista[0])
    print('Última:', lista[-1])

def test_genero_mas_frecuente(lista):
    resultado = genero_mas_frecuente(lista)
    print('El género más frecuente es: ', resultado)

def test_mejor_valorada_por_idioma(lista):
    resultado = mejor_valorada_por_idioma(lista)
    print('Mejor en español (es): ', resultado['es'])

def test_media_calificaciones(lista_peliculas, generos_especificos):
    resultado = media_calificaciones(lista_peliculas, generos_especificos)
    print(resultado)

def test_top_n_por_genero(lista_peliculas, top_n):
    resultado = top_n_por_genero(lista_peliculas, top_n)
    print('Top ', top_n, ' Fake:', resultado['Fake'])
    print('Top ', top_n, ' Action:', resultado['Action'])
```

```python
if __name__=='__main__':
    PELICULAS = leer_peliculas('data/movies_fp.csv', 'data/movies_fp_genres.csv')
    print("----------------------------------------")
    print("TESTING EJERCICIO 1")
    test_lectura(PELICULAS)
    print("----------------------------------------")
    print("TESTING EJERCICIO 2")
    test_genero_mas_frecuente(PELICULAS)
    print("----------------------------------------")
    print("TESTING EJERCICIO 3")
    test_mejor_valorada_por_idioma(PELICULAS)
    print("----------------------------------------")
    print("TESTING EJERCICIO 4")
    print("'Action', 'Adventure', 'Fake': ")
    test_media_calificaciones(PELICULAS,{'Action', 'Adventure', 'Fake'})
    print("'Action', 'Thriller':")
    test_media_calificaciones(PELICULAS, {'Action', 'Thriller'})
    print("----------------------------------------")
    print("TESTING EJERCICIO 5")
    test_top_n_por_genero(PELICULAS, 2)
    print("----------------------------------------")
```