1. **lee\_trending\_videos**: recibe el nombre de un fichero y devuelve una lista de tuplas de tipo Video conteniendo todos los datos almacenados en el fichero. (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

1. **media\_visitas:** recibe una lista de tuplas de tipo Video y una fecha. Devuelve la media de visitas de una fecha dada. Si para esa fecha no hay registros, la función devuelve cero. (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

1. **video\_mayor\_ratio\_likes\_dislikes**: recibe una lista de tuplas de tipo Video y una categoría con valor por defecto None. Devuelve la tupla de tipo Video de la categoría dada como parámetro que ha tenido una mayor ratio *likes*/*dislikes*. Si la categoría toma el valor None, se devolverá la tupla de tipo Video con mayor ratio *likes*/*dislikes* de todas. La ratio *likes*/*dislikes* se calcula como el cociente entre el número de *likes* y el número de *dislikes*. Tenga en cuenta que puede haber vídeos que no hayan recibido *dislikes*, y que no deben ser tenidos en cuenta en el cálculo del máximo. (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)

1. **canales\_top**: recibe una lista de tuplas de tipo Video y un número entero n, con valor por defecto 3. Devuelve una lista de tuplas (canal, num\_videos\_trending) con los n canales que tienen más videos *trending*. Cada tupla contiene el nombre del canal y el número de videos *trending* de ese canal, teniendo en cuenta que si un video es *trending* durante d días contará d veces para el canal. La lista estará ordenada de mayor a menor número de vídeos *trending*. (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\_top2(videos, n=3):

dicc = videos\_por\_canal3(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\_canal2(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\_canal3(videos):

res = Counter(video.canal for video in videos)

return res

1. **video\_mas\_likeability\_por\_categoria**: recibe una lista de tuplas de tipo Video y un número entero k, que representa una constante con valor por defecto 20. Devuelve un diccionario que asocia las categorías (claves), con el vídeo de mayor índice *likeability* de esa categoría. El índice *likeability* se calcula según la fórmula que se indica a continuación. (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

#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

1. **incrementos\_visitas**: recibe una lista de tuplas de tipo Video y un canal. Devuelve una lista con el incremento (o decremento) del total de visitas diarias de los vídeos *trending* de un día con respecto al día anterior para el rango de fechas en que hay mediciones. Note que puede haber días para el que se hayan tomado mediciones en los que no aparezca ningún video de un canal concreto, ya que los videos de ese canal no han sido tendencia ese día. Sin embargo, esos días habrá que tenerlos en cuenta en el cálculo de los incrementos. Por ejemplo, el canal Mr. Tops no tiene ningún vídeo *trending* el día 10/01/2017, y el día 11/01/2017 ha obtenido 200487 visitas en videos que son tendencia. En este caso, en la lista resultado debe aparecer un incremento de 200487, aunque no se haya registrado ningún dato de ese canal el día 10/01/2017. (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\_visitas2(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:])]

1. Complete el código de las funciones de test del módulo test\_youtube.py para probar las funciones anteriores, teniendo en cuenta que la salida por consola de la ejecución de los test debe ser la siguiente. (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))

Los test deben producir el siguiente resultado:

EJERCICIO 1

Leidos ... 9064 vídeos

Los tres primeros vídeos son: [Video(id\_video='SbOwzAl9ZfQ', fecha\_trending=datetime.date(2017, 11, 14), titulo='Capítulo 12 | MasterChef 2017', canal='MasterChef 2017', categoria='Entertainment', visitas=310130, likes=4182, dislikes=361), Video(id\_video='klOV6Xh-DnI', fecha\_trending=datetime.date(2017, 11, 14), titulo='ALEXA EX-INTEGRANTE DEL GRUPO TIMBIRICHE RENUNCIA A ¨La Voz Mexico 7¨TELEVISA 11/11/2017', canal='Micky Contreras Martinez', categoria='People & Blogs', visitas=104972, likes=271, dislikes=174), Video(id\_video='6L2ZF7Qzsbk', fecha\_trending=datetime.date(2017, 11, 14), titulo='LOUIS CKAGÓ - EL PULSO DE LA REPÚBLICA', canal='El Pulso De La República', categoria='News & Politics', visitas=136064, likes=10105, dislikes=266)]

Los tres últimos vídeos son: [Video(id\_video='5hrotwT9Rkg', fecha\_trending=datetime.date(2017, 12, 31), titulo='Nuestros invitados se atrevieron a todo en Verdad o Reto | Hoy', canal='Hoy', categoria='Entertainment', visitas=22661, likes=299, dislikes=27), Video(id\_video='EmUhNptc6a8', fecha\_trending=datetime.date(2017, 12, 31), titulo='Sin Tu Mirada | Marina es hija de Prudencia y de Luis', canal='ERICK NOVELAS', categoria='People & Blogs', visitas=149222, likes=944, dislikes=68), Video(id\_video='o5XXq4ZUU9w', fecha\_trending=datetime.date(2017, 12, 31), titulo='Dragon Ball Super Episódio 122 - Vegeta Supera Seus Limites Legendado PT/BR Fan Animation', canal='Animebr', categoria='Film & Animation', visitas=202774, likes=2266, dislikes=1380)]

EJERCICIO 2

La media de visitas del día 15/11/2017 es 190649.58549222798

La media de visitas del día 11/1/2000 es 0

EJERCICIO 3

El vídeo con mayor ratio de todos es:

Video(id\_video='JmQUmUHq2k8', fecha\_trending=datetime.date(2017, 12, 11), titulo='[BT21] Meet BT21', canal='BT21', categoria='People & Blogs', visitas=80550, likes=22637, dislikes=7)

El vídeo con mayor ratio de la categoría Education es:

Video(id\_video='F6oX3ozIqfo', fecha\_trending=datetime.date(2017, 12, 15), titulo='CORTE GRAPADO - Makeup FX', canal='SARA G', categoria='Education', visitas=16225, likes=2118, dislikes=5)

EJERCICIO 4

El top-3 de canales es:

[('Exatlón', 48), ('Cracks', 42), ('Badabun', 41)]

El top-5 de canales es:

[('Exatlón', 48), ('Cracks', 42), ('Badabun', 41), ('Troom Troom Es', 40), ('Enamorándonos', 40)]

EJERCICIO 5

Vídeo con más likeability por categoría con constante 20

Entertainment --> ODjB1sL\_D2E

People & Blogs --> OUKUYrWOrtU

News & Politics --> iS\_pOZel65c

Howto & Style --> ikP-ovgZ\_N8

Music --> qGohZu-Bh4A

Comedy --> 5cdpF-j0R4Q

Sports --> ubyXi8c097Q

Autos & Vehicles --> XhLVE-IEXtA

Film & Animation --> Xew3Ci5IRdY

Nonprofits & Activism --> aHmmsOQMBfE

Education --> 9dm10rgFKmA

Science & Technology --> JBZx03342eM

Gaming --> s6AQL58y9vE

Pets & Animals --> 1gPXAR-w-Nw

Travel & Events --> lVL8EVbbHiA

Shows --> tGPP0F6uqOE

EJERCICIO 6

Incrementos de visitas para el canal Exatlón

[-325232, 16670, -27297, 203424, -195337, 298667, -256420, -21585, 11724, 62358, 328537, -320609, 392661, 153879, -604864, 333205, 61758, 69525, -479489, 403893, 165957, -569500, 338264, -54478, -266704, -30019, 443346, -384826, -21858, -19084, 286399, -210520, -57257, 494904, -523057, -39327, 114209, -50784, 555004, -597681, 595337, -522645, 462916, -458246, -56157, 224648, -184478]

Incrementos de visitas para el canal Mr. Tops

[-231503, 0, 0, 0, 0, 0, 310425, -310425, 184459, -184459, 0, 0, 0, 132521, -132521, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 200487, -53308, -147179, 0, 80380, -80380, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]