18_Practica_Obligatoria_Pandas_I

November 28, 2023

```
edad_gonzalo = 30
edad_gonzalo_en_estreno = edad_gonzalo - (2023 - serie_lanzamiento)
print(edad_gonzalo_en_estreno)
```

✓ 0.0s

```
Toy Story 4            22
Los Incríbles 2        25
Buscando a Dory        23
Toy Story 3            17
Caco                   24
Inside Out             22
Monsters University    20
Up                     13
Name: lanzamiento, dtype: int64
```

0.1 PRACTICA OBLIGATORIA: Iniciación a Pandas

- La práctica obligatoria de esta unidad consiste en varios ejercicios de programación libre a completar. Descarga este notebook en tu ordenador y trabaja en local. Ten en cuenta que tendrás que descar los directorios de imágenes y datos adicionales, si los hubiera.
- Recuerda que debes subirla a tu repositorio personal antes de la sesión en vivo para que puntúe adecuadamente.

- Recuerda también que no es necesario que esté perfecta, sólo es necesario que se vea el esfuerzo.
- Esta práctica se resolverá en la sesión en vivo correspondiente y la solución se publicará en el repo del curso.

1 #1 Series

Importa pandas y numpy de la forma que hemos visto hasta ahora:

[1]:
```
import numpy as np
import pandas as pd
```

1. A partir de las listas siguientes, crea tres series que tengan como índices los títulos de las películas. Guárdalas cada una en una variable, que usarás a lo largo de la práctica.m

[2]:
```
titulos = ["Toy Story 4", "Los Incríbles 2", "Buscando a Dory", "Toy Story 3",␣
 ↪"Caco", "Inside Out", "Monsters University", "Up"]
lanzamiento = [2015, 2018, 2016, 2010, 2017, 2015, 2013, 2006]
recaudaciones = [1073, 1242, 1029, 1067, 807, 857, 744, 735]  # En millones de␣
 ↪dólares
espectadores = [74.91, 93.42, 76.72, 81.35, 62.75, 68.27, 54.74, 54.34] # En␣
 ↪millones, estimación hecha por aficionados
```

[ ]:
```
```

[3]:
```
# creo un diccionario por si me hiciera falta mas adelante, con todo (PRUEBA␣
 ↪POR SI ME HACIA FALTA MAS ADELANTE)
dict_completo = {}
for i in range(len(titulos)):
    dict_completo[titulos[i]] = {
        "año_lanzamiento": lanzamiento[i],
        "recaudacion(millones $)": recaudaciones[i],
        "personas(millones habitantes": espectadores[i]
        }
#rint(dict_completo)
```

[16]:
```
##1##
# hago arrays mediante su posicion en la lista y despues le hago su dataframe␣
 ↪con indice por peliculas
serie_1_np = np.array(lanzamiento)
serie_1 = pd.DataFrame(serie_1_np, columns = ["año lan."], index =[titulos])
serie_1
```

[16]:

	año lan.
Toy Story 4	2015
Los Incríbles 2	2018
Buscando a Dory	2016
Toy Story 3	2010
Caco	2017
Inside Out	2015
Monsters University	2013
Up	2006

[17]:
```
# hago arrays mediante su posicion en la lista y despues le hago su dataframe␣
 ↪con indice por peliculas y columna
serie_2_np = np.array(recaudaciones)
serie_2 = pd.DataFrame(serie_2_np, columns = ["millones $)"],index =[titulos])
serie_2
```

[17]:

	millones $)
Toy Story 4	1073
Los Incríbles 2	1242

```
Buscando a Dory             1029
Toy Story 3                 1067
Caco                         807
Inside Out                   857
Monsters University          744
Up                           735
```

[18]:
```
# hago arrays mediante su posicion en la lista y despues le hago su dataframe␣
 ↪con indice por peliculas y columna año
serie_3_np = np.array(espectadores)
serie_3 = pd.DataFrame(serie_3_np, columns = ["millones hab."], index␣
 ↪=[titulos])
serie_3
```

[18]:
```
                     millones hab.
Toy Story 4                  74.91
Los Incríbles 2              93.42
Buscando a Dory              76.72
Toy Story 3                  81.35
Caco                         62.75
Inside Out                   68.27
Monsters University          54.74
Up                           54.34
```

[ ]:
```
```

2. El método sort_values() ordena de forma ascendente una serie. Pruébalo con la serie de recaudaciones. Busca el argumento que te permita hacer la ordenación en orden inverso y crea una nueva serie para las recaudaciones que esté ordenada de mayor recaudación a menor.

[19]:
```
##2##
# extructura de este metodo: DataFrame o Serie.sort_values(by, axis=0,␣
 ↪ascending=True, inplace=False, ignore_index=False)(by: especifica columna/s␣
 ↪que se desae ordenar, pudiendo ser nombre de columnas o listas de nombres,
# axis= si va ordenar a lo largo de filas(0) o columnas(1), ascending=␣
 ↪ascendiente con True y lo contrario con False, inplace: si debe hacer la␣
 ↪ordenacion en el original o debe crear una copia del dataframe,
# Ignore_index: si es True, restablecera los indices despues de la ordenacion )
df_recaudaciones = pd.Series(recaudaciones) # ccreamos la serie Pandas␣
 ↪recaudaciones
recaudaciones_values_asc = df_recaudaciones.sort_values()
recaudaciones_values_des = df_recaudaciones.sort_values(ascending= False)
print(recaudaciones_values_asc)
print("\n")
print(recaudaciones_values_des)
```

```
7     735
6     744
```

```
4     807
5     857
2    1029
3    1067
0    1073
1    1242
dtype: int64


1    1242
0    1073
3    1067
2    1029
5     857
4     807
6     744
7     735
dtype: int64
```

[ ]:

3. Utilizando la serie de recaudaciones obtenida en el apartado anterior, recorrela de forma que muestres para cada película su recaudación y ya de paso su año de lanzamiento y el número de espectadores. (Ojo el que corresponda de forma correcta, no te vale el índice de un enumerate por ejemplo, pero no lo necesitas, recuerda que las Series vienen con su índice explícito incorporado y todas las que hayas creado tienen el mismo). Muestra también el precio medio de la entrada que tuvo cada película.

[20]:
```
# FORMA SIN INDICES (INCORRECTA), YA QUE NO DA LOS INDICES CORRECTAMENTE
#crero los Series de los elementos titulos, lanzamiento, recaudaciones y␣
 ↪espectadores
df_titulos = pd.Series(titulos)
df_lanzamientos= pd.Series(lanzamiento)
df_recaudaciones = pd.Series(recaudaciones)
df_espectadores = pd.Series(espectadores)
# para ordenar los cuatro dataframe usando un metodo similar en numpy pero en␣
 ↪pandas que ordenas los dataframe a lo largo del eje que elijas 0 = filas o 1␣
 ↪= columnas
df_mezclado = pd.
 ↪concat([df_titulos,df_lanzamientos,df_recaudaciones,df_espectadores], axis=1)
```

[31]:
```
##3##
# FORMA CON INDICES, CORRECTA PARA VALORAR
#creo una cuarta serie con valores inventados de los tickects de entrada en␣
 ↪dolares; primero pongo la lista normal, la paso a array y despues hago su␣
 ↪dataframe ordando por coumnas y filas
precio= round(df_recaudaciones/df_espectadores, 2)
```

```
serie_4_pn = np.array(precio)
serie_4 = pd.DataFrame(serie_4_pn, columns = ["ticket $"], index =[titulos])
# finalmente creo la tabla con todos los valores
dt_completo= pd.concat([serie_1, serie_2, serie_3, serie_4], axis=1)
dt_completo
```

[31]:

	año lan.	millones $)	millones hab.	ticket $
Toy Story 4	2015	1073	74.91	14.32
Los Incríbles 2	2018	1242	93.42	13.29
Buscando a Dory	2016	1029	76.72	13.41
Toy Story 3	2010	1067	81.35	13.12
Caco	2017	807	62.75	12.86
Inside Out	2015	857	68.27	12.55
Monsters University	2013	744	54.74	13.59
Up	2006	735	54.34	13.53

[ ]:

4. Contesta: (Usando código, of course) 4.1 ¿De que año es la película con menor recaudación? 4.2 ¿Cómo se llama la película con más de 1000 millones de recaudación y menos de 75 millones de espectadores? 4.3 Gonzalo tiene ahora 30 años, ¿con qué años fue a ver cada película, teniendo en cuenta que fue al estreno de todas?

[10]:
```
#(PRUEBA)#
#pelicula año emnor recaudacion
peli_agno_menor_recaudacion = df_lanzamientos.values[7]
#print(peli_agno_menor_recaudacion)
#print("\n")
#película con más de 1000 millones de recaudación y menos de 75 millones de␣
 ↪espectadores
peli_1000_menos_75 = df_titulos.iloc[0:1]
#print(peli_1000_menos_75)
#print("\n")
```

[34]:
```
########FORMA CORRECTA A VALORAR########
##4.1#
# aplico la funcion min para hallar los valores minimos y uso la funion idxmin,␣
 ↪para saber el indice correspondiente al min, o el idxmax para concer el␣
 ↪maximo
minimo_pelicula= dt_completo.min()
print(minimo_pelicula)
print("\n")
##4.2#
# hago el dataframe de titulos para que me de el titulo de la pelicula con␣
 ↪ayuda de otra funcion
titulo_np = np.array(titulos)
dt_titulo = pd.DataFrame(titulo_np, columns= ["Peliculas"])
```

```python
# usando la funcion de panda iat(), conseguimos indexar el indice y conseguir␣
 ↪el valor pelicula que le corresponde a los valores minimos (valor =␣
 ↪dataframe.iat[fila, columna])
peli_min=dt_titulo.iat[7, 0]

print(f"la pelicula {peli_min}) fue la que obtuvo valores minimos en la tabla:
 ↪\n{minimo_pelicula}")
print("\n")
##4.3#
#Gonzalo tiene ahora 30 años, ¿con qué años fue a ver cada película, teniendo␣
 ↪en cuenta que fue al estreno de todas?
#hago una funcion para calcular el año de nacimiento de gonzalo y restarlo a␣
 ↪los años de lanzamiento
def ver_pelis(edad, lanzamiento):
    agno_nacimiento = 2023 - edad
    df_lanzamientos= pd.Series(lanzamiento)
    #df_lanzamientos_ord = df_lanzamientos.sort_values()
    for i in range(8):
        resultado = df_lanzamientos.values[:] - agno_nacimiento
    return resultado
#imprimo por pantalla
print(f"Las peliculas de la lista: {(titulos)}\n , lanzadas en los años␣
 ↪{lanzamiento}\nfueron vistas con las siguientes edades por Gonzalo,teniendo␣
 ↪en cuenta tiene 30 años: {ver_pelis(30,lanzamiento)}")
print("\n")
```

```
año lan.          2006.00
millones $)        735.00
millones hab.       54.34
ticket $            12.55
dtype: float64


la pelicula Up) fue la que obtuvo valores minimos en la tabla:
año lan.          2006.00
millones $)        735.00
millones hab.       54.34
ticket $            12.55
dtype: float64


Las peliculas de la lista: ['Toy Story 4', 'Los Incríbles 2', 'Buscando a Dory',
'Toy Story 3', 'Caco', 'Inside Out', 'Monsters University', 'Up']
 , lanzadas en los años [2015, 2018, 2016, 2010, 2017, 2015, 2013, 2006]
fueron vistas con las siguientes edades por Gonzalo,teniendo en cuenta tiene 30
```

```
años: [22 25 23 17 24 22 20 13]
```

```
edad_gonzalo = 30
edad_gonzalo_en_estreno = edad_gonzalo - (2023 - serie_lanzamiento)
print(edad_gonzalo_en_estreno)
```

✓ 0.0s

```
Toy Story 4            22
Los Incríbles 2        25
Buscando a Dory        23
Toy Story 3            17
Caco                   24
Inside Out             22
Monsters University    20
Up                     13
Name: lanzamiento, dtype: int64
```

[ ]:

5. Corrije los siguientes datos erróneos (pero no repitas los apartados anteriores) en las series (no en las listas iniciales) de forma que tus variables contengan los valores correctos:

- Toy Story 4 es de y Up de 2009
- La recaudación de Monsters University fue de 754 millones.
- Hay 2 nombres que han sufrido el efecto del conocido "error Jaime", corrígelo en todas las series (recuerda que los indices son inmutables, tendrás que hacer algo de código)

[45]:
```
##5##
#5.1#cambiamos mediante un cilo for los años,
# el elemento pop solo borrara el 1 elemnto 2015, por lo que no afectara al␣
 ↪segundo
for i in lanzamiento:
    if i == 2015:
        lanzamiento.pop(0)
        lanzamiento.insert(0, 2009)# aqui uso insert para colocar el nuevo␣
 ↪elelento en el lugar del otro
    elif i == 2006:
        lanzamiento.pop(-1)
        lanzamiento.append(2009)# aqui us append porque ira al final de la lista
#impre la  lista correcta de años
print(lanzamiento)
print("\n")
#5.2 # mediante un pop elimino la cantidad erronea e inserto en su lugar la␣
 ↪cantidad corecta
recaudaciones.pop(6)
```

```
recaudaciones.insert(-1,754)
#imprimo recaudaciones con las cantidades correctas
print(recaudaciones)
print("\n")
#5.3
#1 mediante cilo for elimino de la lista titulos  los 2 elementos e inserto en␣
 ↪las mismas posiciones los elementos corectos, al igual que hago con las␣
 ↪series 1 y 2
for i in titulos:
    if i == "Los Incríbles 2":
        titulos.pop(1)
        titulos.insert(1,"Los Increíbles 2")
        serie_1.pop(1)
        serie_1.append("Los Increíbles 2")
    elif i == "Caco":
        titulos.pop(4)
        titulos.insert(4, "Coco")
        serie_2.pop(2)
        serie_2.append("Coco")
#imprimo resultados con los datos correctos
print(titulos)
print(serie_1)
print(serie_2)
```

```
[2009, 2018, 2016, 2010, 2017, 2015, 2013, 2009]


[1073, 1242, 1029, 1067, 807, 857, 754, 735]


['Toy Story 4', 'Los Increíbles 2', 'Buscando a Dory', 'Toy Story 3', 'Coco',
'Inside Out', 'Monsters University', 'Up']
                     año lan.
Toy Story 4              2015
Los Incríbles 2          2018
Buscando a Dory          2016
Toy Story 3              2010
Caco                     2017
Inside Out               2015
Monsters University      2013
Up                       2006
                     millones $)
Toy Story 4                1073
Los Incríbles 2            1242
Buscando a Dory            1029
Toy Story 3                1067
Caco                        807
Inside Out                  857
```

```
Monsters University          744
Up                           735
```

[ ]:

[ ]:

2 #2 DataFrame

1. Crea un `DataFrame` con las series anteriores, ya corregidas, como columnas (usando la serie ordenada de recaudaciones) y que cómo indice de filas tenga las películas.

[37]:
```
#SIN USAR SORT_VALUES ####
# vamos a ordenar en orden descendente los años de lanzamiento , lo epectadores␣
 ↪y el valor medio de los teckets de entrada
#años
lanzamientos_values_des_= df_lanzamientos.sort_values(ascending= False)
#print(ordenar_values_des_agno)
#millones $
#print(recaudaciones_values_des)
#hab
espectadores_values_des= df_espectadores.sort_values(ascending= False)
#print(ordenar_values_des_hab)
# tickets
df_precio = pd.Series(precio)
precio_values_des= df_precio.sort_values(ascending= False)
#print(ordenar_values_des_precio)
#creo una lista de titulos conforme a la ordenacion descendiente
titulos_ord = ['Los Increíbles 2', 'Coco', 'Buscando a Dory', 'Inside Out',␣
 ↪'Monsters University', 'Toy Story 3', 'Toy Story 4', 'Up']
titulos_ord_np = np.array(titulos_ord)
dt_tirulos_ord = pd.DataFrame(titulos_ord_np)
# creo un nuevo dataframe de la serie 1 con la lista ordenada
serie_1_np_ord = np.array(lanzamientos_values_des_)
serie_1_ord =pd.DataFrame(serie_1_np_ord, columns = ["año lan."], index␣
 ↪=[titulos_ord])

# creo un nuevo dataframe de la serie 2 con la lista ordenada
serie_2_np_ord = np.array(recaudaciones_values_des)
serie_2_ord = pd.DataFrame(serie_2_np_ord, columns = ["millones $"], index␣
 ↪=[titulos_ord])

# creo un nuevo dataframe de la serie 3 con la lista ordenada
serie_3_np_ord = np.array(espectadores_values_des)
serie_3_ord = pd.DataFrame(serie_3_np_ord, columns = ["millones hab."], index␣
 ↪=[titulos_ord])
```

```
# creo un nuevo dataframe de la serie 3 con la lista ordenada
serie_4_np_ord = np.array(precio_values_des)
serie_4_ord = pd.DataFrame(serie_4_np_ord, columns = ["Tickets $"], index␣
 ↪=[titulos_ord])

#creo el conjunto las 3 series ordenadas
dt_conjunto_3_series = pd.concat([serie_1_ord, serie_2_ord, serie_3_ord,␣
 ↪serie_4_ord], axis=1)
dt_conjunto_3_series
```

[37]:

	año lan.	millones $	millones hab.	Tickets $
Los Increíbles 2	2018	1242	93.42	14.32
Coco	2017	1073	81.35	13.59
Buscando a Dory	2016	1067	76.72	13.53
Inside Out	2015	1029	74.91	13.41
Monsters University	2015	857	68.27	13.29
Toy Story 3	2013	807	62.75	13.12
Toy Story 4	2010	744	54.74	12.86
Up	2006	735	54.34	12.55

[36]:
```
#USANDO SORT_VALUES#####
precio= round(df_recaudaciones/df_espectadores, 2)
serie_4_pn = np.array(precio)
serie_4 = pd.DataFrame(serie_4_pn, columns = ["ticket $"], index =[titulos])
# finalmente creo la tabla con todos los valores
dt_completo= pd.concat([serie_1, serie_2, serie_3, serie_4], axis=1)
peliculas_completo= dt_completo.sort_values(by=[ "año lan.", "millones $)",␣
 ↪"millones hab.", "ticket $"], axis =0, ascending = True)

peliculas_completo
```

[36]:

	año lan.	millones $)	millones hab.	ticket $
Up	2006	735	54.34	13.53
Toy Story 3	2010	1067	81.35	13.12
Monsters University	2013	744	54.74	13.59
Inside Out	2015	857	68.27	12.55
Toy Story 4	2015	1073	74.91	14.32
Buscando a Dory	2016	1029	76.72	13.41
Caco	2017	807	62.75	12.86
Los Incríbles 2	2018	1242	93.42	13.29

[ ]:
```

```

2. Muestra los datos de las películas que tengan más de 10 años.

[44]:
```
# Crear un diccionario con los con los titulos y con los años de lanzamiento
dict_titulo_año= {
```

```
    'Película': ['Toy Story 4', 'Los Increíbles 2', 'Buscando a Dory', 'Toy␣
 ↪Story 3', 'Coco', 'Inside Out', 'Monsters University', 'Up'],
    'Año de Lanzamiento': [2009, 2018, 2016, 2010, 2017, 2015, 2013, 2009]
}
#creo un nuevo dataframe con ambos valores
dt_decada = pd.DataFrame(dict_titulo_año)

# año actual
agno_actual = 2023

# accedo al dataframe y a la columna del año delanzamiento, flitrando a␣
 ↪peliculas menores del año actual menos 10
peliculas_mas_una_decada = dt_decada[dt_decada['Año de Lanzamiento'] <␣
 ↪agno_actual - 10]

# Mostrar las películas que cumplen con el criterio
display("Películas con más de una decada son:")
display(peliculas_mas_una_decada)
```

```
'Películas con más de una decada son:'
```

	Película	Año de Lanzamiento
0	Toy Story 4	2009
3	Toy Story 3	2010
7	Up	2009

[ ]:

3. Muestra los datos de las películas que superen los 800 millones de recaudación y los 65 millones de habitantes

[43]:
```
# Crear un diccionario con los títulos, recaudación y número de habitantes
dict_titulo_millones = {
    'Película': ['Toy Story 4', 'Los Increíbles 2', 'Buscando a Dory', 'Toy␣
 ↪Story 3', 'Coco', 'Inside Out', 'Monsters University', 'Up'],
    'millones de recaudacion': [1073, 1242, 1029, 1067, 807, 857, 754, 735],
    'millones de habitantes': [74.91, 93.42, 76.72, 81.35, 62.75, 68.27, 54.74,␣
 ↪54.34]
}

# Crear un DataFrame con el diccionario
dt_titulo_pasta_hab = pd.DataFrame(dict_titulo_millones)

# Accedo al contedo del dataframe, accediendo a las columnas ",illones de␣
 ↪habitantes" y "millones de recaudacion, Filtrar películas con más de 65␣
 ↪millones de habitantes y más de 800 millones de recaudación
```

```
peliculas_mas_65_800 = dt_titulo_pasta_hab[(dt_titulo_pasta_hab["millones de␣
 ↪habitantes"] > 65) & (dt_titulo_pasta_hab["millones de recaudacion"] > 800)]

# accedo al contenido del dataframe, mediante la funcion shape, con el que se␣
 ↪obtene la forma (número de filas y columnas, shape[0] seran las filas y␣
 ↪shape[1] las columnas, y filtro aqui si las filas son mayor que cero
if peliculas_mas_65_800.shape[0] > 0:# si hay filas, revisa su contenido␣
 ↪accediendo y comprobando la condicion establecida anteriormente
    display("Las películas que han recaudado más de 800 millones y han sido␣
 ↪vistas por más de 65 millones de personas son:")
    display(peliculas_mas_65_800)
else:
    display("No hay películas que cumplan con los criterios especificados.")
```

```
'Las películas que han recaudado más de 800 millones y han sido vistas por más␣
 ↪de 65 millones de personas son:'
```

	Película	millones de recaudacion	millones de habitantes
0	Toy Story 4	1073	74.91
1	Los Increíbles 2	1242	93.42
2	Buscando a Dory	1029	76.72
3	Toy Story 3	1067	81.35
5	Inside Out	857	68.27

[ ]:

4. Añade una columna "Ingreso_por_espectador" que contenga eso... el ingreso por espectador de cada película

[37]:
```
# para realziar el calculo covierto la listas recaudacion y espectadore sa a␣
 ↪arrays para poder dividir ambas listas
recaudaciones_np =np.array(recaudaciones)
espectadores_np =np.array(espectadores)
#resultado de la division de ambas listas arrays
ingresos_por_espectador = recaudaciones_np / espectadores_np
ingresos_por_espectador
```

[37]:
```
array([14.32385529, 13.29479769, 13.41240876, 13.11616472, 12.86055777,
       12.55309799, 13.77420533, 13.52594774])
```

[42]:
```
# Crear un diccionario con los títulos, recaudación y número de habitantes,␣
 ↪añadiendole ingresos por espectador, obtenido en la celda anterior
dict_titulo_millones = {
    'Película': ['Toy Story 4', 'Los Increíbles 2', 'Buscando a Dory', 'Toy␣
 ↪Story 3', 'Coco', 'Inside Out', 'Monsters University', 'Up'],
    'millones de recaudacion': [1073, 1242, 1029, 1067, 807, 857, 754, 735],
    'millones de habitantes': [74.91, 93.42, 76.72, 81.35, 62.75, 68.27, 54.74,␣
 ↪54.34],
```

```
    'ingresos por espectador': [14.32385529, 13.29479769, 13.41240876, 13.
 ↪11616472, 12.86055777,
       12.55309799, 13.77420533, 13.52594774]
}

# Crear un DataFrame con el diccionario
dt_titulo_pasta_hab_precio = pd.DataFrame(dict_titulo_millones)

# Accedo al contedo del dataframe, accediendo a las columnas ",illones de␣
 ↪habitantes" y "millones de recaudacion, Filtrar películas con más de 65␣
 ↪millones de habitantes y más de 800 millones de recaudación
peliculas_mas_65_800_precio =␣
 ↪dt_titulo_pasta_hab_precio[(dt_titulo_pasta_hab_precio["millones de␣
 ↪habitantes"] > 65) & (dt_titulo_pasta_hab_precio["millones de recaudacion"]␣
 ↪> 800) &  (dt_titulo_pasta_hab_precio["ingresos por espectador"])]


# accedo al contenido del dataframe, mediante la funcion shape, con el que se␣
 ↪obtene la forma (número de filas y columnas, shape[0] seran las filas y␣
 ↪shape[1] las columnas, y filtro aqui si las filas son mayor que cero
if peliculas_mas_65_800_precio .shape[0] > 0:# si hay filas, revisa su␣
 ↪contenido accediendo y comprobando la condicion establecida anteriormente
    display("Las películas que han recaudado más de 800 millones y  han sido␣
 ↪vistas por más de 65 millones de personas  son:")
    display(peliculas_mas_65_800_precio)
else:
    display("No hay películas que cumplan con los criterios especificados.")
```

```
'Las películas que han recaudado más de 800 millones y  han sido vistas por más␣
 ↪de 65 millones de personas  son:'
```

	Película	millones de recaudacion	millones de habitantes	\
0	Toy Story 4	1073	74.91	
1	Los Increíbles 2	1242	93.42	
2	Buscando a Dory	1029	76.72	
3	Toy Story 3	1067	81.35	
5	Inside Out	857	68.27	

	ingresos por espectador
0	14.323855
1	13.294798
2	13.412409
3	13.116165
5	12.553098

[ ]:

5. Igual que existe sort_values para `Series`, existe para sort_values para `DataFrame`, solo que

además le puedes indicar más de una columna. Ordena el `DataFrame` para que muestre las películas ordenadas de forma ascendente por Año. Es decir que la variable que contenga el `DataFrame` termine teniendo el dataframe ordenado después de ejecutar el código.

NOTA PERSONAL = extructura dataframe.sort_values: **by**: especifica columna/s que se desae ordenar, pudiendo ser nombre de columnas o listas de nombres, **axis**= si va ordenar a lo largo de filas(0) o columnas(1), **ascending**= ascendiente con True y lo contrario con False, **inplace**: si debe hacer la ordenacion en el original o debe crear una copia del dataframe,**Ignore_index**: si es True, restablecera los indices despues de la ordenacion ##### (by, axis=0, ascending=True, inplace=False, ignore_index=False)

[99]:
```
peliculas_asc_año= dt_decada.sort_values(by=[ "Año de Lanzamiento",␣
↪"Película"], axis =0, ascending = True)

peliculas_asc_año
```

[99]:

	Película	Año de Lanzamiento
0	Toy Story 4	2009
7	Up	2009
3	Toy Story 3	2010
6	Monsters University	2013
5	Inside Out	2015
2	Buscando a Dory	2016
4	Coco	2017
1	Los Increíbles 2	2018

6. Ordena el `DataFrame` ahora para que quede ordenado de mayor a menor recaudación por espectador.

[45]:
```
peliculas_asc_dinero= dt_titulo_pasta_hab_precio.sort_values(by=[ "ingresos por␣
↪espectador", "Película", "millones de recaudacion", "millones de␣
↪habitantes"], axis =0, ascending = False)

peliculas_asc_dinero
```

[45]:

	Película	millones de recaudacion	millones de habitantes	ingresos por espectador
0	Toy Story 4	1073	74.91	14.323855
6	Monsters University	754	54.74	13.774205
7	Up	735	54.34	
2	Buscando a Dory	1029	76.72	
1	Los Increíbles 2	1242	93.42	
3	Toy Story 3	1067	81.35	
4	Coco	807	62.75	
5	Inside Out	857	68.27	

```
7                13.525948
2                13.412409
1                13.294798
3                13.116165
4                12.860558
5                12.553098
```

[ ]:

7. Deshaz las modificaciones de el punto 5 de la parte #1

[97]:
```
eliculas_asc_año= dt_decada.sort_values(by=[ "Año de Lanzamiento", "Película"],␣
↪axis =0, ascending = True)

peliculas_asc_año
```

[97]:
```
              Película  Año de Lanzamiento
0          Toy Story 4                2009
7                   Up                2009
3          Toy Story 3                2010
6  Monsters University                2013
5           Inside Out                2015
2      Buscando a Dory                2016
4                 Coco                2017
1     Los Increíbles 2                2018
```

8. Finalmente, queremos que el título sea una columna más y no el índice, investiga el método set_index y el reset_index para que el título pase a ser una columna y el año el nombre de las filas (o índice del DataFrame)

la funcion **set_index** tiene la siguiente sintaxis : **DF.set_index(keys, drop=True, append=False, inplace=False)**, siendo Keys representa la nueva/s etiqueta/s que se quieren poner como indice, pudiendo ser 1 o varias columnas. drop tiene 2 estados booleanos, siendo su valor predeterminado *True*, realizando que la columna que antes era indice se elimine. Si es *False*, se conservaran como columnas en el DF, agragando como columna a la columna que has puesto de indice(columna repetida). append tiene 2 estados booleanos, siendo su valor predeterminado *True*, por el cual las columnas usadas como indice se conservaran en el DF y se agragaran a las existentes como un indice multiple si ya hubiera un indice. inpalce tiene 2 estados booleanos, siendo su valor predeterminado *False*. Si se establece en True, se aplicarán los cambios directamente al DataFrame, en lugar de crear un nuevo DataFrame con el índice modificado.

Lafuncion **reset_index** iene la siguiente sintaxis : **DF.reset_index(level=None, drop=False, inplace=False)**, siendo level Especifica el nivel del índice para restablecer. Si se omite, todos los niveles del índice se restablecerán. Puede ser una etiqueta única o una lista de etiquetas si el índice es jerárquico. drop tiene 2 estados booleanos, siendo su valor predeterminado *False*, el indice se mantendra como columna del DF, si es *True* lo eliminara. inplace igual que en el set_index

[47]:
```
#hago copia del Df original
dt_completo_e = dt_completo.copy()
```

[52]:
```
cambio_indice_año = dt_completo_e.set_index ("año lan.", drop= True,append =␣
↪True, inplace =False)
cambio_indice_año
```

[52]:

	año lan.	millones $)	millones hab.	ticket $
Toy Story 4	2015	1073	74.91	14.32
Los Incríbles 2	2018	1242	93.42	13.29
Buscando a Dory	2016	1029	76.72	13.41
Toy Story 3	2010	1067	81.35	13.12
Caco	2017	807	62.75	12.86
Inside Out	2015	857	68.27	12.55
Monsters University	2013	744	54.74	13.59
Up	2006	735	54.34	13.53

[50]:
```
cambio_indice_añoR= dt_completo_e.reset_index(names= "Pelicula")
cambio_indice_añoR
```

[50]:

	Pelicula	año lan.	millones $)	millones hab.	ticket $
0	Toy Story 4	2015	1073	74.91	14.32
1	Los Incríbles 2	2018	1242	93.42	13.29
2	Buscando a Dory	2016	1029	76.72	13.41
3	Toy Story 3	2010	1067	81.35	13.12
4	Caco	2017	807	62.75	12.86
5	Inside Out	2015	857	68.27	12.55
6	Monsters University	2013	744	54.74	13.59
7	Up	2006	735	54.34	13.53

[36]:

[ ]:

[ ]:

[ ]: