PRÁCTICA 1 - Gestión de Datos. Estadística Descriptiva Univariante.

Objetivos:

Manejo básico de estadística en Python:

- Gestión de datos
- Manejo de ficheros
- Análisis descriptivo de una y dos variables
- Búsqueda de relación entre variables
- Realización de análisis estadísticos y generación de informes

Instalamos las librerías necesarias

Primero, es necesario instalar las librerías `numpy`, `pandas`, `scipy`, `statsmodels`, `matplotlib` y `seaborn`

Si tenéis instalado `conda`:

`conda install -c conda-forge numpy`

`conda install -c conda-forge pandas`

`conda install -c conda-forge scipy`

`conda install -c conda-forge statsmodels`

`conda install -c conda-forge matplotlib`

`conda install seaborn`

Si vuestra instalación de `Python` es básica:

`pip3 install numpy`

`pip3 install pandas`

`pip3 install scipy`

`pip3 install statsmodels`

`pip3 install matplotlib`

`pip3 install seaborn`

Todos estos comandos los tenéis que ejecutar en vuestra Terminal, para que la instalación sea en vuestro ordenador y no en un ambiente creado.

In [1]:
```python
#Importamos las librerías que vamos a utilizar

import numpy  as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Introducción a la estadística en Python: Conceptos Básicos

1. Creación manual de un fichero de datos (`pd.DataFrame`): Introducción de las variables Estatura (numérica) y Sexo (entero, 1=Hombre, 2=Mujer).

Estatura de 60 personas expresada en metros:

1.62 1.71 1.57 1.61 1.80 1.91 1.58 1.63 1.62 1.70 1.75 1.68 1.54 1.79 1.72 1.68 1.90 1.69 1.73 1.85 1.60 1.60 1.62 1.77 1.71 1.89 1.92 1.65 1.99 2.05 1.41 1.67 1.93 1.55 2.04 1.73 1.80 1.83 1.75 1.66 1.93 1.85 1.84 1.68 1.63 1.75 1.77 1.84 1.85 1.90 2.00 1.83 2.01 1.82 1.65 1.72 1.68 1.73 1.54 1.65

Sexos correspondientes a los datos anteriores:

2 1 2 1 1 2 2 2 2 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 1 1 2 2 2 2 1 1 1 2 2 2 2 2 1 1 1 2 1 2 2 2 1 2 2 2 2 1 2 2 1 1 2 2 2

Con los datos de Sexos, reemplazar el valor númerico por texto (1=Hombre; 2=Mujer) y crear un `pd.DataFrame`

In [2]:
```python
#1a. Creamos una lista de datos para la Estatura y el Sexo correspondiente
estatura = [1.62, 1.71, 1.57, 1.61, 1.80, 1.91, 1.58, 1.63, 1.62, 1.70, 1.75, 1.68, 1.54, 1.79, 1.72, 1.68, 1.90, 1.69, 1.73, 1.85, 1.60, 1.60, 1.62, 1.77, 1.71, 1.89, 1.92, 1.65, 1.99, 2.05, 1.41, 1.67, 1.93,
sexo = [2, 1, 2, 1, 1, 2, 2, 2, 2, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 2, 2, 2, 2, 1, 1, 1, 2, 2, 2, 2, 2, 1, 1, 1, 2, 1, 2, 2, 2, 1, 2, 2, 2, 2, 1, 2, 2, 1, 1, 2, 2, 2]
```

In [3]:
```python
#1b. Reemplazamos el valor numérico por texto
for i in range(len(sexo)):
    if int(sexo[i]) == 1:
        sexo[i] = "Hombre"
    else:
        sexo[i] = "Mujer"

print(sexo)
```

```
['Mujer', 'Hombre', 'Mujer', 'Hombre', 'Hombre', 'Mujer', 'Mujer', 'Mujer', 'Mujer', 'Hombre', 'Hombre', 'Hombre', 'Hombre', 'Mujer', 'Mujer', 'Mujer', 'Mujer', 'Mujer', 'Mujer', 'Mujer', 'Mujer', 'Mujer', 'Muje
r', 'Mujer', 'Mujer', 'Hombre', 'Hombre', 'Mujer', 'Mujer', 'Mujer', 'Mujer', 'Hombre', 'Hombre', 'Hombre', 'Mujer', 'Mujer', 'Mujer', 'Mujer', 'Mujer', 'Hombre', 'Hombre', 'Hombre', 'Mujer', 'Hombre', 'Mujer',
'Mujer', 'Mujer', 'Hombre', 'Mujer', 'Mujer', 'Mujer', 'Mujer', 'Hombre', 'Mujer', 'Mujer', 'Hombre', 'Hombre', 'Mujer', 'Mujer', 'Mujer']
```

In [4]:
```python
#1c. Generamos un DataFrame con pd.DataFrame()
df = pd.DataFrame({'Estatura': estatura,
                   'Sexo': sexo})
df
```

Out[4]:

	Estatura	Sexo
0	1.62	Mujer
1	1.71	Hombre
2	1.57	Mujer
3	1.61	Hombre
4	1.80	Hombre
5	1.91	Mujer
6	1.58	Mujer
7	1.63	Mujer
8	1.62	Mujer
9	1.70	Hombre
10	1.75	Hombre
11	1.68	Hombre
12	1.54	Hombre
13	1.79	Mujer
14	1.72	Mujer
15	1.68	Mujer
16	1.90	Mujer
17	1.69	Mujer
18	1.73	Mujer
19	1.85	Mujer
20	1.60	Mujer
21	1.60	Mujer
22	1.62	Mujer
23	1.77	Mujer
24	1.71	Mujer
25	1.89	Hombre
26	1.92	Hombre
27	1.65	Mujer
28	1.99	Mujer
29	2.05	Mujer
30	1.41	Mujer
31	1.67	Hombre
32	1.93	Hombre
33	1.55	Hombre
34	2.04	Mujer
35	1.73	Mujer
36	1.80	Mujer
37	1.83	Mujer
38	1.75	Mujer
39	1.66	Hombre
40	1.93	Hombre
41	1.85	Hombre
42	1.84	Mujer
43	1.68	Hombre
44	1.63	Mujer
45	1.75	Mujer
46	1.77	Mujer
47	1.84	Hombre
48	1.85	Mujer
49	1.90	Mujer
50	2.00	Mujer
51	1.83	Mujer
52	2.01	Hombre
53	1.82	Mujer
54	1.65	Mujer
55	1.72	Hombre
56	1.68	Hombre
57	1.73	Mujer
58	1.54	Mujer
59	1.65	Mujer

2. Análisis descriptivo / variables cuantitativas: Análisis descriptivo numérico y gráfico.

- Resumen estadístico (media, mediana, desviación estandar, quantiles...) y tabla de frecuencias.
- Hacer un gráfico de dispersión, un gráfico de caja y bigotes, y un histograma

In [5]:
```python
#2a. Realizamos el resumen estadístico. PISTA: hay una función que genera las
#    variables básicas que necesitamos.
df.groupby(['Sexo']).describe()
```

Out[5]:

	Estatura							
	count	mean	std	min	25%	50%	75%	max
Sexo								
Hombre	20.0	1.756	0.133511	1.54	1.6775	1.715	1.8600	2.01
Mujer	40.0	1.745	0.143420	1.41	1.6300	1.730	1.8325	2.05

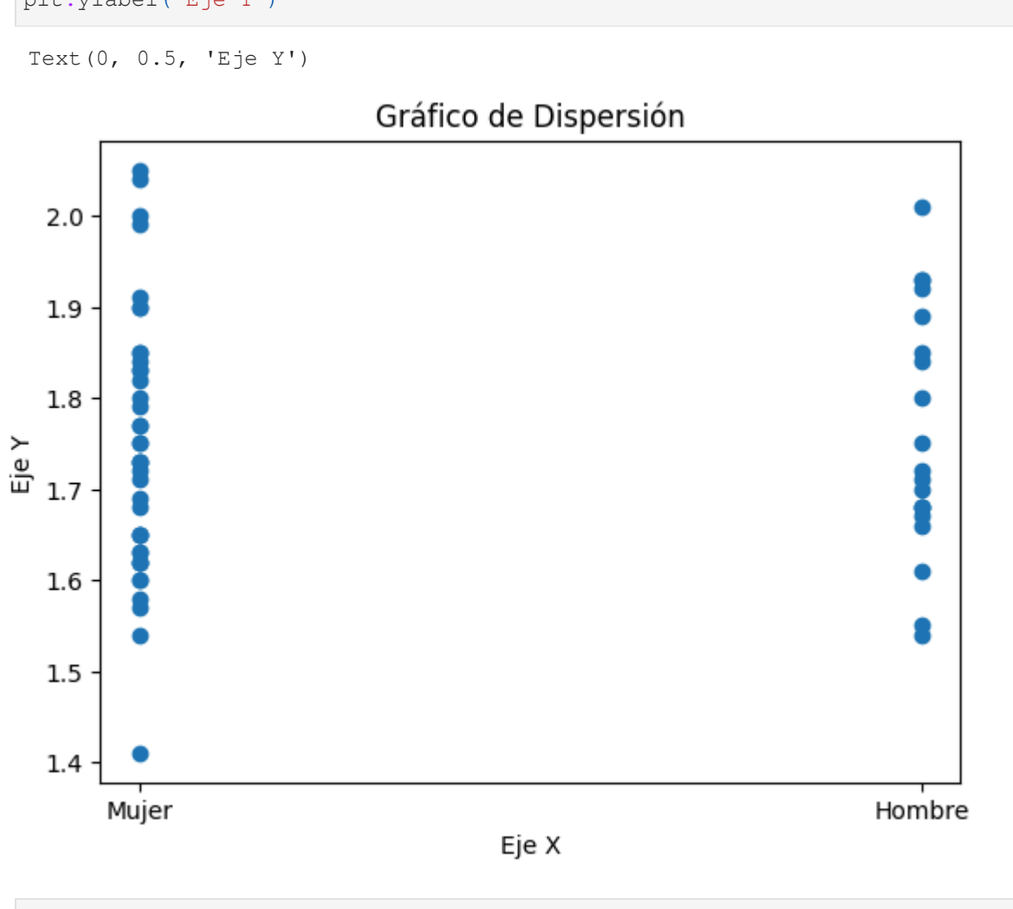
In [6]:
```python
#2b. Generamos una tabla de frecuencias.
df.groupby(['Sexo']).agg(frequency=("Sexo", "count"))
```

Out[6]:

	frequency
Sexo	
Hombre	20
Mujer	40

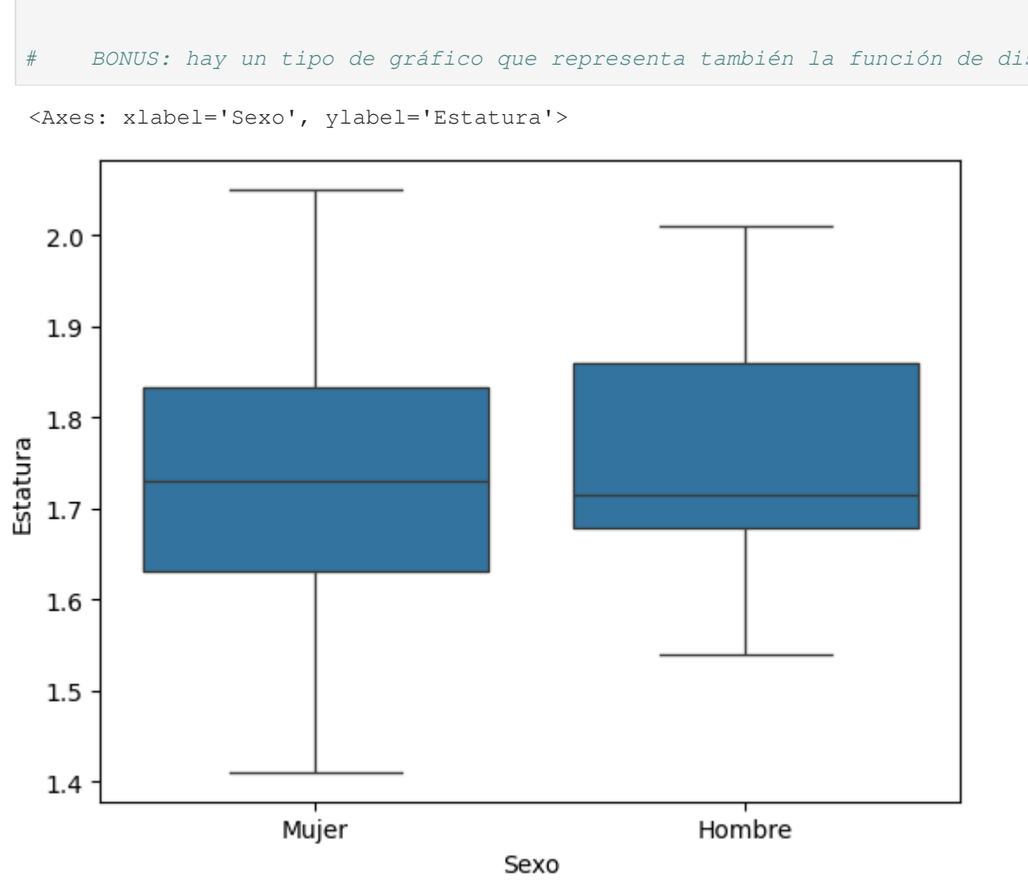
In [7]:
```python
#2c. Generamos un gráfico de dispersión.
plt.scatter(x=df.Sexo, y=df.Estatura)
plt.title('Gráfico de Dispersión')
plt.xlabel('Eje X')
plt.ylabel('Eje Y')
```

Out[7]: `Text(0, 0.5, 'Eje Y')`



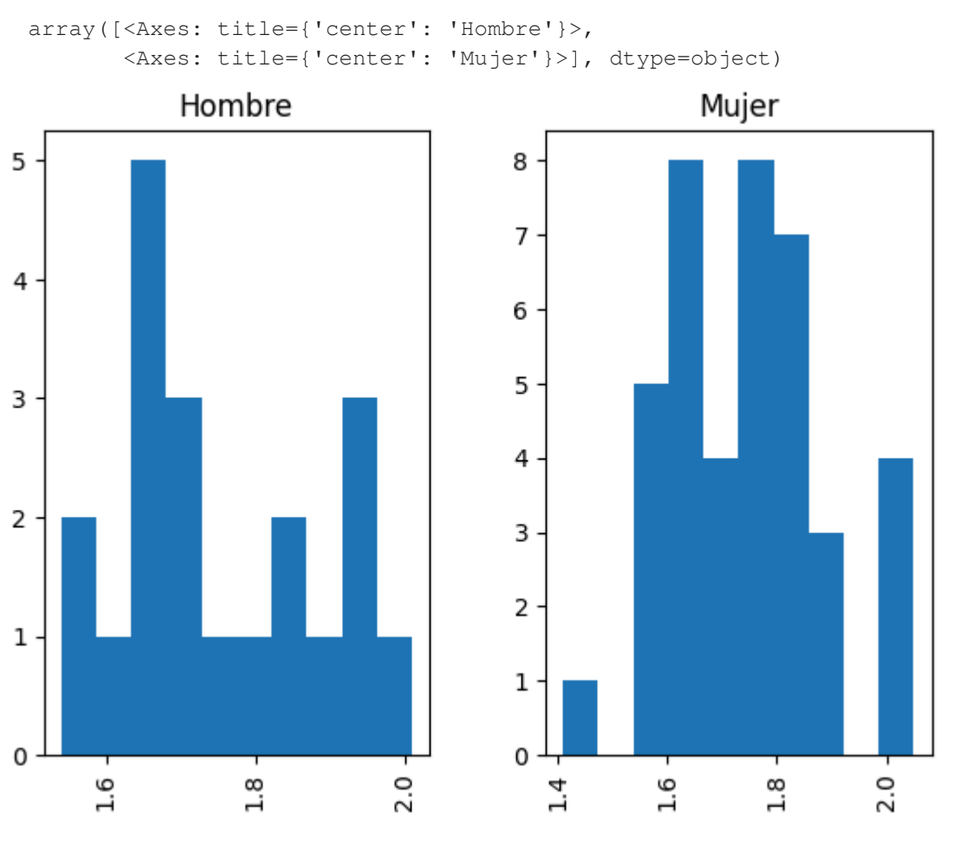
In [8]:
```python
#2d. Generamos un gráfico de cajas y bigotes.259629529252
sns.boxplot(data=df, x='Sexo', y='Estatura')


#    BONUS: hay un tipo de gráfico que representa también la función de distribución. Intenta encontrarlo.
```

Out[8]: `<Axes: xlabel='Sexo', ylabel='Estatura'>`



In [9]:
```python
#2e. Generamos un histograma.
df.hist(by='Sexo')
```

Out[9]:
```
array([<Axes: title={'center': 'Hombre'}>,
       <Axes: title={'center': 'Mujer'}>], dtype=object)
```



3. Análisis descriptivo / variables cualitativa: Análisis descriptivo numérico y gráfico.

- Resumen estadístico (frecuencia) y tabla de frecuencias.
- Hacer un diagrama de barras y un diagrama de sectores

In [10]:
```python
#3a. Realizamos el resumen estadístico. PISTA: hay una función que genera las
#    variables básicas que necesitamos.
print(df['Estatura'].describe())
```

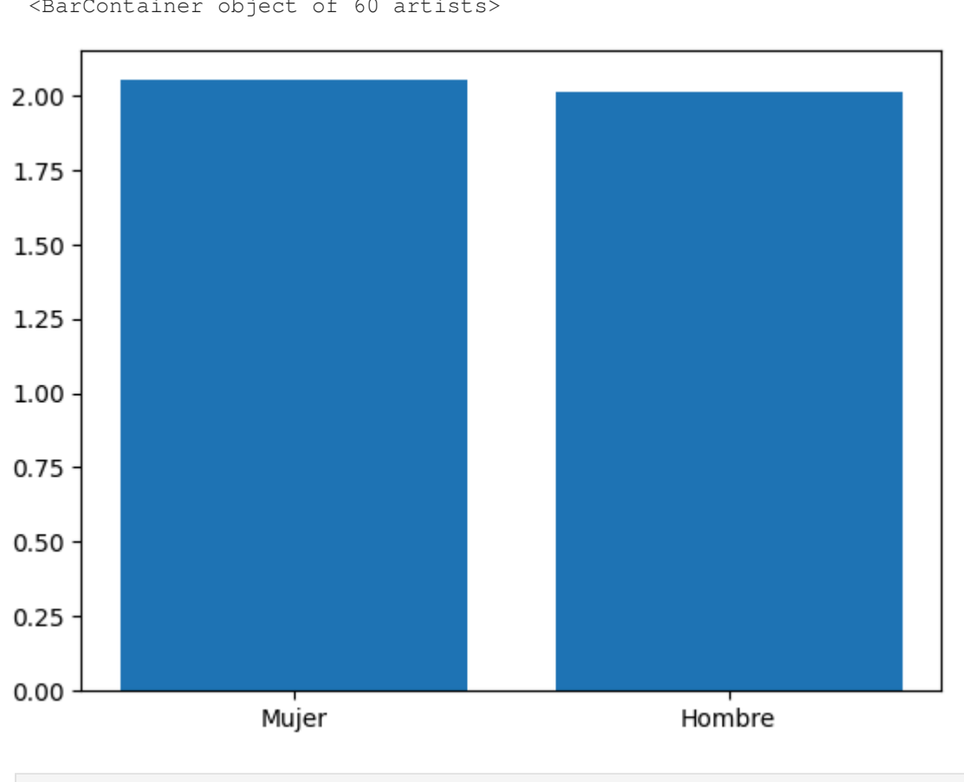
```
count    60.000000
mean      1.748667
std       0.139156
min       1.410000
25%       1.650000
50%       1.730000
75%       1.842500
max       2.050000
Name: Estatura, dtype: float64
```

In [11]:
```python
#3b. Generamos una tabla de frecuencias.
print(df['Sexo'].value_counts())
```

```
Sexo
Mujer     40
Hombre    20
Name: count, dtype: int64
```

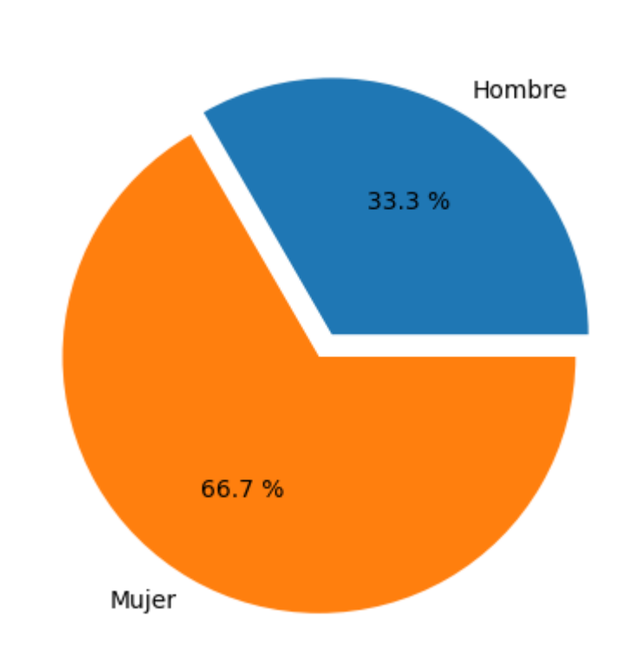
In [12]:
```python
#3c. Generamos un diagrama de barras.
plt.bar(df.Sexo,df.Estatura)
```

Out[12]: `<BarContainer object of 60 artists>`



In [13]:
```python
#3d. Generamos un diagrama de sectores.
sect = df.groupby(['Sexo']).agg(frequency=("Sexo", "count"))
#plt.pie(sect.frequency,labels=sect.index) Esto es el codigo para que esté junto
desfase=(0, 0.1)
plt.pie(sect.frequency,labels=sect.index,autopct="%0.1f %%", explode = desfase)
```

Out[13]:
```
([<matplotlib.patches.Wedge at 0x1be1054b500>,
  <matplotlib.patches.Wedge at 0x1be0e24e2d0>],
 [Text(0.5499999722395388, 0.9526279601903919, 'Hombre'),
  Text(-0.5999998788634385, -1.0392305544795433, 'Mujer')],
 [Text(0.29999998485793017, 0.5196152510129409, '33.3 %'),
  Text(-0.3499999293370058, -0.6062178234464002, '66.7 %')])
```



4. Hallar la estatura media por sexo y generar los histogramas de alturas por sexo para compararlos.

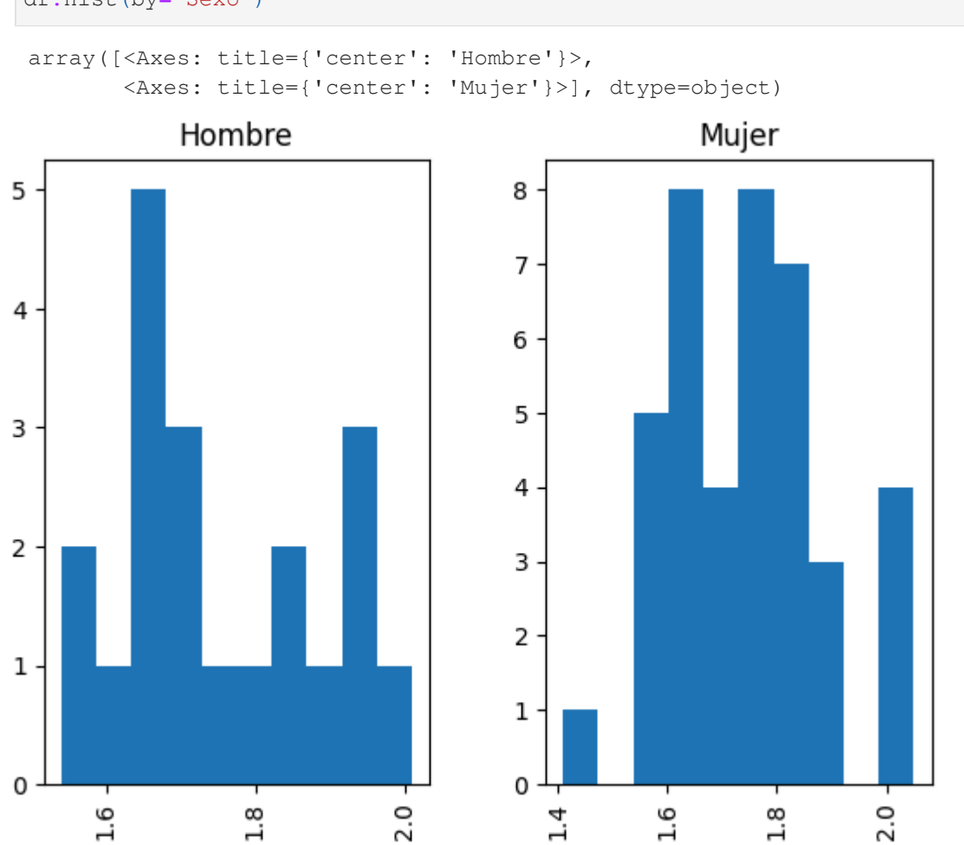
- Previamente, manipula las opciones gráficas de la forma que se desee.

In [14]:
```python
#4a. Halla la estatura media por sexo
media = df.groupby(['Sexo']).mean()
print(media)
```

```
        Estatura
Sexo
Hombre     1.756
Mujer      1.745
```

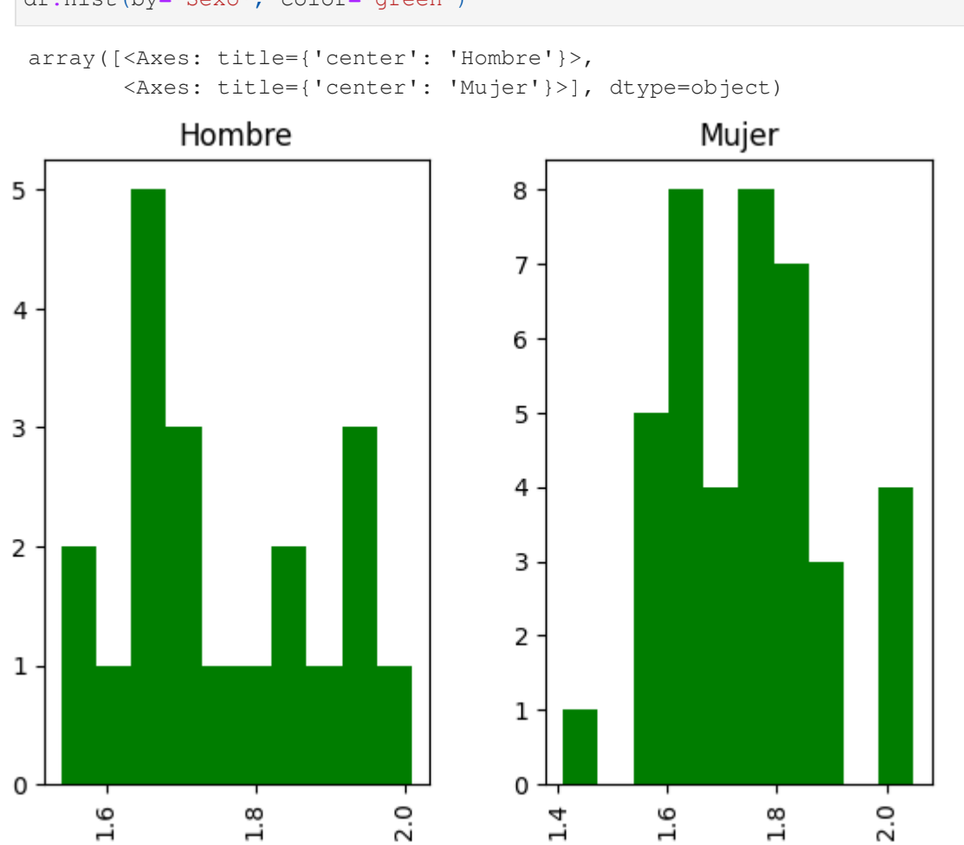
In [15]:
```python
#4b. Genera los histogramas de alturas por sexo y poder compararlos.
df.hist(by='Sexo')
```

Out[15]:
```
array([<Axes: title={'center': 'Hombre'}>,
       <Axes: title={'center': 'Mujer'}>], dtype=object)
```



In [16]:
```python
#4c. Manipula el gráfico anterior en la forma que desees para poder ser presentado.
df.hist(by='Sexo', color='green')
```

Out[16]:
```
array([<Axes: title={'center': 'Hombre'}>,
       <Axes: title={'center': 'Mujer'}>], dtype=object)
```



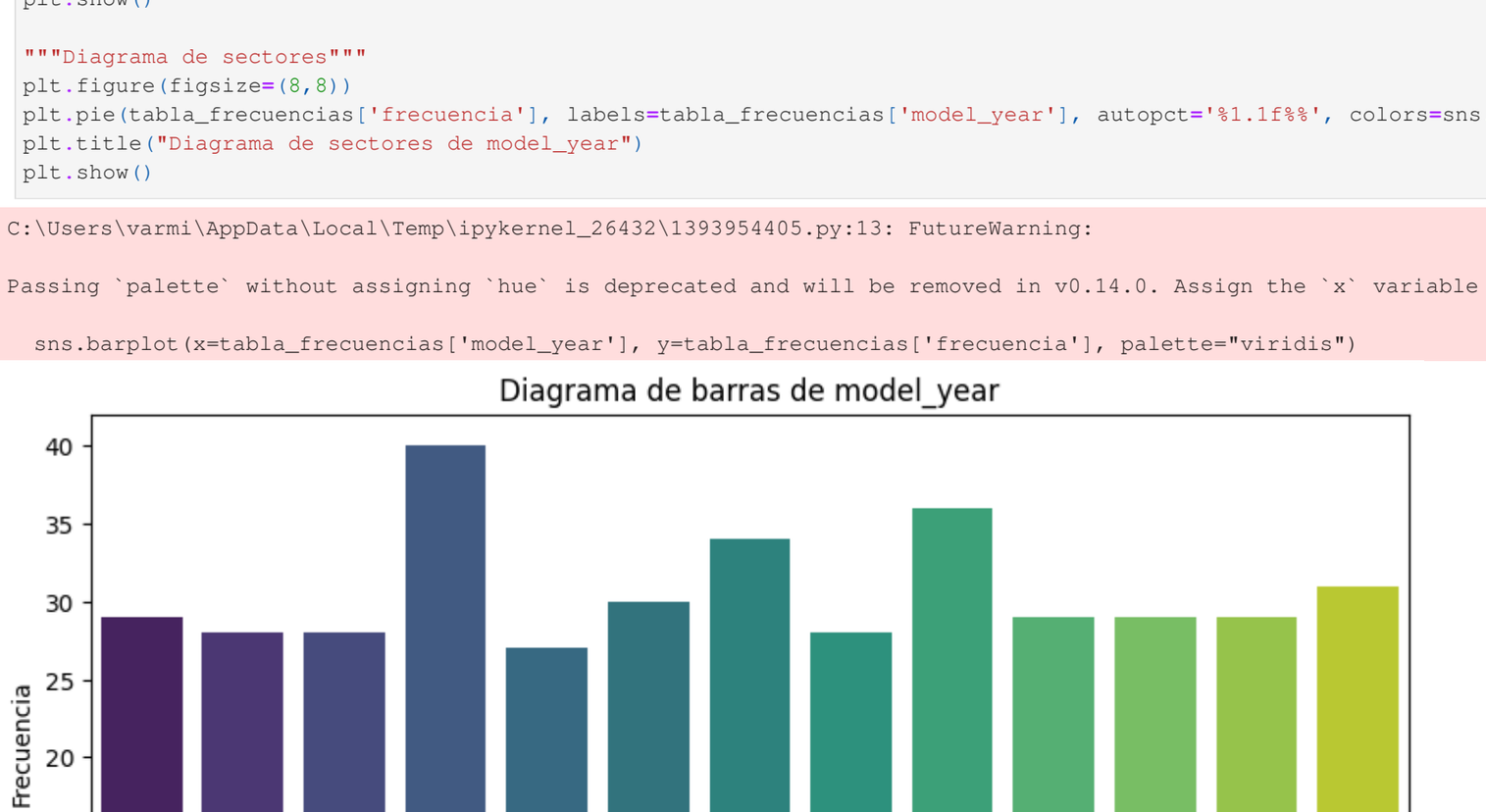
5. El dataset `mpg` contiene información sobre características de un conjunto de vehículos.

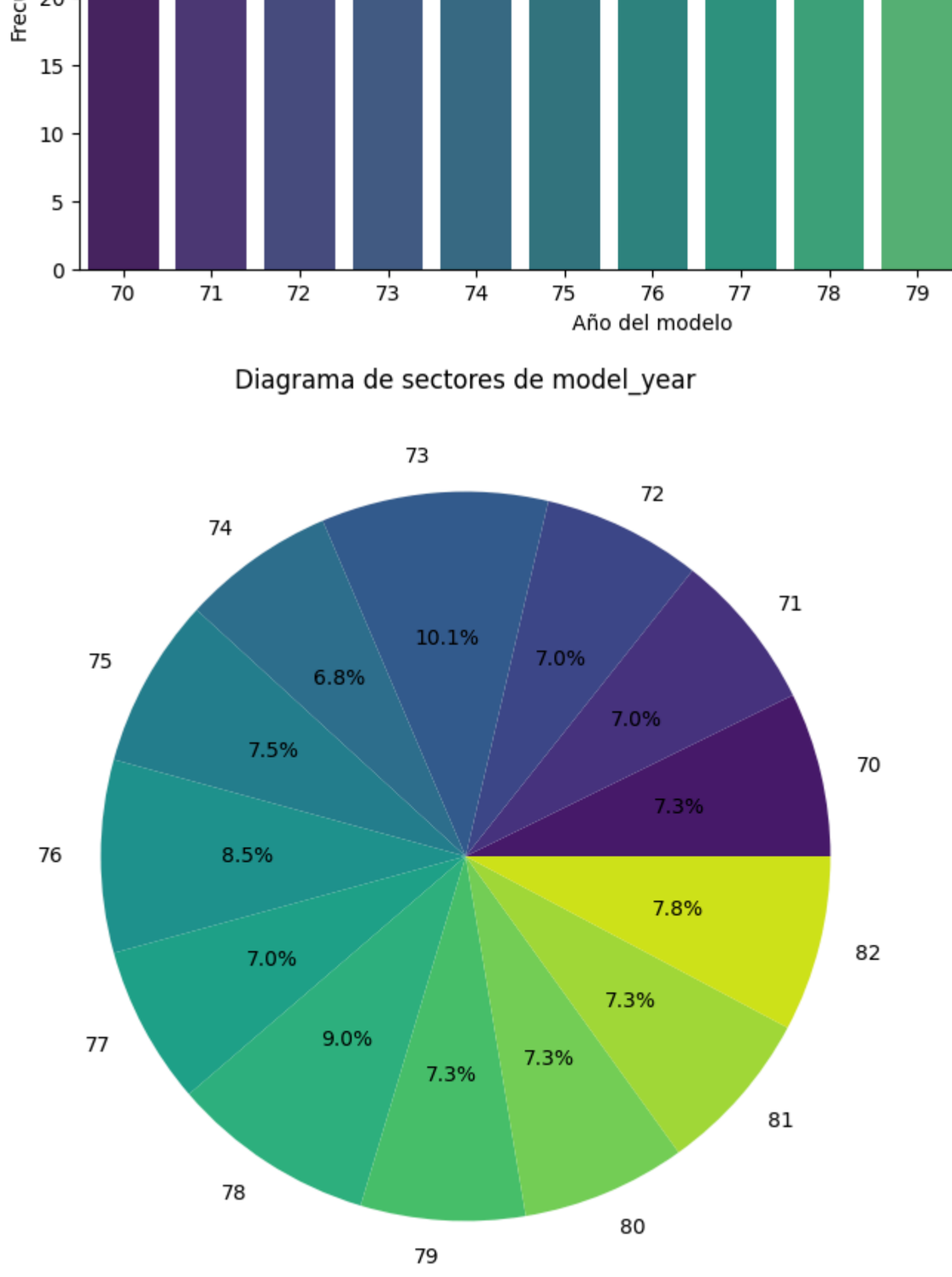
1. Para cargar el dataset, usad la siguiente función `sns.load_dataset('mpg')`
2. Haz una tabla de frecuencias, un diagrama de barras y un diagrama de sectores de la variable `model_year` (año de fabricación).
3. Haz una tabla de frecuencias para la variable `mpg`. ¿Qué porcentaje de coches consumenos menos de 22 mpg (usa intervalos de 22)?
4. Haz el box-and-whisker plot de alguna de las variables cuantitativas y comenta sus características más importantes (concentraciones, asimetrías, atípicos...).
5. Haz un polígono de frecuencias RELATIVAS acumuladas de la variable `horsepower` (potencia - caballos).
6. Utilizando el polígono o diagrama de frecuencias relativas acumuladas, di aproximadamente la proporción de vehículos que tienen una potencia menor o igual a 100 CV.

In [17]:
```python
#5x. Carga el dataset mpg.

mpg = sns.load_dataset('mpg')
```

In [18]:
```python
#5a. Genera una tabla de frecuencias, un diagrama de barras y un diagrama de sectores de la variable
#    'model_year' (año de fabricación)
"""Tabla de frecuencias"""
tabla_frecuencias = pd.DataFrame({
    'model_year': mpg['model_year'].value_counts().index,
    'frecuencia': mpg['model_year'].value_counts().values
}).sort_values(by='model_year')

tabla_frecuencias

"""Diagrama de barras"""
plt.figure(figsize=(10,5))
sns.barplot(x=tabla_frecuencias['model_year'], y=tabla_frecuencias['frecuencia'], palette="viridis")
plt.xlabel("Año del modelo")
plt.ylabel("Frecuencia")
plt.title("Diagrama de barras de model_year")
plt.show()

"""Diagrama de sectores"""
plt.figure(figsize=(8,8))
plt.pie(tabla_frecuencias['frecuencia'], labels=tabla_frecuencias['model_year'], autopct='%1.1f%%', colors=sns.color_palette("viridis", len(tabla_frecuencias)))
plt.title("Diagrama de sectores de model_year")
plt.show()
```

```
C:\Users\varmi\AppData\Local\Temp\ipykernel_26432\1393954405.py:13: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=tabla_frecuencias['model_year'], y=tabla_frecuencias['frecuencia'], palette="viridis")
```

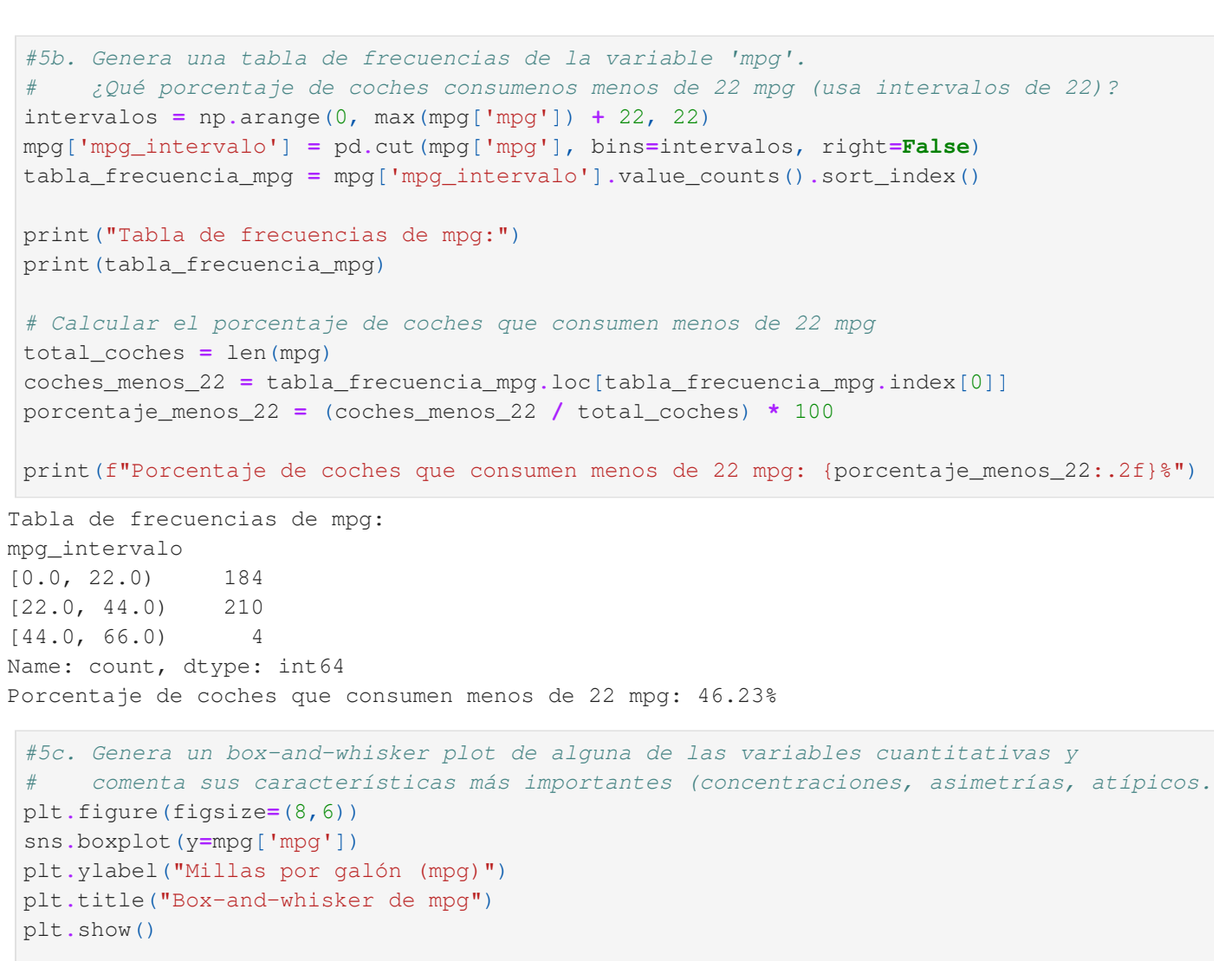


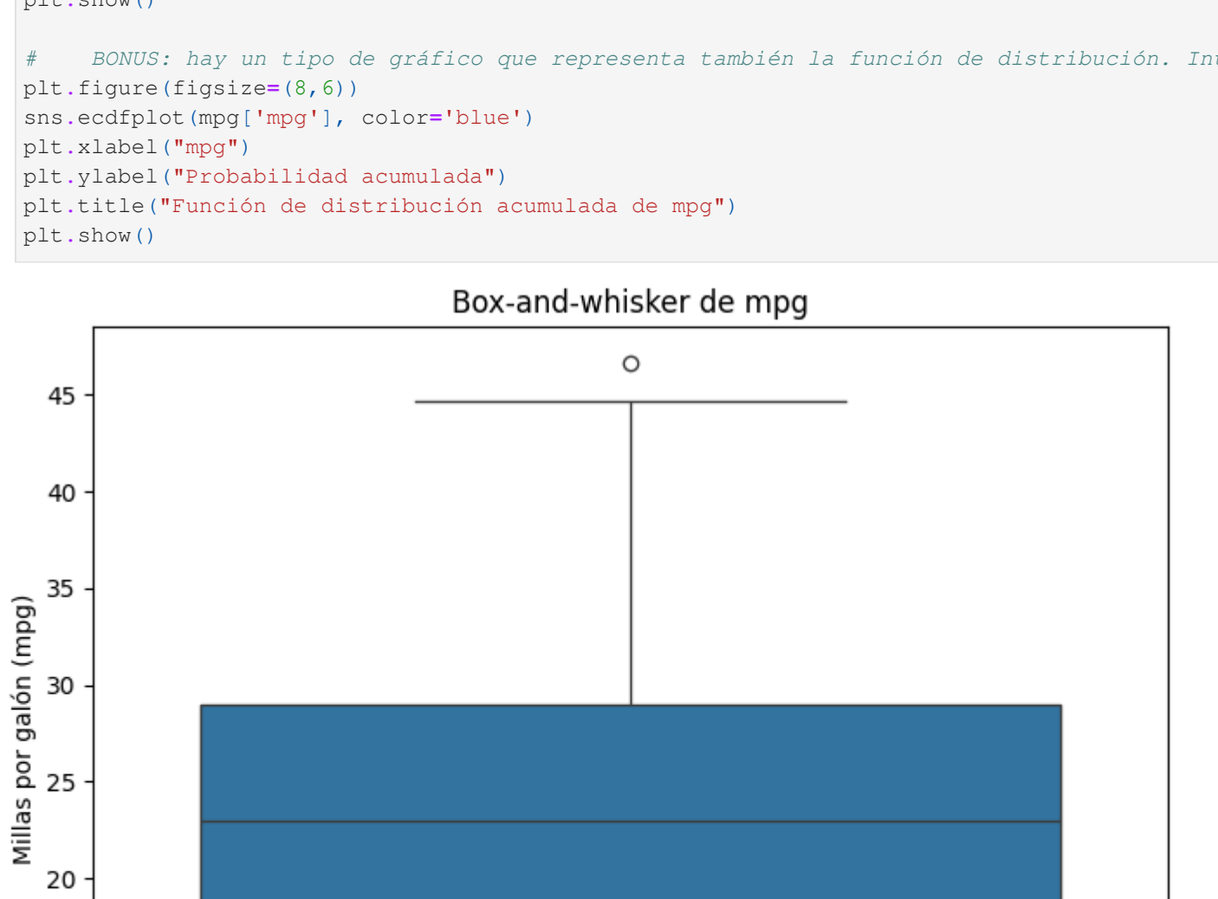


In [19]:
```python
#5b. Genera una tabla de frecuencias de la variable 'mpg'.
#    ¿Qué porcentaje de coches consumenos menos de 22 mpg (usa intervalos de 22)?
intervalos = np.arange(0, max(mpg['mpg']) + 22, 22)
mpg['mpg_intervalo'] = pd.cut(mpg['mpg'], bins=intervalos, right=False)
tabla_frecuencia_mpg = mpg['mpg_intervalo'].value_counts().sort_index()

print("Tabla de frecuencias de mpg:")
print(tabla_frecuencia_mpg)

# Calcular el porcentaje de coches que consumen menos de 22 mpg
total_coches = len(mpg)
coches_menos_22 = tabla_frecuencia_mpg.loc[tabla_frecuencia_mpg.index[0]]
porcentaje_menos_22 = (coches_menos_22 / total_coches) * 100

print(f"Porcentaje de coches que consumen menos de 22 mpg: {porcentaje_menos_22:.2f}%")
```

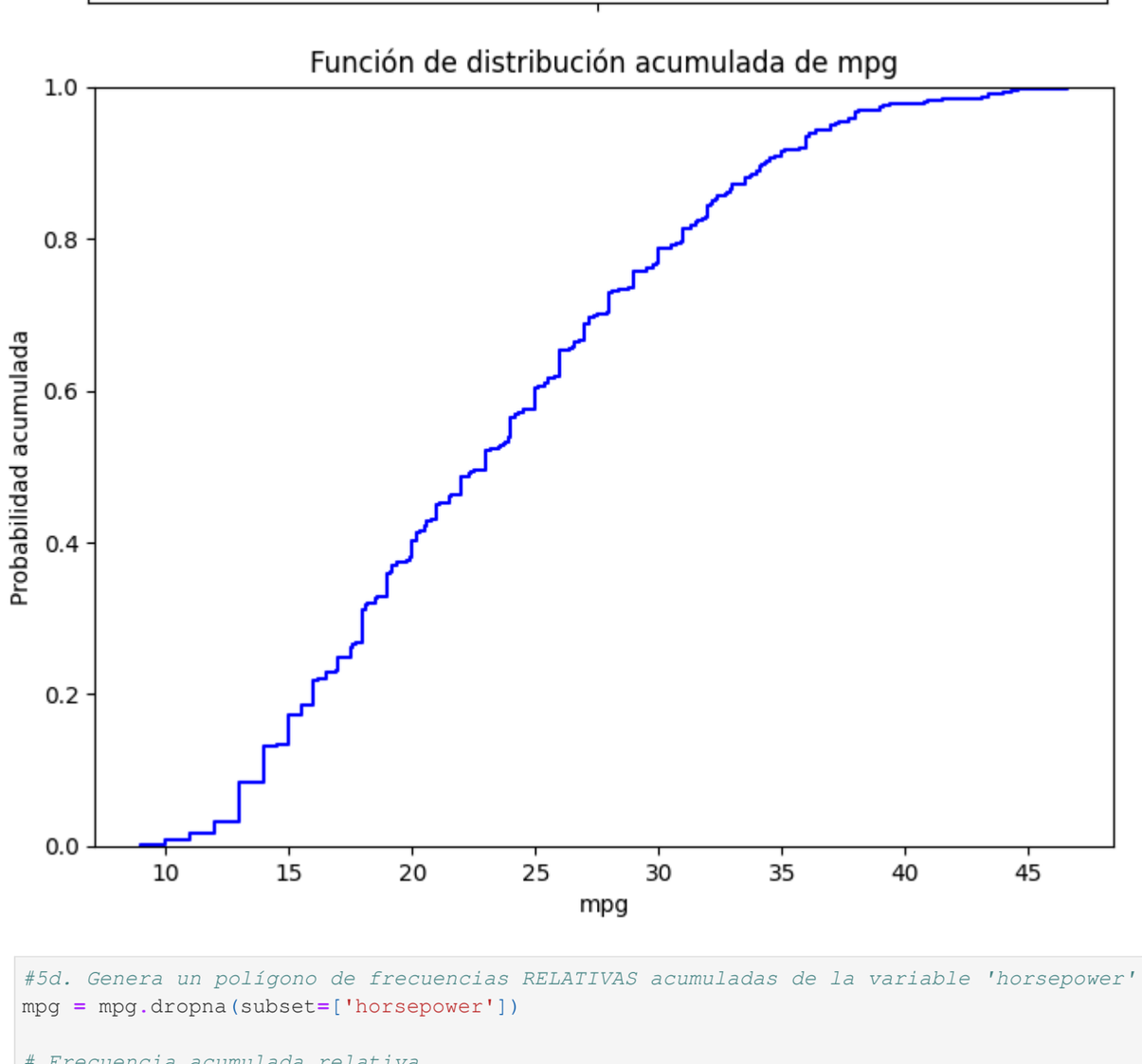
```
Tabla de frecuencias de mpg:
mpg_intervalo
[0.0, 22.0)     184
[22.0, 44.0)    210
[44.0, 66.0)      4
Name: count, dtype: int64
Porcentaje de coches que consumen menos de 22 mpg: 46.23%
```

In [20]:
```python
#5c. Genera un box-and-whisker plot de alguna de las variables cuantitativas y
#    comenta sus características más importantes (concentraciones, asimetrías, atípicos...)
plt.figure(figsize=(8,6))
sns.boxplot(y=mpg['mpg'])
plt.ylabel("Millas por galón (mpg)")
plt.title("Box-and-whisker de mpg")
plt.show()

#    BONUS: hay un tipo de gráfico que representa también la función de distribución. Intenta encontrarlo.
plt.figure(figsize=(8,6))
sns.ecdfplot(mpg['mpg'], color='blue')
plt.xlabel("mpg")
plt.ylabel("Probabilidad acumulada")
plt.title("Función de distribución acumulada de mpg")
plt.show()
```





In [21]:
```python
#5d. Genera un polígono de frecuencias RELATIVAS acumuladas de la variable 'horsepower' (potencia - caballos).
mpg = mpg.dropna(subset=['horsepower'])

# Frecuencia acumulada relativa
tabla_frecuencia_hp = mpg['horsepower'].value_counts().sort_index().cumsum()
tabla_frecuencia_hp_relativa = tabla_frecuencia_hp / tabla_frecuencia_hp.max()

# Graficar las frecuencias relativas acumuladas
plt.figure(figsize=(8,6))
plt.plot(tabla_frecuencia_hp_relativa.index, tabla_frecuencia_hp_relativa.values, marker='o', linestyle='-', color='pink')
plt.xlabel("Caballos de fuerza (horsepower)")
plt.ylabel("Frecuencia relativa acumulada")
plt.title("Polígono de frecuencias relativas acumuladas de horsepower")
plt.grid()
plt.show()
```



In [22]:
```python
#5e. Utilizando la figura previa, di aproximadamente la proporción de vehículos que tienen una
#    potencia menor o igual a 100 CV.
# Obtenemos la frecuencia relativa acumulada para los coches con 100 cv
frecuencia_relativa_100cv = tabla_frecuencia_hp_relativa.loc[100]

# Calcular el número de vehículos con 100 cv o menos
numero_vehiculos_100cv_o_menos = frecuencia_relativa_100cv * len(mpg)
print(f"Número de vehículos con 100 cv o menos: {numero_vehiculos_100cv_o_menos:.0f}")
```

```
Número de vehículos con 100 cv o menos: 242
```

Realizar un informe con los análisis realizados de cada uno de los apartados anteriores y exportarlo en un fichero .PDF

El informe se realizará en un grupos de dos personas analizando los resultados obtenidos en cada uno de los apartados anteriores. Todas las gráficas deberán estar bien maquetadas: título, titulo en los ejes, ejes con un intervalo lógico y leyenda (si es el caso). El código empleado también se deberá incluir.

En el Moodle se encuentra un trabajo tipo para que tengáis de referencia. El archivo incluirá los nombres de todos autores en el siguiente formato:

Nombre del archivo: Practica1_ApellidosNombre1_ApellidosNombre2.pdf (extensión obligatoria en .pdf)

Se avisará en Moodle la fecha límite de entrega de este informe.

In [ ]:

In [ ]:

In [ ]:

In [ ]: