In [11]:
```
# Ficheros de configuración .ini
# [BBDD]
# Server = nombre-de-mi-servidor
# Database = nombre-de-mi-base-de-datos
# UserId = nombre-de-mi-usuario

import configparser

config = configparser.ConfigParser()

config['BBDD'] = {'Server': 'localhost',

                  'Database': 'test.db',

                  'UserId': 'admin'}

with open('test.ini', 'w') as configfile:
        config.write(configfile)
```

In [12]:
```
config = configparser.ConfigParser()
config.read('test.ini')
usuari = config['BBDD']['UserId']

print(usuari)
```

```
admin
```

In [14]:
```
from datetime import date

def mylog (message, level ="DEBUG") :
    f = open ('./logs/myapp.log', 'a', encoding='UTF-8',)
    f.write( f" {date.today()} {level} {message}\n")
    f.close()

mylog("Inicio el programa")
mylog("Error en floats y tal", "ERROR")
mylog("Adios", "INFO")

# Creamos una función log a mano
# Ejercicio 2. Modifica para que además de la fecha grave la hora.
```

In [3]:
```
import logging

logging.basicConfig(filename='example4.log', level=logging.DEBUG,
                    format='%(levelname)s%(asctime)s %(message)s')
logging.debug('This message should go to the log file')
logging.info('So should this')
logging.warning('And this, too')
logging.error('And non-ASCII stuff, too, like Øresund and Malmö')
```

In [4]:
```
from datetime import datetime


def mylog (message, level ="DEBUG") :
    f = open ('./logs/myapp.log',  'a', encoding='UTF-8',)
    f.write( f" {datetime.today()} {level} {message}\n")
    f.close()

mylog("Inicio el programa")
mylog("Error en floats y tal", "ERROR")
mylog("Adios", "INFO")
```

In [17]:
```
import pickle

#Podemos guardar lo que queramos, listas, diccionarios,tuplas..

lista = [1,2,3,4,5]

# Escritura en modo binario, vacía el fichero si existe

fichero = open('./dat/lista.pckl','wb')

#Escribe la colección en el fichero

pickle.dump(lista, fichero)



fichero.close()
```

In [18]:
```
# Lectura en modo binario
fichero = open('./dat/lista.pckl','rb')

# Cargamos los datos del fichero
lista_fichero = pickle.load(fichero)

print(lista_fichero)

fichero.close()
```

```
[1, 2, 3, 4, 5]
```

In [26]:
```
# frutas = {'Platano': 1.35, 'Manzana' : 0.8, 'Pera' : 0.85, 'Naranja' : 0.70 }

def pide_opcion () :
    print ("1.Listar, 2.Crear, 3.Eliminar, 0.Salir")
    while (opcion := input("opcion: ")) not in ('1230'):
        print ("opcion inválida")

    return opcion

def pide_fruta () :
    return input("fruta").capitalize()

def pide_precio () :
    try :
        precio = input("precio").replace(",", ".")
        precio = float(precio)
    except Exception as e:
        precio = 0.0
        print ("precio invalido")

    return precio

def cargar_fichero(frutas):
    try:
        fichero = open('./dat/frutas.pckl','rb')
        frutas = pickle.load(fichero)
        fichero.close()
    except Exception as e:
        guardar_fichero(frutas)
    return(frutas)

def guardar_fichero(frutas):
    try:
        fichero = open('./dat/frutas.pckl','wb')
        pickle.dump(frutas,fichero)
        fichero.close()
    except Exception as e:
        print("Error,e)")



#--------------------------- Inicio del programa

frutas = dict()

frutas = cargar_fichero(frutas)

opcion = pide_opcion ()

while opcion != '0' :
    if opcion == '1' :
        print (frutas)
    elif opcion == '2' :
        fruta = pide_fruta ()
        precio = pide_precio()
        frutas [fruta] = precio
    elif opcion == '3' :
        fruta = pide_fruta ()
```

```
        if fruta in frutas :
            frutas.pop(fruta)
    opcion = pide_opcion ()


guardar_fichero (frutas)
```

```
1.Listar, 2.Crear, 3.Eliminar, 0.Salir
opcion: 1
{'Pera': 23.0, 'Kiwi': 0.99}
1.Listar, 2.Crear, 3.Eliminar, 0.Salir
opcion: 0
```

In [23]:
```
fichero = open('./dat/frutas.pckl','wb')
pickle.dump(dict(),fichero)
fichero.close()
```

In [42]:
```
import pandas as pd
df = pd.read_csv('./dat/colesterol.csv', sep=";", decimal=",")
df.info()
# print(df)
df
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14 entries, 0 to 13
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   nombre      14 non-null     object 
 1   edad        14 non-null     int64  
 2   sexo        14 non-null     object 
 3   peso        13 non-null     float64
 4   altura      14 non-null     float64
 5   colesterol  13 non-null     float64
dtypes: float64(3), int64(1), object(2)
memory usage: 800.0+ bytes
```

Out[42]:

	nombre	edad	sexo	peso	altura	colesterol
0	José Luis Martínez Izquierdo	18	H	85.0	1.79	182.0
1	Rosa Díaz Díaz	32	M	65.0	1.73	232.0
2	Javier García Sánchez	24	H	NaN	1.81	191.0
3	Carmen López Pinzón	35	M	65.0	1.70	200.0
4	Marisa López Collado	46	M	51.0	1.58	148.0
5	Antonio Ruiz Cruz	68	H	66.0	1.74	249.0
6	Antonio Fernández Ocaña	51	H	62.0	1.72	276.0
7	Pilar Martín González	22	M	60.0	1.66	NaN
8	Pedro Gálvez Tenorio	35	H	90.0	1.94	241.0
9	Santiago Reillo Manzano	46	H	75.0	1.85	280.0
10	Macarena Álvarez Luna	53	M	55.0	1.62	262.0
11	José María de la Guía Sanz	58	H	78.0	1.87	198.0
12	Miguel Angel Cuadrado Gutiérrez	27	H	109.0	1.98	210.0
13	Carolina Rubio Moreno	20	M	61.0	1.77	194.0

In [43]:
```
# Ejercicio P1 : Carga el archivo colesterol.csv en pandas, estudia los datos.
# 1. Elimina las filas con datos nulos
# 2. Ordena el resto por tasa de colesterol y muéstralo
# 3. Genera un gráfico de líneas que diferencie hombres y mujeres y muestre la rela
# entre peso y tasa de colesterol.

df = df.dropna()

df
```

Out[43]:

	nombre	edad	sexo	peso	altura	colesterol
0	José Luis Martínez Izquierdo	18	H	85.0	1.79	182.0
1	Rosa Díaz Díaz	32	M	65.0	1.73	232.0
3	Carmen López Pinzón	35	M	65.0	1.70	200.0
4	Marisa López Collado	46	M	51.0	1.58	148.0
5	Antonio Ruiz Cruz	68	H	66.0	1.74	249.0
6	Antonio Fernández Ocaña	51	H	62.0	1.72	276.0
8	Pedro Gálvez Tenorio	35	H	90.0	1.94	241.0
9	Santiago Reillo Manzano	46	H	75.0	1.85	280.0
10	Macarena Álvarez Luna	53	M	55.0	1.62	262.0
11	José María de la Guía Sanz	58	H	78.0	1.87	198.0
12	Miguel Angel Cuadrado Gutiérrez	27	H	109.0	1.98	210.0
13	Carolina Rubio Moreno	20	M	61.0	1.77	194.0

In [44]:
```
by_colesterol = df.sort_values('colesterol')
by_colesterol
```

Out[44]:

	nombre	edad	sexo	peso	altura	colesterol
4	Marisa López Collado	46	M	51.0	1.58	148.0
0	José Luis Martínez Izquierdo	18	H	85.0	1.79	182.0
13	Carolina Rubio Moreno	20	M	61.0	1.77	194.0
11	José María de la Guía Sanz	58	H	78.0	1.87	198.0
3	Carmen López Pinzón	35	M	65.0	1.70	200.0
12	Miguel Angel Cuadrado Gutiérrez	27	H	109.0	1.98	210.0
1	Rosa Díaz Díaz	32	M	65.0	1.73	232.0
8	Pedro Gálvez Tenorio	35	H	90.0	1.94	241.0
5	Antonio Ruiz Cruz	68	H	66.0	1.74	249.0
10	Macarena Álvarez Luna	53	M	55.0	1.62	262.0
6	Antonio Fernández Ocaña	51	H	62.0	1.72	276.0
9	Santiago Reillo Manzano	46	H	75.0	1.85	280.0

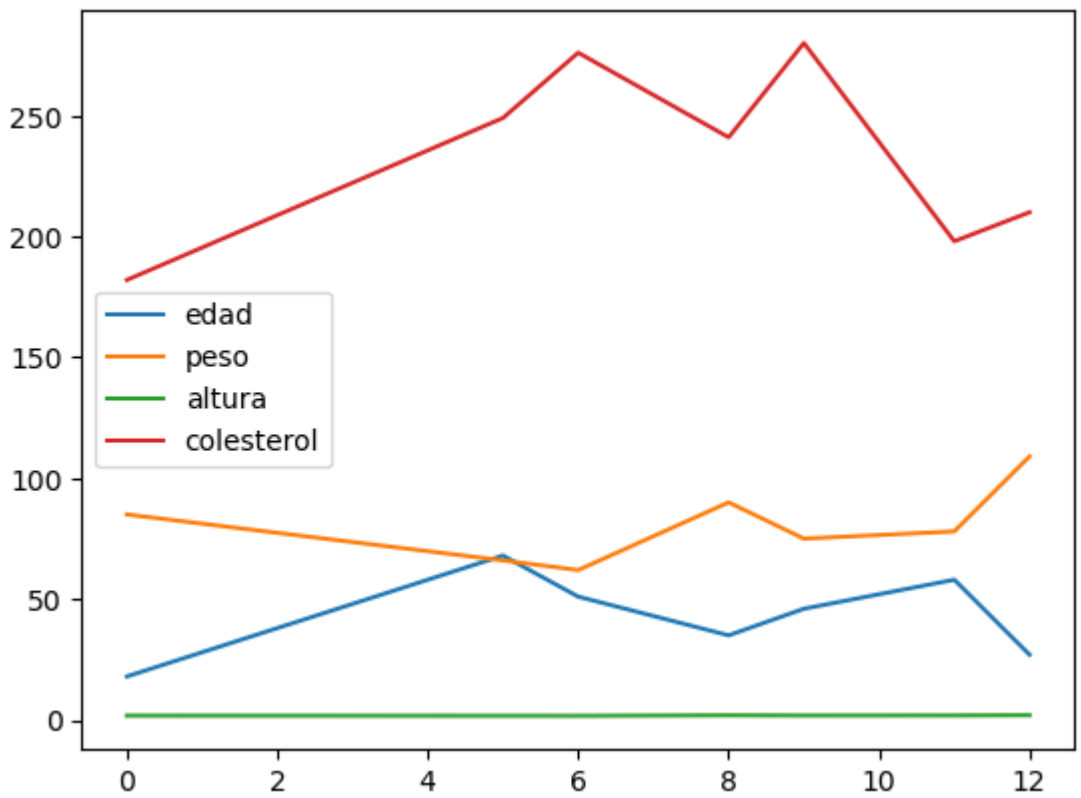
In [46]:
```
dfh = df[df['sexo'] == 'H']

dfh


# import matplotlib.pyplot as plt
# fig, df = plt.subplots(sexo)
# sexo.plot(['H'], ['M'])
# plt.show()
```

Out[46]:

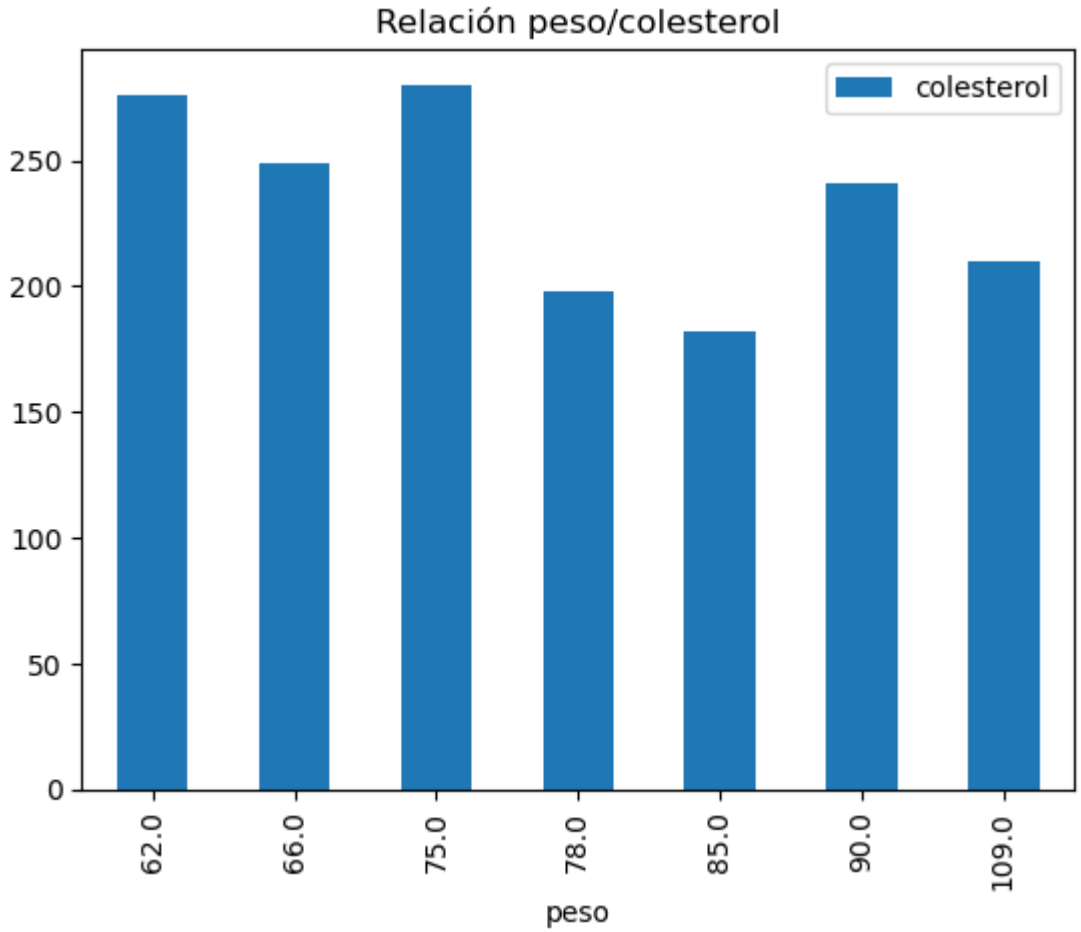
	nombre	edad	sexo	peso	altura	colesterol
0	José Luis Martínez Izquierdo	18	H	85.0	1.79	182.0
5	Antonio Ruiz Cruz	68	H	66.0	1.74	249.0
6	Antonio Fernández Ocaña	51	H	62.0	1.72	276.0
8	Pedro Gálvez Tenorio	35	H	90.0	1.94	241.0
9	Santiago Reillo Manzano	46	H	75.0	1.85	280.0
11	José María de la Guía Sanz	58	H	78.0	1.87	198.0
12	Miguel Angel Cuadrado Gutiérrez	27	H	109.0	1.98	210.0

In [51]:
```
dfh.plot()
```

Out[51]:
```
<Axes: >
```



In [57]:
```
import matplotlib.pyplot as plot

dfh =dfh.sort_values(by=['peso'])
dfh.plot.bar(x='peso', y='colesterol', title="Relación peso/colesterol"),
plot.show()
```

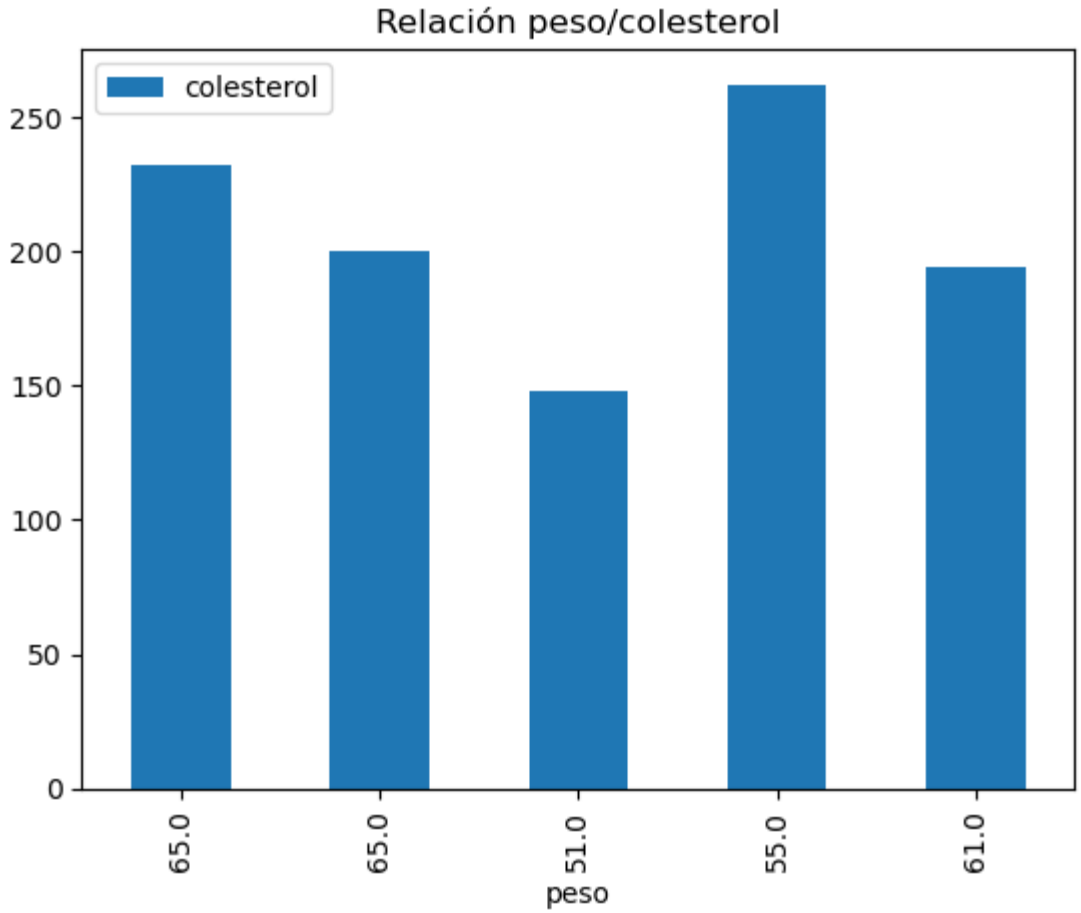


In [54]:
```
dfm = df[df['sexo'] == 'M']
dfm
```

Out[54]:

	nombre	edad	sexo	peso	altura	colesterol
1	Rosa Díaz Díaz	32	M	65.0	1.73	232.0
3	Carmen López Pinzón	35	M	65.0	1.70	200.0
4	Marisa López Collado	46	M	51.0	1.58	148.0
10	Macarena Álvarez Luna	53	M	55.0	1.62	262.0
13	Carolina Rubio Moreno	20	M	61.0	1.77	194.0

In [55]:
```
dfm.plot.bar(x='peso', y='colesterol', title="Relación peso/colesterol"),
plot.show()
```



In [72]:
```
# Genera un fichero csv (que se llame colesterol_anonimo.csv) que no contenga ni el
# nombre ni el sexo de la persona. (ho fem abans creant un df amb menys columnes)
# - Usa el punto y coma como separador
# - Y la coma como punto decimal.

dfh.to_csv('./dat/colesterol_anonimo.csv', index=False, sep=';',decimal=",")
```

In [75]:
```
dfh.to_xml('./dat/colesterol_anonimo.xml')
dfh.to_html('./dat/colesterol_anonimo.html')
dfh.to_excel('./dat/colesterol_anonimo.xlsx')

js = dfh.to_json(orient = 'columns')
print(js)
```

```
{"nombre":{"6":"Antonio Fern\u00e1ndez Oca\u00f1a","5":"Antonio Ruiz Cruz","9":"San
tiago Reillo Manzano","11":"Jos\u00e9 Mar\u00eda de la Gu\u00eda Sanz","0":"Jos\u00
e9 Luis Mart\u00ednez Izquierdo","8":"Pedro G\u00e1lvez Tenorio","12":"Miguel Angel
Cuadrado Guti\u00e9rrez"},"edad":{"6":51,"5":68,"9":46,"11":58,"0":18,"8":35,"12":2
7},"sexo":{"6":"H","5":"H","9":"H","11":"H","0":"H","8":"H","12":"H"},"peso":{"6":6
2.0,"5":66.0,"9":75.0,"11":78.0,"0":85.0,"8":90.0,"12":109.0},"altura":{"6":1.72,"
5":1.74,"9":1.85,"11":1.87,"0":1.79,"8":1.94,"12":1.98},"colesterol":{"6":276.0,"
5":249.0,"9":280.0,"11":198.0,"0":182.0,"8":241.0,"12":210.0}}
```

In [ ]: