Objetivo

Diseñar programas y funciones en Python que contengan datos estructurados diccionarios (dict) y DataFrames de Pandas.

Diccionarios

1. **Frecuencia de pares**. Diseña una función `dpar(M)`, en que, dada una lista `M` de números enteros positivos, devuelva un diccionario donde se muestre la frecuencia de aparición de los números pares de `M`. Ejemplo en el *doctring* siguiente:

```
def dpar(M):
    '''
    >>> M = [12, 19, 19, 18, 18, 16, 18, 13, 19, 18, 12, 18, 11, 20, 14,\
             14, 20, 20, 20, 16, 13, 15, 19, 14, 12]
    >>> b = dpar(M)
    >>> b == {12: 3, 18: 5, 16: 2, 20: 4, 14: 3}
    True
    '''
```

2. **Temperaturas ciudades**. Se tienen las temperaturas de ciudades durante los primeros 4 meses del año en una lista de listas, donde cada lista representa la información de una ciudad (nombre y luego las temperaturas). Diseña una función `TempMaxMin(lst)` en que, dada una lista de listas como que la que se presenta, devuelva un diccionario con los **nombres de las ciudades** como **claves** y como **valores** una lista con **las temperaturas máxima y mínima**. Ejemplo de lista anidada: lst_ciudad.

```
lst_ciudad = [['Londres', 3.4, 6.3, 10.5, 6.8], ['Oslo', -3.8, -5.0, 5.1, 4.2],
       ['Berlin', 7.5, 4.1, 12.3, 13.0], ['Málaga', 14.7, 12.3, 19.5, 18.4]]
def TempMaxMin(lst):
    """
    >>> lst_ciudad = [['Londres', 3.4, 6.3, 10.5, 6.8], \
        ['Oslo', -3.8, -5.0, 5.1, 4.2], ['Berlin', 7.5, 4.1, 12.3, 13.0], \
        ['Málaga', 14.7, 12.3, 19.5, 18.4]]
    >>> M = TempMaxMin(lst_ciudad)
    >>> M == {'Berlin': [13.0, 4.1], 'Londres': [10.5, 3.4], \
              'Málaga': [19.5, 12.3], 'Oslo': [5.1, -5.0]}
    True
    """
```

3. **Temperaturas nevada**. Se dispone de dos diccionarios. En uno de ellos (ej. `dPersC`) se guardan, como claves, nombres de personas y como valores los nombres de las ciudades donde residen; y en el otro (ej. `dCiudT`) se guardan, como claves, nombres de ciudades y como valores las respectivas temperaturas de esas ciudades (en ºC), medidas el día de la primera nevada del año.

Diseña una función `PersMayTemp(dPersC, dCiudT)` en que, dados dos diccionarios como los descritos, devuelva una lista con los nombres de las personas que residen en ciudades en las cuales la temperatura estuvo por debajo de 0ºC el día de esa primera nevada de este año. La lista resultante debe estar ordenada alfabéticamente. En caso de no haber ciudades con temperaturas bajo cero, se devolverá la lista vacía.

Nota: se considera que toda ciudad del diccionario `dPersC` está en `dCiudT`. Ejemplo:

```
def PersMayTemp(dPersC, dCiudT):
    """
    >>> dCi = {'Manchester': 1.1, 'Madrid': -8.9, 'Gava': 4, \
               'Pobla de Segur': -5.6, 'Lleida': -3.2, 'Elche': 2.1, \
               'Burgos': -6.0, 'Sant Boi': 4.5}
    >>> dPe = {'Pepe': 'Manchester', 'Lionel': 'Gava', 'Mike': 'Sant Boi', \
               'Puyol': 'Pobla de Segur', 'Jaime': 'Elche', 'Sergi': 'Lleida',\
                'Ernesto': 'Madrid', 'Carlos': 'Burgos'}
    >>> PersMayTemp(dPe, dCi)
    ['Carlos', 'Ernesto', 'Puyol', 'Sergi']
    """
```

4. **Hipertensión**. Se dispone de un diccionario de personas con su presión arterial. En cada elemento del diccionario la clave es el nombre de la persona y el valor es una lista con la edad y las presiones sistólica (alta) y diastólica (baja). Si consideramos que una persona sufre de hipertensión si la presión sistólica es mayor o igual a 140 mmHg o la diastólica es mayor o igual a 90 mmHg, diseña una función `lst_hiper(dic, edad)` en que, dado un diccionario `dic` como el descrito y una `edad`, devuelva la lista de los nombres de las personas menores de esa edad que sufren hipertensión arterial.

Se valorará devolver la lista de nombres ordenada alfabéticamente.

```
def lst_hiper(dicc, edad):
    """
    >>> dpers = {'Maria': [40, 135, 90],'Nuria': [63, 141, 92], \
                 'Jose': [47, 110, 59], 'Luis': [49, 146, 94], \
                 'Oriol': [52, 130, 89], 'Carlos': [65, 125, 89], \
                 'Pepe': [70, 130, 92] }
    >>> lst_hiper(dpers, 45)
    ['Maria']
    >>> lst_hiper(dpers, 70)
    ['Luis', 'Maria', 'Nuria']
    """
```

5. **Nivel de potasio en sangre**. Se tiene un diccionario con los valores de concentración de potasio en sangre ([K+]) de un grupo pacientes medidos antes de entrar a terapia de hemodiálisis. La clave es el nombre del paciente y el valor es la [K+] (en mmol/L). Además, se tiene una lista del tipo [valor1, valor2, valor3, valor4] con los distintos valores que clasifican la condición clínica en que están los pacientes, de acuerdo a las concentraciones de potasio en sangre ([K+]). Estos valores dependen del tipo de población (niños, adultos, etc.).

La condición clínica del paciente sigue el siguiente criterio de clasificación:
- Si el valor de [K+] es menor que valor1 indica 'hipokalemia crítica',
- si es mayor o igual al valor1 y menor que el valor2 sería 'hipokalemia leve',
- si está entre valor2 y valor3 (ambos inclusive) indica 'normal',
- si es mayor que valor3 y menor o igual que valor4 es 'hiperkalemia moderada'
- y valores mayores que valor4 sería 'hiperkalemia severa'.

Ejemplo de una lista con elementos [valor1, valor2, valor3, valor4] para un adulto de mediana edad: [2.0, 3.5, 5.2, 7.0]

Diseña una función `nivelKsang(dK, lst)` en que, dado un diccionario `dK` que tiene como clave el nombre de un paciente y como valor su nivel de [K+] en sangre, y una lista `lst` con 4 valores de clasificación, como los descritos, devuelva un diccionario que tenga como claves los nombres de los pacientes y como valores su clasificación, de acuerdo a los niveles de [K+] en sangre.

```
def nivelKsang(dK, lst):
    """
    Parameters
    ----------
    dK: tipo diccionario
        Clave: nombre de persona; valor: concentración de potasio en sangre
        ([K+]) en unidades de mmol/L
    lst: tipo lista
        Contiene 4 elementos con los valores de [K+] que se usan de límite en
        la clasificación de los niveles de potasio en sangre

    Returns
    -------
    un diccionario donde las claves son nombres de pacientes (strings)
    y los valores son la clasificación de acuerdo a la [K+] que tengan

    Ejemplo:
    >>> dK1 = {'Luis': 2.2, 'Carlos': 7.0, 'Laia': 4.0, 'Mikel': 5.5, \
               'Jordi': 5.2, 'Anna': 3.6, 'Joe': 7.2}
    >>> ls1 = [2.0, 3.5, 5.2, 7.0]
    >>> dic = nivelKsang(dK1, ls1)
    >>> dic == {'Luis': 'hipokalemia leve', \
                'Carlos': 'hiperkalemia moderada', 'Laia': 'normal', \
                'Mikel': 'hiperkalemia moderada', 'Jordi': 'normal', \
                'Anna': 'normal', 'Joe': 'hiperkalemia severa'}
    True
    """
```

DataFrames

6. **Temperatura ciudades en DataFrame**. A partir de `lst_ciudad` de las temperaturas de las ciudades de los primeros 4 meses del año del ejercicio 2:

a) Diseña un código para crear un objeto DataFrame que contenga en sus columnas: 'Ciudad', 'Enero', 'Febrero', 'Marzo', 'Abril' y los datos sean los valores de las listas de ciudades de `lst_ciudad`. Agregar como nombre del DataFrame: 'Temperatura ciudades'. El DataFrame será como:

	Ciudad	Enero	Febrero	Marzo	Abril
0	Londres	3.4	6.3	10.5	6.8
1	Oslo	-3.8	-5.0	5.1	4.2
2	Berlin	7.5	4.1	12.3	13.0
3	Málaga	14.7	12.3	19.5	18.4

b) Escribe un código para agregar la temperatura mínima, máxima, media y desviación estándar de los 4 primeros meses del año al DataFrame anterior. Resultado de la forma:

	Ciudad	Enero	Febrero	Marzo	Abril	Min	Max	Media	StdDev
0	Londres	3.4	6.3	10.5	6.8	3.4	10.5	6.750	2.914904
1	Oslo	-3.8	-5.0	5.1	4.2	-5.0	5.1	0.125	5.260783
2	Berlin	7.5	4.1	12.3	13.0	4.1	13.0	9.225	4.201091
3	Málaga	14.7	12.3	19.5	18.4	12.3	19.5	16.225	3.326034

7. **Base de datos cardiaca**. La base de datos de enfermedades cardiacas UCI heart.csv, que se encuentra disponible en el aula y en el enlace https://www.kaggle.com/ronitf/heart-disease-uci , contiene 14 atributos (columnas):

1. **age**: edad
2. **sex**: sexo (1: hombre, 0: mujer)
3. **cp**: tipo de dolor en el pecho (4 valores)

4. **trestbps**: presión arterial sistólica en reposo
5. **chol**: colesterol sérico en mg/dl
6. **fbs**: azúcar en sangre (en ayunas) > 120 mg/dl
7. **restecg**: Resultados electrocardiográficos en reposo (valores 0,1,2)
8. **thalach**: frecuencia cardíaca máxima alcanzada
9. **exang**: angina inducida por ejercicio (1: sí, 0: no)
10. **oldpeak**: depresión del segmento ST inducida por el ejercicio relativo al descanso
11. **slope**: pendiente del segmento ST de ejercicio máximo
12. **ca**: número de vasos principales (0-3) coloreados por flourosopía
13. **thal**: 3 = normal; 6 = defecto fijo; 7 = defecto reversible

Y el atributo 14, **target** se refiere a la presencia (1) o no (0) de enfermedad cardiaca.

En este ejercicio se pide que se lea o cargue la base de datos desde heart.csv a un objeto DataFrame (por ejemplo, dfCardio) y se presenten los siguientes resultados:

a) Mostrar las primeras 10 instancias (filas) del DataFrame.
b) Calcular el número (conteo) de hombres y mujeres.
c) Calcular el número (conteo) de casos de angina de pecho inducida (atributo: **exang)**
d) Hallar el DataFrame con la estadística descriptiva de la frecuencia cardiaca (**thalach**)
e) Hallar un DataFrame que incluya la estadística descriptiva de la presión arterial sistólica en reposo (**trestbps**) y el colesterol (**chol**). (ambos en el mismo DataFrame)

8. **Base de datos presión arterial en DataFame**. Dado un diccionario sobre hipertensión como el del ejemplo en el *docstring* del ejercicio 4, diseña una función que entre un diccionario donde la clave es el nombre de la persona y el valor es una lista con la edad y las presiones sistólica y diastólica (en unidades: mmHg), devuelva un dataFrame que incluya como columnas: 'Nombre', 'Edad', 'Sistólica', 'Diastólica' y una última columna calculada, etiquetada 'Diagnóstico', con datos categóricos con valores 'baja', 'normal', 'alta'. La presión arterial 'baja' la definimos así: si la presión sistólica es menor que 90 mmHg o la diastólica menor que 60 mmg. La presión 'alta' sería si la sistólica es mayor o igual a 140 mmHg o la diastólica es mayor o igual a 90 mmHg. En otras condiciones sería 'normal'. Ejemplo de diccionario de entrada y resultado.

Nota. Se considerará que no habrá casos donde la diferencia entre la presión sistólica y diastólica (presión de pulso) supere los 60 mmHg, es decir, no habrá casos donde simultáneamente se encuentre un paciente con la tensión diastólica menor que la 'baja' y la sistólica mayor que la 'alta'

```
dhiper = {'Maria': [40, 135, 90],'Nuria': [63, 141, 92], \
          'Jose': [47, 110, 59], 'Luis': [49, 146, 94], \
          'Oriol': [52, 130, 89], 'Carlos': [65, 125, 89], \
          'Pepe': [70, 130, 92]}
```

Resultado de la forma:

```
   Nombre  Edad  Sistólica  Diastólica Diagnóstico
0   Maria    40        135          90        alta
1   Nuria    63        141          92        alta
2    Jose    47        110          59        baja
3    Luis    49        146          94        alta
4   Oriol    52        130          89      normal
5  Carlos    65        125          89      normal
6    Pepe    70        130          92        alta
```