SPRINT 9

Nociones y conocimientos básicos de Python

Tascas Nivell 1:

Nivell 1

1. Calculadora de l'índex de massa corporal
   - Escriu una funció que calculi l'IMC ingressat per l'usuari/ària, és a dir, qui ho executi haurà d'ingressar aquestes dades. Pots obtenir més informació del seu càlcul en: Índice de masa corporal IMC que es y como se calcula.
   - La funció ha de classificar el resultat en les seves respectives categories.

Consell: Intenta validar les dades prèviament, perquè enviï un missatge d'advertència si les dades introduïdes per l'usuari/a no estan en el format adequat o no pren valors raonables.

Desarrollo:
A través del link del enunciado:
Índice de masa corporal (IMC): ¿qué es y cómo se calcula? - Muy Salud
Por primero he verificado los datos inserito con “try y except” luego he creado una condicional para definir en cual categoría apartienen los datos inserito dal usuario
He creado los parámetros de referencia de peso y altura con:
Página Oficial de Guinness World Records | Guinness World Records

Función.

```python
### Codigo función

def masa_corporal():
    """
    Calcula IMC a partir del peso (kg) y la altura (m).
    Valida los datos introducidos y devuelve el IMC (float) y la categoría correspondiente.
    """
    while True:
        try:
            peso = float(input('Inserir su peso en kilogramos , gracias:').replace(",","."))
            if 0.212 < peso < 635:
                break
            else:
                print("El peso debe estar entre 0.212 kg y 635 kg. Si no entra en ese rango, solo lo acepto si tienes un Guinness World Record")
        except ValueError:
            print("Valor introducido anómalo, recuerda debe ser un numero intero o decimal")


    while True:
        try:
            altura = float(input('Inserir su altura en metros: ').replace(",","."))
            if 0.24 < altura < 2.72:
                break
            else:
                print("Altura fuera de rango. Si no está entre 0.24 m y 2.72 m, solo te la doy por buena si sales en el Guinness World Records")
        except ValueError:
            print("Valor introducido anómalo ,recuerda debe ser un numero decimal")


    imc = round(peso / (altura ** 2),2)

    x = ""

    if imc < 18.5:
        Categoria = "Bajo peso"
    elif 18.5 <= imc < 25:
        Categoria = "Peso normal"
    elif 25 <= imc < 30:
        Categoria = "Sobrepeso"
    else:
        Categoria = "Obesidad"

    return Categoria,imc
```

0.0s

Execution.

```python
### Ejecución de código

imc_valoración = masa_corporal()

categoria,valor = imc_valoración


print("El valor de su IMC es:",valor,",","estàs en la categoria: ",categoria)
```

14.9s

```
El valor de su IMC es: 20.68 , estàs en la categoria:  Peso normal
```

2. Convertidor de temperatures

Existeixen diverses unitats de temperatura utilitzades en diferents contextos i regions. Les més comunes són Celsius (°C), Fahrenheit (°F) i Kelvin (K). També existeixen altres unitats com Rankine (°Ra) i Réaumur (°Re).

Selecciona almenys 2 conversors, de tal manera que en introduir una temperatura retorni, com a mínim, dues conversions, de manera que es puguin guardar (recorda que un print() no es pot guardar mai).

Consell: Intenta validar les dades prèviament, perquè enviï un missatge d'advertència si les dades introduides per l'usuari/a no estan en el format adequat.

(EXTRA): Pensa una manera d'emmagatzemar totes les possibles conversions en un sol objecte (Llista? Diccionari? DataFrame?) en comptes d'escriure molts if else en funció de la temperatura d'origen i la temperatura de destí.

Desarrollo:

Para resolver esta tasca he pensado en crear un mapa de conversiones de unidades de las temperaturas , para poder ver todas las varias posibilidades.
He encontrado esta pagina muy utile y interesante:
[Metric Conversion charts and calculators]

Función.

```python
def conversor_temperaturas():

    '''
    Esta función pregunta una temperatura y su unidad, entre:
    (C = Celsius, F = Fahrenheit , K =  Kelvin , RA = Rankine , RE = Réaumur).
    Verifica que los datos introducido sean corectos y en fin converte
    las otras unidades a través del mapa de conversiones.
    '''

    Mapa_conversiones = {
    "C": {
        "C":  lambda t: t,
        "F":  lambda t: round(t * 9/5 + 32,2),
        "K":  lambda t: round(t + 273.15,2),
        "RA": lambda t: round((t + 273.15) * 9/5,2),
        "RE": lambda t: round(t * 0.8,2)},
    "F": {
        "C":  lambda t: round((t - 32) * 5/9,2),
        "F":  lambda t: t,
        "K":  lambda t: round((t - 32) * 5/9 + 273.15,2),
        "RA": lambda t: round(t + 459.67,2),
        "RE": lambda t: round((t - 32) * 4/9,2)},
    "K": {
        "C":  lambda t: round(t - 273.15,2),
        "F":  lambda t: round((t - 273.15) * 9/5 + 32,2),
        "K":  lambda t: t,
        "RA": lambda t: round(t * 9/5,2),
        "RE": lambda t: round((t - 273.15) * 0.8,2)},
    "RA": {
        "C":  lambda t: round((t - 491.67) * 5/9,2),
        "F":  lambda t: round(t - 459.67,2),
        "K":  lambda t: round(t * 5/9,2),
        "RA": lambda t: t,
        "RE": lambda t: round((t - 491.67) * 4/9,2)},
    "RE": {
        "C":  lambda t: round(t * 1.25,2),
        "F":  lambda t: round(t * 2.25 + 32,2),
        "K":  lambda t: round(t * 1.25 + 273.15,2),
        "RA": lambda t: round((t * 1.25 + 273.15) * 9/5,2),
        "RE": lambda t: t}}


    while True:
        try:
            temperatura = round(float(input("Por favor inserir la temparatura para poder hacer conversión :").replace(",",".")),2)
            break
        except ValueError:
            print("Debes introducir un numero válido, entero o decimal, vuelvas a intentarlo")


    while True:
            unidad_origen = input("Inserir la unidad de la temperatura entre : 'C', 'F', 'K', 'RA', 'RE'").strip().upper()
            if unidad_origen in ("C", "F", "K", "RA", "RE"):
                break
            print("Unidad no válida, vuelve a intentarlo.")


    resultado = {}
    for unidad_destino, conversiones in Mapa_conversiones[unidad_origen].items():
        resultado[unidad_destino] = conversiones(temperatura)

    return resultado
```

0.0s

Execution.

```python
### Ejecución función


resultado_conversion = conversor_temperaturas()

print("Los resultados de las conversiones son:\n",resultado_conversion)
```

[19] ✓ 7.4s

```
Los resultados de las conversiones son:
{'C': 30.0, 'F': 86.0, 'K': 303.15, 'RA': 545.67, 'RE': 24.0}
```

Extra.

Para resolver esta parte he pensado en crear un dataframe y pedir al usuario cuantas conversiones quiere inserir,poniendo como columnas las varias unidades de temperaturas y como filas el números de conversiones que quiere efectuar el usuario

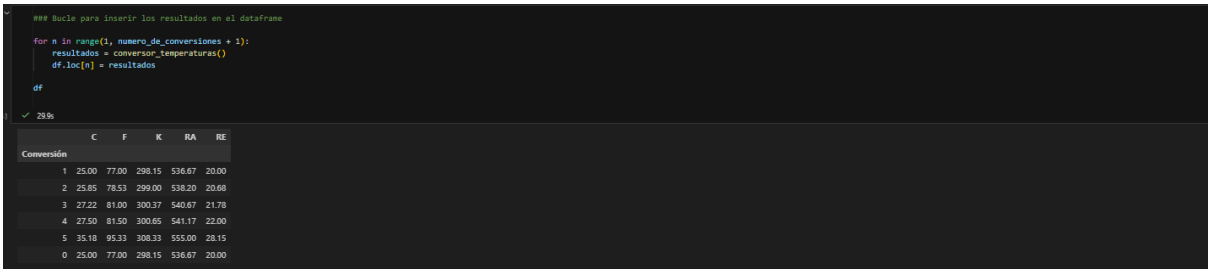
Creación dataframe.

```
### Extra , creo un dataframe para guardar los resultados :
### Index = numero de conversiones que quiere hacer el usuario
### Columnas = unidades_temp

import pandas as pd

unidades_temp = ["C", "F", "K", "RA", "RE"]

while True:
    try:
        numero_de_conversiones = int(input('por favor inserir el numero de las conversiones que se desea hacer :'))
        break
    except ValueError:
        print('Por favor vuelvas a intentarlos , has inserito un numero no valido')


df = pd.DataFrame(index=range(1, numero_de_conversiones + 1), columns=unidades_temp)

df.index.name = "Conversión"

df
```

✓ 2.4s

Conversión	C	F	K	RA	RE
1	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN
5	NaN	NaN	NaN	NaN	NaN

Ejecución.



```
### Bucle para inserir los resultados en el dataframe

for n in range(1, numero_de_conversiones + 1):
    resultados = conversor_temperaturas()
    df.loc[n] = resultados

df
```

✓ 29.9s

Conversión	C	F	K	RA	RE
1	25.00	77.00	298.15	536.67	20.00
2	25.85	78.53	299.00	538.20	20.68
3	27.22	81.00	300.37	540.67	21.78
4	27.50	81.50	300.65	541.17	22.00
5	35.18	95.33	308.33	555.00	28.15
0	25.00	77.00	298.15	536.67	20.00

3. Comptador de paraules d'un text.

Escriu una funció que donat un text, mostri les vegades que apareix cada paraula. Intenta que es gestionin totes les casuístiques possibles que facin que el programa no funcioni correctament.

(EXTRA): Quina és la llargada mitjana de les paraules del text que has escrit? “Hola com va?” hauria de retornar (4+3+2) / 3 = 3

Desarrollo:

He importado el módulo RE para limpiar de manera rápida y segura el texto introducido por el usuario y prepararlo para calcular el número de palabras y la media de letras.

La función SUB( ) me ha resultado especialmente útil y práctica.
NOTA: (Función pensada para texto en castellano)

[abecedario | Diccionario panhispánico de dudas | RAE - ASALE]

Función.

```python
### Para gestionar los posibles errores voy a importar RE,que a través de su función SUB,(IDIOMA CASTELLANO)
### me permite eliminar todo lo que pueda interferir : simbolos, numeros , signos, espacios multiples.

import re

def contador_de_palabras(texto):

    '''
    Comprueba que haya texto y lo limpia, eliminando :
    "simbolos , numeros, espacios multiples y signos".
    Asi da poder contar el numero de palabras y
    devolver un diccionario con el numero de veces que
    aparece cada palabra
    '''

    if texto is None:
        raise ValueError("Error,tienes que inserir un testo.")
    texto = str(texto).strip()
    if texto == "":
        raise ValueError("Se requiere un texto no vacio.")


    texto = texto.lower()
    texto_limpio = re.sub("[^a-záéíóúüñ ]"," ", texto)
    texto_limpio = re.sub(r"\s+", " ", texto_limpio).strip()
    texto_limpio = texto_limpio.split()

    if len(texto_limpio) == 0:
        return {}, 0.0

    dics_contador_palabras = {}

    for palabra in texto_limpio:
        if palabra in dics_contador_palabras:
            dics_contador_palabras[palabra] +=1
        else:
            dics_contador_palabras[palabra] = 1

    '''
    En esta parte, la función calcula
    la media de las letras de las palabras
    '''

    lista_n_letras = []
    tot_palabras = len(texto_limpio)

    for palabra in texto_limpio:
        lista_n_letras.append(len(palabra))


    avg_letras_palabras = round(sum(lista_n_letras)/tot_palabras,1)


    return dics_contador_palabras,avg_letras_palabras
```

Ejecución.

```python
#Execución de la función

resultado = contador_de_palabras(texto = input("Inserir un texto"))

print("Aquí podemos ver cuántas veces aparece cada palabra :\n",resultado[0])
print("La media de letras por palabras es :\n",resultado[1])
```

```
Aquí podemos ver cuántas veces aparece cada palabra :
 {'hola': 1, 'hermaño': 2, 'como': 1, 'estás': 1, 'hasta': 1, 'pronto': 1}
La media de letras por palabras es :
 5.4
```

4. Diccionari invers (amb possibilitat de duplicats)

Resulta que el client té una enquesta molt antiga que s'emmagatzema en un diccionari i els resultats els necessita al revés, és a dir, intercanviant les claus i els valors. Els valors i claus en el diccionari original són únics; si aquest no és el cas, la funció hauria d'imprimir un missatge d'advertiment, juntament amb una llista amb els valors associats a la clau repetida.

Fixa't en el següents exemples per veure l'output esperat.

Desarrollo:

Esta me pareció la tarea más sencilla. A través de la función ITEMS() propia de los diccionarios, he creado un bucle con una condición que me ha permitido devolver los dos escenarios posibles: diccionario con valores duplicados y sin ellos.

Función.

```
### función

def dics_inverso(dics):
    '''
    Esta función recibe un diccionario y lo invierte,
    siempre que no haya duplicados en los valores que pasarán a ser claves.
    En caso de que existan duplicados, la función muestra un mensaje de error
    y devuelve un diccionario con los valores repetidos y las claves asociadas.
    '''

    dic_inverso = {}

    for clave, valor in dics.items():
        if valor not in dic_inverso:
            dic_inverso[valor] = [clave]
        else:
            dic_inverso[valor].append(clave)

    for valor, claves in dic_inverso.items():
        if len(claves) > 1:
            print("Advertencia! La clave", valor, "tiene valores duplicados:", claves)

    return dic_inverso
```

Ejecución.

```
### Ejecución primero escenario sin duplicados.

a = {"A": 10, "B": 20, "C": 30}

dics_inverso(a)
```

```
{10: ['A'], 20: ['B'], 30: ['C']}
```

```
### Ejecución secundo escenario con duplicados.

b = {"A": 1, "B": 2, "C": 1, "D": 3, "E": 2}

dics_inverso(b)
```

```
Advertencia! La clave 1 tiene valores duplicados: ['A', 'C']
Advertencia! La clave 2 tiene valores duplicados: ['B', 'E']

{1: ['A', 'C'], 2: ['B', 'E'], 3: ['D']}
```

Tascas Nivel 2:

Nivell 2

1. Comptador i endreçador de paraules d'un text.

El client va quedar content amb el comptador de paraules, però ara vol llegir arxius TXT i que calculi la freqüència de cada paraula ordenades dins de les entrades habituals del diccionari segons la lletra amb la qual comencen, és a dir, les claus han d'anar de la A a la Z i dins de la A hem d'anar de la A la Z. Per exemple, per a l'arxiu "tu_me_quieres_blanca.txt" la sortida esperada seria:

```
{'a': {'a': 3,
  'agua': 1,
  'al': 2,
  'alba': 4,
  'alcobas': 1,
  'alimenta': 1,
  'alma': 1,
  'amarga': 1,
  'azucena': 1},
 'b': {'baco': 1,
  'banquete': 1,
  'bebe': 1,
  'blanca': 3,
  'boca': 1,
  'bosques': 1,
  'buen': 1},
 'c': {'cabañas': 1,
  'carnes': 2,
  'casta': 3,
  'cerrada': 1,
  'con': 4,
  'conservas': 1,
  'copas': 1,
  'corola': 1,
  'corriste': 1,
...
  'tornadas': 1,
  'tú': 8},
 'u': {'un': 1, 'una': 1},
 'v': {'vestido': 1, 'vete': 1, 'vive': 1},
 'y': {'y': 5}}
```

Desarrollo:

Para visualizar el texto he utilizado la librería Pandas y posteriormente lo he mostrado a través de un dataframe.
Para crear la función me he basado en el idioma del texto del archivo, es decir , el castellano.
He dividido la función en dos partes:

La primera, donde limpio los datos y ordeno la lista de palabras obtenidas;

La segunda donde creo el diccionario que debo devolver , junto con los bucles que me permiten insertar las claves en el diccionario y sus respectivas palabra y recuentos.

Importo el texto:

```
### Import Pandas

import pandas as pd

archivo_txt = r"C:\Users\eirad\Desktop\IT ACADEMY\Python\Especializacion\File texto\tu me quieres blanca.txt"

df = pd.read_csv(archivo_txt,sep = "\t",header = None)

df
```

[64] Python

	0
0	Tú me quieres alba,
1	me quieres de espumas,
2	me quieres de nácar.
3	Que sea azucena
4	Sobre todas, casta.
5	De perfume tenue.
6	Corola cerrada .
7	Ni un rayo de luna
8	filtrado me haya.
9	Ni una margarita
10	se diga mi hermana.
11	Tú me quieres nívea,
12	tú me quieres blanca,
13	tú me quieres alba.
14	Tú que hubiste todas
15	las copas a mano,
16	de frutos y mieles

Función:

```python
### Codigo Función, importo RE para limpiar de forma rapida y segura

import re

def creación_dict(df):
    '''
    Primera parte:
    Limpio el texto para dejarlo preparado y creo
    una lista con todas las palabras, ordenadas de "a-z".
    '''

    lista_palabras_totales = []

    for fila in df.iloc[:,0]:
        fila = fila.lower()
        fila_limpia = re.sub(r"[^a-záéíóúüñ ]", " ", fila)
        fila_limpia = re.sub(r"\s+", " ", fila_limpia).strip()

        lista_palabras = fila_limpia.split()

        lista_palabras_totales.extend(lista_palabras)


    lista_palabras_totales.sort()


    '''
    Segunda parte:
    Creo una lista "abecedario" para generar las claves del diccionario.
    Luego, mediante un bucle, asigno cada palabra a la letra correspondiente
    e incremento su número de repeticiones.
    '''


    abecedario = []

    for palabra in lista_palabras_totales:
        if palabra[0] not in abecedario:
            abecedario.append(palabra[0])

    abecedario.sort()


    diccionario = {letra : {} for letra in abecedario}


    for palabra in lista_palabras_totales:
        letra = palabra[0]
        if letra in diccionario:
            if palabra in diccionario[letra]:
                diccionario[letra][palabra] += 1
            else:
                diccionario[letra][palabra] = 1

    '''
    Toma el diccionario de esa letra,lo conviérte en lista,
    ordéna alfabéticamente y vuelve a convertirlo en diccionario.
    '''

    for letra in diccionario:
        diccionario[letra] = dict(sorted(diccionario[letra].items()))


    return diccionario
```

0.0s

Ejecución.

```python
### Ejecución

creación_dict(df)
```

✓ 0.0s

```
{'a': {'a': 3,
  'agua': 1,
  'al': 2,
  'alba': 4,
  'alcobas': 1,
  'alimenta': 1,
  'alma': 1,
  'amarga': 1,
  'azucena': 1},
 'b': {'baco': 1,
  'banquete': 1,
  'bebo': 1,
  'blanca': 3,
  'boca': 1,
  'bosques': 1,
  'buen': 1},
 'c': {'cabañas': 1,
  'carmes': 2,
  'casta': 3,
  'cerrada': 1,
  'con': 4,
  'conservas': 1,
  'copas': 1,
  'corola': 1,
  'corriste': 1,
...
  'tornadas': 1,
  'tú': 8},
 'u': {'un': 1, 'una': 1},
 'v': {'vestido': 1, 'vete': 1, 'vive': 1},
 'y': {'y': 5}}
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

2. Conversió de tipus de dades.

El client rep una llista de dades i necessita generar dues llistes, la primera on estaran tots els elements que es van poder convertir en flotants i l'altra on estan els elements que no es van poder convertir. Exemple de la llista que rep el client:

['1.3', 'one' , '1e10' , 'seven', '3-1/2', ('2',1,1.4,'not-a-number'), [1,2,'3','3.4']]

```
conversion([ '1.3', 'one' , '1e10' , 'seven', '3-1/2', ('2',1,1.4,'not-a-number'), [1,2,'3','3.4']])
```

```
([1.3, 10000000000.0, 2.0, 1.0, 1.4, 1.0, 2.0, 3.0, 3.4],
 ['one', 'seven', '3-1/2', 'not-a-number'])
```

Desarrollo:

La clave estuvo en encontrar una forma de recorrer cualquier estructura interna (listas, diccionarios, tuplas o sets) y usar TYPE () para detectar su tipo. Primero definí una función interna que sigue iterando hasta extraer todos los valores, sin importar la profundidad. En fin clasifiqué esos valores en dos listas: los que pueden convertirse y los que no.

Función.

```python
### Función , utilizar TYPE para verificar tipo da datos y función interna a la principal

lista_usuario = ['1.3', 'one', '1e10', 'seven', '3-1/2', ('2', 1, 1.4, 'not-a-number'), [1, 2, 3, '3.4']]


def clasificación_datos(lista):
    lista_datos_float = []
    lista_datos_no_convertibles = []

    '''
    La función recorre toda la estructura (listas, tuplas, sets, dics),
    extrae los valores internos sin importar la profundidad
    e intenta convertir cada uno a float,
    separando los convertibles de los no.
    '''

    def sub_elementos(elemento):
        if type(elemento) in (list,dict,tuple,set):
            for sub_valor in elemento:
                sub_elementos(sub_valor)
        else:
            try:
                float(elemento)
                lista_datos_float.append(float(elemento))
            except:
                lista_datos_no_convertibles.append(elemento)

    sub_elementos(lista)


    return lista_datos_float,lista_datos_no_convertibles
```

Ejecución.

```python
x = clasificación_datos(lista_usuario)


lista_datos_float,lista_datos_no_convertibles = x

print('lista de datos float:\n',lista_datos_float)
print('lista de datos no convertibles:\n',lista_datos_no_convertibles)
```

```
0.0s
lista de datos float:
 [1.3, 10000000000.0, 2.0, 1.0, 1.4, 1.0, 2.0, 3.0, 3.4]
lista de datos no convertibles:
 ['one', 'seven', '3-1/2', 'not-a-number']
```

Nivell 3

1. Generador de Contrasenyes

Explora el funcionament del mòdul random de la llibreria numpy. 🔗 Random module in NumPy

En aquest punt, el client ha detectat un problema amb les contrasenyes que utilitzen els seus treballadors. Asdf1234, dates d'aniversari o similars. Per a solucionar-ho ens ha encarregat una funció de Python que generi contrasenyes més segures. La funció ha de dependre dels següents paràmetres:

- longitud (int): Longitud de la contrasenya
- majuscules (bool = True): Si hi ha d'aparèixer majúscules
- minuscules (bool = True): Si hi ha d'aparèixer minúscules
- numeros (bool = True): Si hi ha d'aparèixer números
- signes (bool = False): Si hi ha d'aparèixer caràcters especials (,-$? o similars)

Així doncs, si executem la funció de la següent manera:

crear_contrasenya(10, True, True, True, True)

Hauríem d'obtenir un output (que hem de poder guardar) de l'estil:

9Er,5Vn8P$

Assegura't que es compleixin tots els criteris, i que aquestes contrasenyes siguin realment aleatòries.

(EXTRA) Explora com podríem fer que la funció copiés la contrasenya automàticament al porta papers de l'ordinador (com si l'haguéssim seleccionat i fet ctrl+copy).

Desarrollo:

Por primero he importado las librerías necesarias para crear la función,luego a través de los métodos propios de NumPy he creado el código

Función.

```python
# importo las varias librerias y modulos

import numpy as np

import string
```

0.0s

```python
def generador_contrasenas(longitud, letras_may=True, letras_min=True, numeros=True, simbolos=False):
    '''
    crea los grupos de caracteres:
    -letras mayuscola y minuscola
    -numeros y simbolos
    '''

    letras_min_set = list(string.ascii_lowercase) + ["Ñ"]
    letras_may_set = list(string.ascii_uppercase) + ["Ñ"]
    numeros_set = list("0123456789")
    simbolos_set = list("@#$%&/()=+-_"<>[]{}|\\^~*¨€.,;:!?¿¡'")

    '''
    verifica qué tipos de caracteres quiere incluir el usuario
    en su contraseña
    '''

    contrasena = []

    if letras_may:
        contrasena += letras_may_set
    if letras_min:
        contrasena += letras_min_set
    if numeros:
        contrasena += numeros_set
    if simbolos:
        contrasena += simbolos_set

    if not contrasena:
        raise ValueError("Debes activar al menos un tipo de carácter")

    '''
    en fin crea la contraseña, mezcla los caracteres y
    devuelve uno string
    '''


    rng = np.random.default_rng()
    resultado = [rng.choice(contrasena) for _ in range(longitud)]
    rng.shuffle(resultado)

    return "".join(resultado)
```

0.0s

Ejecución.

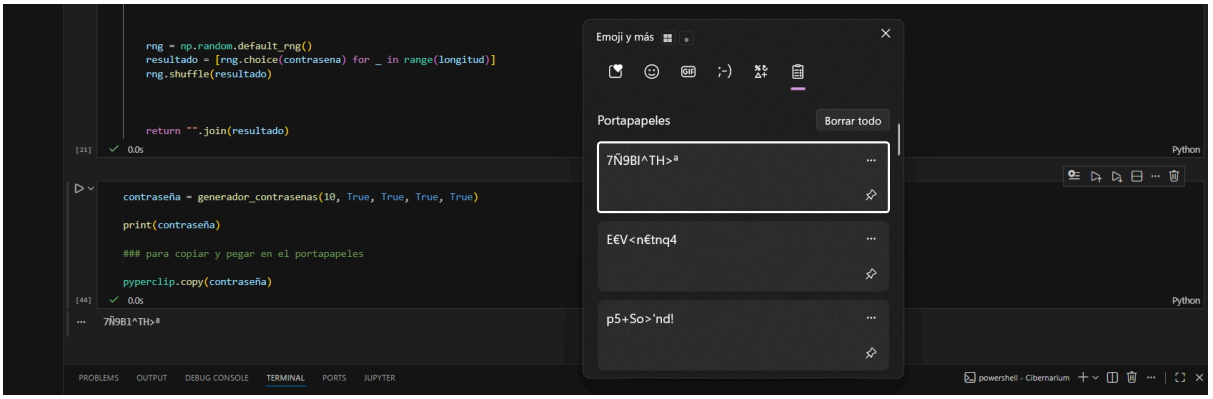
```python
print(generador_contrasenas(10, True, True, True, True))
```

0.0s

```
>4>AZ\y}^M
```

Extra.

A través de la librería Pyperclip y el método copy() , guarda directamente la contraseña creada en el portapapeles.



2. Processament de dades simple

Una companya de feina ens ha demanat un favor, aprofitant que sap que estem aprenent a programar. Té un històric de partits de futbol català en un fitxer, on hi ha emmagatzemat els noms dels equips i els resultats. Necessita que processem les dades de manera automàtica, per extreure'n els resultats que necessita.

Utilitza el fitxer "historic_partits.txt"

Necessita un programa que retorni:

- El nombre total de gols que ha fet cada equip.
- El nom de l'equip més golejador.
- El nom de l'equip més golejat
- La classificació global (cada victòria: 3 pts, empat 1 pts, derrota 0 pts)

Com a referència, els equips amb més punts serien els següents:

	partits guanyats	partits empatats	partits perduts	gols	gols encaixats	punts
Girona FC	31	3	13	139	94	96
Llagostera	29	7	20	159	142	94
Sabadell	26	7	15	141	121	85
Cornellà	25	7	22	147	146	82
RCD Espanyol	23	11	21	131	144	80
Figueres	23	10	23	161	154	79
Lleida Esportiu	23	6	23	129	133	75
Terrassa	20	14	23	147	164	74
FC Barcelona	22	7	15	125	115	73

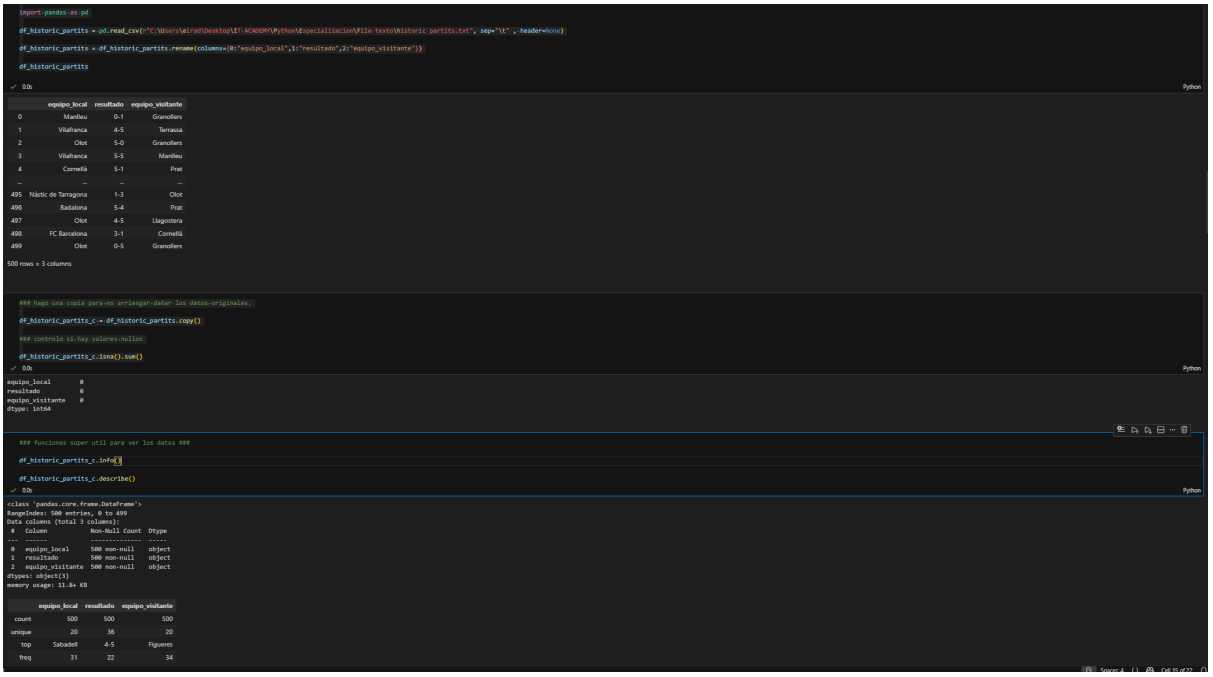
Pots utilitzar el format que et sigui més convenient.

Desarrollo:

Esta es seguramente la tasca más interesante de todas, he tenido que organizar el trabajo en diferentes bloque de código que podemos resumir en 3 partes:

1.Importación y control de los datos
2.Preparación de los datos para facilitar la creación de la función
3.Creación de la función y su ejecución

Primera parte.



```
import pandas as pd

df_historic_partits = pd.read_csv("C:\Users\eirad\Desktop\IT ACADEMY\Python\Especialitzacion\File texto\historic partits.txt", sep="\t" , header=None)

df_historic_partits = df_historic_partits.rename(columns={0:"equipo_local",1:"resultado",2:"equipo_visitante"})

df_historic_partits
```

0.0s — Python

	equipo_local	resultado	equipo_visitante
0	Manlleu	0-1	Granollers
1	Vilafranca	4-5	Terrassa
2	Olot	5-0	Granollers
3	Vilafranca	5-5	Manlleu
4	Cornellà	5-1	Prat
...	...	...	...
495	Nàstic de Tarragona	1-3	Olot
496	Badalona	5-4	Prat
497	Olot	4-5	Llagostera
498	FC Barcelona	3-1	Cornellà
499	Olot	0-5	Granollers

500 rows × 3 columns

```
### hago una copia para no arriesgar-dañar los datos originales.

df_historic_partits_c = df_historic_partits.copy()

### controlo si hay valores nulos

df_historic_partits_c.isna().sum()
```

0.0s — Python

```
equipo_local      0
resultado         0
equipo_visitante  0
dtype: int64
```

```
### funciones super util para ver los datos ###

df_historic_partits_c.info()

df_historic_partits_c.describe()
```

0.0s — Python

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   equipo_local     500 non-null    object
 1   resultado        500 non-null    object
 2   equipo_visitante 500 non-null    object
dtypes: object(3)
memory usage: 11.8+ KB
```

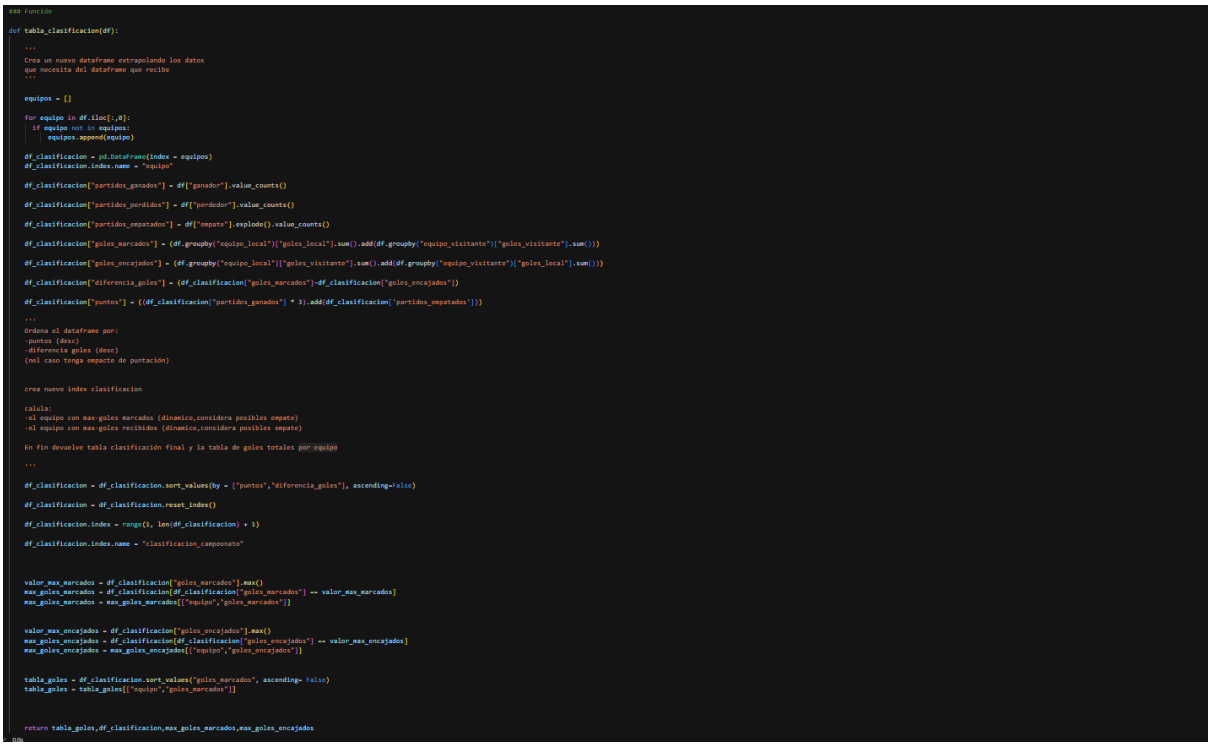
	equipo_local	resultado	equipo_visitante
count	500	500	500
unique	20	36	20
top	Sabadell	4-5	Figueres
freq	31	22	34

Segunda parte.

```
### Preparo los datos da inserir en la función:

# divido los goles del resultado

df_historic_partits_c[["goles_local", "goles_visitante"]] = df_historic_partits_c["resultado"].str.split("-", expand=True)

# ganador

df_historic_partits_c["ganador"] = None


def ganador(fila):
    if int(fila["goles_local"]) > int(fila["goles_visitante"]):
        return fila.iloc[0]
    elif int(fila["goles_local"]) < int(fila["goles_visitante"]):
        return fila.iloc[2]
    else:
        return None


# perdedor

df_historic_partits_c["perdedor"] = None


def perdedor(fila):
    if int(fila["goles_local"]) < int(fila["goles_visitante"]):
        return fila.iloc[0]
    elif int(fila["goles_local"]) > int(fila["goles_visitante"]):
        return fila.iloc[2]
    else:
        return None


# empate

df_historic_partits_c[["empate"]] = None


def empate(fila):
    if int(fila["goles_local"]) == int(fila["goles_visitante"]):
        return fila.iloc[0], fila.iloc[2]


### aplico

df_historic_partits_c["ganador"] = df_historic_partits_c.apply(ganador, axis=1)
df_historic_partits_c["perdedor"] = df_historic_partits_c.apply(perdedor, axis=1)
df_historic_partits_c["empate"] = df_historic_partits_c.apply(empate, axis=1)


### transformo tipo columna "goles" en INT

df_historic_partits_c["goles_local"] = df_historic_partits_c["goles_local"].astype(int)

df_historic_partits_c["goles_visitante"] = df_historic_partits_c["goles_visitante"].astype(int)


df_historic_partits_c
```

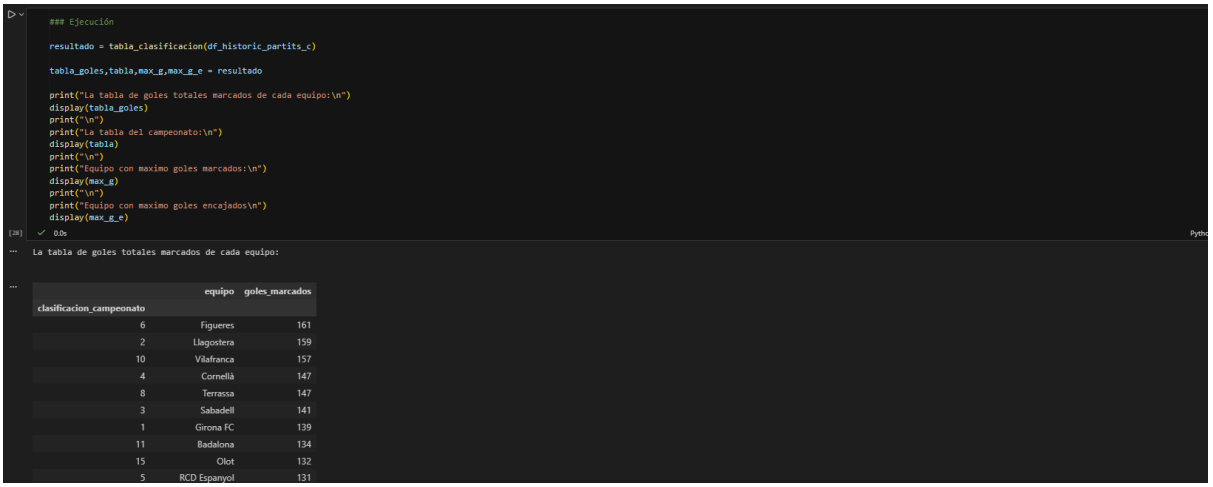
0.0s

Tercera parte.



```
### Función

def tabla_clasificacion(df):

    '''
    Crea un nuevo dataframe extrapolando los datos
    que necesita del dataframe que recibe
    '''

    equipos = []

    for equipo in df.iloc[:,0]:
        if equipo not in equipos:
            equipos.append(equipo)

    df_clasificacion = pd.DataFrame(index = equipos)
    df_clasificacion.index.name = "equipo"

    df_clasificacion["partidos_ganados"] = df["ganador"].value_counts()

    df_clasificacion["partidos_perdidos"] = df["perdedor"].value_counts()

    df_clasificacion["partidos_empatados"] = df["empate"].explode().value_counts()

    df_clasificacion["goles_marcados"] = (df.groupby("equipo_local")["goles_local"].sum().add(df.groupby("equipo_visitante")["goles_visitante"].sum()))

    df_clasificacion["goles_encajados"] = (df.groupby("equipo_local")["goles_visitante"].sum().add(df.groupby("equipo_visitante")["goles_local"].sum()))

    df_clasificacion["diferencia_goles"] = (df_clasificacion["goles_marcados"]-df_clasificacion["goles_encajados"])

    df_clasificacion["puntos"] = ((df_clasificacion["partidos_ganados"] * 3).add(df_clasificacion['partidos_empatados']))

    '''
    Ordena el dataframe por:
    -puntos (desc)
    -diferencia goles (desc)
    (nel cass tenga empacte de puntación)


    crea nuevo índex clasificacion

    calula:
    -el equipo con max-goles marcados (dinamico,considera posibles empate)
    -el equipo con max-goles recibidos (dinamico,considera posibles empate)

    En fin devuelve tabla clasificación final y la tabla de goles totales por equipo

    '''

    df_clasificacion = df_clasificacion.sort_values(by = ["puntos","diferencia_goles"], ascending=False)

    df_clasificacion = df_clasificacion.reset_index()

    df_clasificacion.index = range(1, len(df_clasificacion) + 1)

    df_clasificacion.index.name = "clasificacion_campeonato"



    valor_max_marcados = df_clasificacion["goles_marcados"].max()
    max_goles_marcados = df_clasificacion[df_clasificacion["goles_marcados"] == valor_max_marcados]
    max_goles_marcados = max_goles_marcados[["equipo","goles_marcados"]]


    valor_max_encajados = df_clasificacion["goles_encajados"].max()
    max_goles_encajados = df_clasificacion[df_clasificacion["goles_encajados"] == valor_max_encajados]
    max_goles_encajados = max_goles_encajados[["equipo","goles_encajados"]]


    tabla_goles = df_clasificacion.sort_values("goles_marcados", ascending= False)
    tabla_goles = tabla_goles[["equipo","goles_marcados"]]



    return tabla_goles,df_clasificacion,max_goles_marcados,max_goles_encajados
0.0s
```

Ejecución.



```
### Ejecución

resultado = tabla_clasificacion(df_historic_partits_c)

tabla_goles,tabla,max_g,max_g_e = resultado

print("La tabla de goles totales marcados de cada equipo:\n")
display(tabla_goles)
print("\n")
print("La tabla del campeonato:\n")
display(tabla)
print("\n")
print("Equipo con maximo goles marcados:\n")
display(max_g)
print("\n")
print("Equipo con maximo goles encajados\n")
display(max_g_e)
```

[28] ✓ 0.0s

```
La tabla de goles totales marcados de cada equipo:
```

clasificacion_campeonato	equipo	goles_marcados
6	Figueres	161
2	Llagostera	159
10	Vilafranca	157
4	Cornellà	147
8	Terrassa	147
3	Sabadell	141
1	Girona FC	139
11	Badalona	134
15	Olot	132
5	RCD Espanyol	131

La tabla del campeonato:

clasificacion_campeonato	equipo	partidos_ganados	partidos_perdidos	partidos_empatados	goles_marcados	goles_encajados	diferencia_goles	puntos
1	Girona FC	31	13	3	139	94	45	96
2	Llagostera	29	20	7	159	142	17	94
3	Sabadell	26	15	7	141	121	20	85
4	Cornellà	25	22	7	147	146	1	82
5	RCD Espanyol	23	21	11	131	144	-13	80
6	Figueres	23	23	10	161	154	7	79
7	Lleida Esportiu	23	23	6	129	133	-4	75
8	Terrassa	20	23	14	147	164	-17	74
9	FC Barcelona	22	15	7	125	115	10	73
10	Vilafranca	20	25	12	157	172	-15	72
11	Badalona	18	16	14	134	124	10	68
12	Reus Deportiu	20	16	8	119	111	8	68
13	Nàstic de Tarragona	20	27	8	130	148	-18	68
14	Granollers	21	24	4	122	117	5	67
15	Olot	18	26	10	132	144	-12	64
16	Sant Andreu	17	25	11	123	134	-11	62
17	Manlleu	16	18	9	106	118	-12	57
18	Prat	16	21	8	106	111	-5	56
19	Cerdanyola	17	28	5	117	133	-16	56
20	Europa	12	16	5	93	93	0	41

Equipo con maximo goles marcados:

clasificacion_campeonato	equipo	goles_marcados
6	Figueres	161

Equipo con maximo goles encajados

clasificacion_campeonato	equipo	goles_encajados
10	Vilafranca	172

Conclusiones:

Para la resolución fue esencial utilizar métodos propios de pandas como apply(), add(), explode(), value_counts(), isin() y astype().

Una vez terminado, seguí preguntándome cómo podía mejorarlo, así que quise añadir un pequeño extra: el problema de la gestión de los empates en la clasificación.
Busqué el reglamento oficial de la **Federación Catalana de Fútbol** y apliqué las reglas que encontré para los casos de empate entre equipos.

FCF | Estatuts i Reglament General

Resumen de reglas específicas — Criterios de desempate (Art. 220è)
Cuando dos o más equipos terminan empatados a puntos en la clasificación, se aplica el siguiente orden jerárquico y excluyente:

1. **Puntos particulares:**
   Puntuación obtenida exclusivamente en los enfrentamientos directos entre los equipos empatados.
2. **Diferencia de goles particular:**
   Balance de goles a favor y en contra únicamente en esos duelos directos.
3. **Diferencia de goles general:**
   Diferencia total entre goles marcados y encajados a lo largo de toda la competición.
4. **Goles marcados:**
   Mayor número total de goles anotados durante el campeonato.
5. **Empate absoluto:** Si la igualdad persiste tras aplicar todos los criterios anteriores, se debe disputar un partido de desempate en campo neutral.

Los cambios hechos en el código de la Función:

```
'''
Nueva parte del código para gestionar empates en la clasificación:
Crea dos nuevas columnas que representan los puntos y la diferencia
de goles obtenidos únicamente en los enfrentamientos directos entre
equipos con la misma puntuación total. Luego agrupa los equipos por
puntos y calcula estos valores para cada uno.

'''

df_clasificacion[["puntos_particulares", "dif_goles_particular"]] = 0

for _, tabla_empatados in df_clasificacion.groupby("puntos"):
    if len(tabla_empatados) <= 1:
        continue


    equipos_empatados = tabla_empatados.index

    partidos_directos = df[df["equipo_local"].isin(equipos_empatados) &df["equipo_visitante"].isin(equipos_empatados)]


    for equipo in equipos_empatados:

        victorias = (partidos_directos["ganador"] == equipo).sum()

        empates = ((partidos_directos["empate"] == True) & ((partidos_directos["equipo_local"] == equipo) | (partidos_directos["equipo_visitante"] == equipo))).sum()

        puntos_particulares = victorias * 3 + empates

        goles_favor = (partidos_directos.loc[partidos_directos["equipo_local"] == equipo, "goles_local"].sum() +
                       partidos_directos.loc[partidos_directos["equipo_visitante"] == equipo, "goles_visitante"].sum())

        goles_contra = (partidos_directos.loc[partidos_directos["equipo_local"] == equipo, "goles_visitante"].sum() +
                        partidos_directos.loc[partidos_directos["equipo_visitante"] == equipo, "goles_local"].sum())

        df_clasificacion.loc[equipo, "puntos_particulares"] = puntos_particulares
        df_clasificacion.loc[equipo, "dif_goles_particular"] = goles_favor - goles_contra
```

Ejecución:

```
resultado = tabla_clasificacion(df_historic_partits_c)

tabla_goles,tabla,max_g,max_g_c = resultado

print("La tabla de goles marcados totales de cada equipo:\n")
display(tabla_goles)
print("\n")
print("La tabla del campeonato:\n")
display(tabla)
print("\n")
print("Equipo con maximo goles marcados:\n")
display(max_g)
print("\n")
print("Equipo con maximo goles encajados\n")
display(max_g_c)
```

```
✓ 0.0s
```

```
La tabla de goles marcados totales de cada equipo:
```

clasificacion_campeonato	equipo	goles_marcados
6	Figueres	161
2	Llagostera	159
10	Vilafranca	157
4	Cornellà	147
8	Terrassa	147
3	Sabadell	141
1	Girona FC	139
12	Badalona	134
15	Olot	132
5	RCD Espanyol	131
11	Nàstic de Tarragona	130
7	Lleida Esportiu	129
9	FC Barcelona	125
16	Sant Andreu	123
14	Granollers	122
13	Reus Deportiu	119
19	Cerdanyola	117
17	Manlleu	106
18	Prat	106
20	Europa	93

```
La tabla del campeonato:
```

clasificacion_campeonato	equipo	partidos_ganados	partidos_perdidos	partidos_empatados	goles_marcados	goles_encajados	diferencia_goles	puntos	puntos_particulares	dif_goles_particular
1	Girona FC	31	13	3	139	94	45	96	0	0
2	Llagostera	29	20	7	159	142	17	94	0	0
3	Sabadell	26	15	7	141	121	20	85	0	0
4	Cornellà	25	22	7	147	146	1	82	0	0
5	RCD Espanyol	23	21	11	131	144	-13	80	0	0
6	Figueres	23	23	10	161	154	7	79	0	0
7	Lleida Esportiu	23	23	6	129	133	-4	75	0	0
8	Terrassa	20	23	14	147	164	-17	74	0	0
9	FC Barcelona	22	15	7	125	115	10	73	0	0
10	Vilafranca	20	25	12	157	172	-15	72	0	0
11	Nàstic de Tarragona	20	27	6	130	148	-18	68	12	3
12	Badalona	18	16	14	134	124	10	68	1	-1
13	Reus Deportiu	20	16	8	119	111	8	68	0	-2
14	Granollers	21	24	4	122	117	5	67	0	0
15	Olot	18	26	10	132	144	-12	64	0	0
16	Sant Andreu	17	25	11	123	134	-11	62	0	0
17	Manlleu	16	18	9	106	118	-12	57	0	0
18	Prat	16	21	8	106	111	-5	56	3	1
19	Cerdanyola	17	28	5	117	133	-16	56	3	-1