



PYTHON PANDA

UT - 2. 1. PYTHON PANDA

Apellidos: Alves Mascareña

Nombre: María

Nº PC: 00

Centro: I.E.S. La Marisma (Huelva)

Curso: 2º Técnico Superior en Desarrollo de Aplicaciones Web

Asignatura: Horas de Libre Configuración – Python

Profesor: Gonzalo Cañadillas Rueda

Fecha: 29/11/2024

Sumario

Descripción del proyecto.....	3
Preparación de datos.....	3
Inicialización de datos.....	3
Funciones.....	4
Preparación de datos.....	10
Actividades.....	11
Actividad 1.....	11
Actividad 2.....	11
Actividad 3.....	12
Actividad 4.....	13
Actividad 5.....	14

Descripción del proyecto

Este proyecto consiste en la simulación de una competición de saltos, donde contamos con una serie de participantes que realizarán un total de tres saltos.

Cada salto es evaluado individualmente por cinco jueces, además, cada salto cuenta con un grado de dificultad que se empleará a la hora de calcular el total

Durante el proyecto surgirán diversos casos en los que tendremos que mostrar los datos de diferentes formas

Usaremos las librerías pandas para la creación de tablas (*dataframes*) y random para generar números aleatorios

```
import random as rand
import pandas as pd
```

Preparación de datos

Antes de comenzar a construir la estructura de nuestros dataframes, vamos a crear funciones que nos permitan generar los valores que nos hacen falta y así tenerlos listos a la hora de crear nuestras estructuras de datos

Inicialización de datos

► Grado de dificultad de cada salto

Crearemos una lista donde guardaremos 3 valores generados aleatoriamente entre 0 y 1. Con un for establecemos el rango de 0 a 3 y le añadimos a nuestra lista los valores aleatorios usando la librería random, multiplicamos y dividimos entre 10 para obtener valores del tipo $\rightarrow 0.0, 0.4, 0.9, \dots$

```
""" """ """ PREPARACION DE DATOS """ """ """ """
grado_dificultad = [] #Grado de dificultad de los 3 saltos
for i in range(0,3):
    grado_dificultad.append(int(rand.random()*10)/10)
```

► Datos personales de los saltadores

Simplemente es una lista de listas ya definida con datos sobre los participantes

```
saltadores = [
    ["Raúl", "Capablanca", "Federación Cubana", 2780],
    ["José", "Martínez", "Federación Bética", 2638],
    ["Iván", "Gómez", "Federación Suiza", 2809],
    ["Pepe", "Pepito", "Federación Pepe", 2323],
    ["Ubuntu", "Proxmox", "Federación Torretas", 2080],
    ["Skibidi", "Sigma", "Federación Rizzler", 2679],
    ["Impac", "Tum", "Federación PC", 2180]
```

► Lista de posibles notas

Para generar notas aleatorias de 0 a 10 con saltos de 0.5 he decidido crear una lista con todas las notas posibles con un for. De esta lista obtendremos aleatoriamente un elemento (*una nota*) para cada salto más adelante

```
posibles_notas = []
for i in range(0,10):
    posibles_notas.append(i)
    posibles_notas.append(i+0.5)
    i-=0.5
posibles_notas.append(10)
```

Funciones

Teniendo en cuenta que se nos presentarán casos en los que tendremos que repetir procedimientos ya hechos pero con otros datos, vamos a realizar nuestras funciones para que puedan adaptarse a todos los casos necesarios

► calcular_todos_salto

Esta función nos devolverá una lista de listas de tres valores cada una, siendo estos el total de las 5 puntuaciones de cada salto de cada saltador, nos podemos hacer una idea de su estructura así:

```
saltos_X_de_todos_saltadores = [ #ANTIGUO saltos
    #Saltador 1 -> [salto_1, salto_2, salto_3],
    #Saltador 2 -> [salto_1, salto_2, salto_3], ...
]
```

Horas de Libre Configuración – PHP – CFGS 2º DAW – Alves Mascareña, María

La función recibe dos parámetros:

- lista_saltadores = Una lista con los datos de los saltadores, esto nos servirá para saber cuántos saltadores participan
- saltos = La cantidad de saltos que se van a realizar

```
#Todos los saltos de todos los saltadores
def calcular_todos_saltos(lista_saltadores, saltos):
    saltos_X_de_todos_saltadores = [
        #Saltador 1 -> [salto_1, salto_2, salto_3],
        #Saltador 2 -> [salto_1, salto_2, salto_3], ....
    ]
    for i in lista_saltadores:
        saltos_X = [] #nota final de los X saltos
        for salto in range(saltos):
            puntuaciones_salto = [] #las 5 puntuaciones de 1 salto
            for juez in range(5):
                puntuaciones_salto.append(rand.choice(posibles_notas))
            puntuaciones_salto.sort()
            puntuacion_salto = 0 #suma de las puntuaciones
            for punt_valido in range(1, len(puntuaciones_salto)-1):
                puntuacion_salto += puntuaciones_salto[punt_valido]
            saltos_X.append(puntuacion_salto)
        saltos_X_de_todos_saltadores.append(saltos_X)
    return saltos_X_de_todos_saltadores
```

Empezamos creando una lista que será donde almacenaremos los datos a devolver

Iniciamos un bucle que recorrerá la lista de los saltadores, no utilizaremos el iterador pero así sabremos cuántos son

Preparamos una lista donde guardaremos las notas finales de el número de saltos introducidos

Iniciamos otro bucle que recorra según el parámetro de saltos introducidos

Preparamos otra lista donde guardaremos las 5 notas de un salto

Generemos otro bucle que iterará 5 veces y generará una nota aleatoria que se añadirá a la última lista creada

Una vez la lista rellena se ordenará y preparamos una variable donde iremos sumando la puntuación final

Hacemos un bucle que recorra del 2º elemento (*no contamos la nota más baja*) hasta el penúltimo (*no contamos la nota más alta*) y la sumamos a la variable de puntuación total

Cuando finalice, esa nota se añadirá a la lista de los X saltos (*saltos_X*)

Cuando el bucle de la cantidad de saltos finalice, esta lista se añadirá a la lista creada al principio, que cuando el bucle de todos los participantes finalice, se devolverá, habiendo conseguido la estructura deseada

► datos_saltadores_para_tabla

Para mostrar los datos en el DataFrame necesitamos reorganizarlos de manera que así se nos sea más cómodo, para automatizarlo creamos un método con la siguiente estructura:

```
#Organizar nombres, apellidos y rankings para tabla
def datos_saltadores_para_tabla(lista_saltadores): # ← SALTADORES
    competi_nombre = []; competi_apellido = []; competi_ranking = []
    for i in lista_saltadores:
        competi_nombre.append(i[0])
        competi_apellido.append(i[1])
        competi_ranking.append(i[3])
    datos = [competi_nombre, competi_apellido, competi_ranking]
    return datos
```

Le pasaremos como parámetro la lista de los saltadores y crearemos tres listas, correspondiente a los tres datos personales que nos hace falta de los saltadores (*nombre, apellido, ranking*)

Hacemos un bucle que recorra la lista pasada como parámetro y, conociendo la estructura que tiene nuestra lista, añadimos los datos a su lista correspondiente usando los índices

Finalmente añadimos las tres listas a una para devolverlos

► total_por_saltador

En este método calcularemos el total de las notas del número de saltos que le pasemos de cada jugador, ajustando dichas notas con su grado de dificultad

Le pasaremos como parámetro la lista de los saltos de cada jugador

```
def total_por_saltador(lista_saltos):
    total_por_saltador = [
        #[TOTAL, SALTADOR],
        #[TOTAL, SALTADOR], ...
    ]; num_saltador=1
    for saltador in lista_saltos: #[SALTO 1, SALTO 2,
        total_y_saltador = [] #TOTAL, SALTADOR
        total = 0; grado = 0
        for salto in saltador: #SALTO X
            total += salto * grado_dificultad[grado]
            grado+=1
        total_y_saltador.append(int(total*10)/10)
        total_y_saltador.append(num_saltador) # = [TOT
        num_saltador+=1
        total_por_saltador.append(total_y_saltador)
    return total_por_saltador
```

Comenzamos generando una lista donde almacenaremos el total y el número de saltador, esto nos ayudará más tarde cuando ordenemos las notas. Además establecemos la variable que indicará el saltador a 1

Hacemos un bucle que recorra la lista pasada por parámetro

Creamos la lista donde guardaremos el total y el número del saltador, `total_y_saltador`, también creamos dos variables inicializadas a 0, `total`, donde se guardará la suma de las notas y `grado`, que iterará el número de salto que estamos tratando para sacar su grado de dificultad correspondiente

Iniciamos otro bucle que recorrerá la lista iterada en el bucle anterior

Por cada salto en la lista que estamos recorriendo sumamos al total el valor del salto multiplicado por su grado de dificultad, tomándolo de la lista que creamos al principio y usando la variable “grado” como índice. Aumentamos “grado” para el próximo salto

Una vez finalizado este primer bucle se añadirá el valor a la lista `total_y_saltador`, además del número de saltador, que después aumentará en 1. Una vez rellena la lista se añade a la lista donde almacenaremos todas las listas, `total_por_saltador`

Una vez finalice el bucle y se añadan todas las listas, la devolveremos

► `datos_salts_para_tabla`

Igual que en el caso anterior, necesitamos reordenar los datos para poder mostrarlos directamente en nuestro DataFrame

```
#Organizar saltos para tabla # ↓ SALTOS POR SALTADOR
def datos_salts_para_tabla(lista_salts , n_salts):
    todos_salto_1 = []; todos_salto_2 = []; todos_salto_3 = []; datos=[]
    if n_salts==2:
        datos = [todos_salto_1, todos_salto_2]
    elif n_salts==3:
        datos = [todos_salto_1, todos_salto_2, todos_salto_3]

    for n_jug in lista_salts:
        salto_x = 0
        for salto in datos:
            salto.append(n_jug[salto_x])
            salto_x +=1
    return datos
```

Le pasamos como parámetros la lista del total de cada salto por cada saltador (*lista que nos devuelve el método `total_por_saltador`*) y el número de saltos que queremos recibir

Empezamos creando las tres listas para los tres saltos posibles y una lista “datos” que será donde almacenaremos las listas de los saltos

Este método lo usaremos cuando queramos 3 o 2 saltos, así que preguntaremos si el parámetro correspondiente al número de saltos es 2 o 3, dependiendo de la respuesta añadiremos a “datos” las listas correspondientes

Una vez nuestra lista “datos” lista comenzamos un bucle que itera el parámetro “lista_salto” e iniciamos una variable con la que llevaremos el conteo de en qué salto vamos

Dentro, iniciamos otro bucle que recorrerá la lista “datos” que hemos preparado anteriormente y añadiremos a la lista iterada el valor total del saltador que estemos iterando en ese momento, aumentamos el conteo del salto

Finalmente devolvemos la lista datos, que tendrá 3 o 2 listas con todos los totales del salto X de cada saltador

► sacar_totales_de_tps

Si bien dijimos que al crear los totales necesitábamos guardarnos el número de saltador con él, para presentarlo en el DataFrame necesitamos sólo los totales

```
#Solo la puntuación
def sacar_totales_de_tps(total_por_saltador):
    totales = []
    for t in total_por_saltador:
        totales.append(t[0])
    return totales
```

En esta función simplemente le pasamos como parámetro la lista que obtenemos del método `total_por_saltador` y creamos una nueva lista donde guardaremos el valor correspondiente al total de cada lista, siendo este el primero

► ordenar_puesto

Finalmente, esta función crea una lista con el número de cada saltador ordenado según sus notas totales

Recibe como parámetro la lista que obtenemos del método `total_por_saltador` y la ordenamos

Aunque la lista este compuesta de listas, el método `sorted` funciona de manera que ordena según el primer elemento de dicha lista, en nuestro caso serían los totales, por lo cual nos viene perfecto, añadimos el parámetro `reverse=True` para que ordene de mayor a menor


```
#Ordenar por total
def ordenar_puesto(total_por_saltador):
    total_ordenado = sorted(total_por_saltador, reverse=True)
    ranking = []
    for i in total_ordenado:
        ranking.append(123)
    for i in range(0, len(ranking)):
        ranking[total_ordenado[i][1]-1] = i+1
    return ranking
```

Una vez nuestra lista ordenada, creamos una nueva lista donde guardaremos el resultado final

Iniciamos un bucle que recorra la lista ordenada, para tener la misma longitud, y rellenamos la nueva lista con valores provisionales, esto es para tener la longitud y valores definidos

Después iniciamos otro bucle que recorra la lista `ranking` y hacemos que el valor correspondiente al número del saltador del saltador iterado en ese momento -1 sea la posición de la lista `ranking` donde colocaremos el puesto correspondiente, aprovechamos la variable iteradora “i” y le sumamos 1, para que no empiece en 0.

[Representación ↓](#)

total_ordenado = [[13, 5], [11, 2], [10, 3], [9, 1], [5, 4]]
posicion = 1º
nota ↑ saltador

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

Con esta lista, cuando ordenemos el DataFrame no se nos descuadrarán los datos

Estas son todas las funciones que usaremos en nuestro código, comenzaremos a preparar los datos para mostrarlos en los dataframes

Preparación de datos

Comenzamos ejecutando nuestras funciones para sacar los datos:

```
total_salto_saltadores = calcular_todos_salto(saltadores, 3) #saltos_por_saltador / #Lista_salto
ronda2_salto_saltadores = calcular_todos_salto(saltadores, 2) #saltos_por_saltador r2
```

Estas dos variables serán listas que almacenarán el número de saltos correspondiente (3 y 2) de cada saltador, la primera la usaremos para la actividad general y la segunda para el ejercicio 3

```
datos_saltadores = datos_saltadores_para_tabla(saltadores) #n, a, r
datos_salto = datos_salto_para_tabla(total_salto_saltadores, 3) #s1, s2, s3
```

Preparamos los datos de los saltadores y de los saltos para mostrarlos en el DataFrame

```
total_y_saltador = total_por_saltador(total_salto_saltadores) #[total, saltador], ...
total_y_saltador_r2 = total_por_saltador(ronda2_salto_saltadores) # ↑ RONDA 2 (EJ 3)
```

Finalmente sacamos los totales junto al número del saltador, tanto para el ejercicio en general como en el ejercicio 3

Creamos las listas que usaremos en el diccionario para crear el DataFrame usando los resultados obtenidos anteriormente, para los datos de los saltadores y los saltos sacamos las listas de su lista correspondiente y para el total y los puestos usamos sus funciones. Además prepararemos los datos para el ejercicio 3

```
# ----- Nombre
competi_nombre = datos_saltadores[0]
# ----- Apellido
competi_apellido = datos_saltadores[1]
# ----- Ranking
competi_ranking = datos_saltadores[2]
# ----- Salto 1
todos_salto_1 = datos_salto[0]
# ----- Salto 2
todos_salto_2 = datos_salto[1]
# ----- Salto 3
todos_salto_3 = datos_salto[2]
# ----- Total
totales_acti1 = sacar_totales_de_tps(total_y_saltador)
# ----- Puesto
puesto_acti1 = ordenar_puesto(total_y_saltador)
```

```
# ----- Total
totales_ronda2 = sacar_totales_de_tps(total_y_saltador_r2)
# ----- Puesto
puesto_ronda2 = ordenar_puesto(total_y_saltador_r2)
```

Actividades

Actividad 1

Creamos el diccionario

```
competicion_act1 = {  
    'Nombre': competi_nombre,  
    'Apellido': competi_apellido,  
    'Ranking': competi_ranking,  
    'Salto 1': todos_salto_1,  
    'Salto 2': todos_salto_2,  
    'Salto 3': todos_salto_3,  
    'Total': totales_acti1,  
    'Puesto': puesto_acti1,  
}
```

Creamos el DataFrame con dicho diccionario y lo ordenamos con `sort_values()` según la columna 'Puesto'. La mostramos

```
tabla = pd.DataFrame(competicion_act1)  
tabla_ordenada = tabla.sort_values(by=['Puesto'], ascending=True)  
print("-----")  
print(tabla_ordenada)  
print("-----\n")
```

Hemos obtenido este resultado

	Nombre	Apellido	Ranking	Salto 1	Salto 2	Salto 3	Total	Puesto
6	Impac	Tum	2180	17.0	22.0	5.5	17.6	1
0	Raúl	Capablanca	2780	22.0	20.0	15.0	16.0	2
1	José	Martínez	2638	6.0	19.5	10.5	15.6	3
5	Skibidi	Sigma	2679	16.5	15.5	8.5	12.4	4
4	Ubuntu	Proxmox	2080	11.5	13.5	12.0	10.8	5
2	Iván	Gómez	2809	18.5	11.0	24.0	8.8	6
3	Pepe	Pepito	2323	9.5	8.5	12.5	6.8	7

Actividad 2

Para esta actividad no necesitamos crear un DataFrame, sólo sacar los datos:

```
for i in range(1,4):  
    salto = 'Salto ' + str(i)  
    tabla_salto = tabla.sort_values(by=[salto], ascending=False)  
    saltador = tabla_salto[tabla_salto[salto] == tabla_salto[salto].max()]  
    datos_salt = saltador.iloc[0]  
    print("El saltador " + str(datos_salt['Nombre']) + " " + str(datos_salt['Apellido']) + " hizo el mejor salto " + str(i) + "  
    obteniendo: " + str(datos_salt[salto]) + " puntos")
```

Creamos un bucle que iterará 3 veces, comenzando por 1

Usamos una variable (*salto*) que se actualizará en cada bucle que usaremos como el nombre de la columna de la que vamos a sacar los datos

Usamos el DataFrame del ejercicio anterior y lo ordenamos por dicha columna

Para obtener el saltador sacamos el índice de la tabla ordenada donde el valor del salto iterado sea el máximo. Para sacar sus datos usamos `.iloc[0]`, devolviendo así una lista con los valores de la primera lista

Finalmente mostramos por pantalla los datos sacándolos de la lista recién creada

Obtenemos este resultado:

```
El saltador Raúl Capablanca hizo el mejor salto 1 obteniendo: 22.0 puntos
El saltador Impac Tum hizo el mejor salto 2 obteniendo: 22.0 puntos
El saltador Iván Gómez hizo el mejor salto 3 obteniendo: 24.0 puntos
```

Actividad 3

Al igual que en la actividad 1, ya tenemos los datos listos, solo necesitamos crear el diccionario

```
## ACTIVIDAD 3
ronda_2 = {
    'Nombre': competi_nombre,
    'Apellido': competi_apellido,
    'Salto 1': todos_salto_1,
    'Salto 2': todos_salto_2,
    'Total': totales_ronda2,
    'Puesto': puesto_ronda2,
}
tabla_ranking_prov = pd.DataFrame(ronda_2)
tabla_ranking_prov_ordenada = tabla_ranking_prov.sort_values(by=['Puesto'], ascending=True)
print("\n-----RANKING PROVISIONAL PARA LA SEGUNDA RONDA-----")
print(tabla_ranking_prov_ordenada)
print("-----")
```

Repetimos el proceso, sin añadir la columna “Salto 3” y “Ranking”, usando los datos correspondientes que creamos anteriormente para la actividad 3

```
-----RANKING PROVISIONAL PARA LA SEGUNDA RONDA-----
   Nombre  Apellido  Salto 1  Salto 2  Total  Puesto
5  Skibidi    Sigma    16.5    15.5    12.8      1
0   Raúl  Capablanca    22.0    20.0     9.2      2
6   Impac      Tum    17.0    22.0     8.0      3
4  Ubuntu  Proxmox    11.5    13.5     8.0      4
2   Iván    Gómez    18.5    11.0     7.2      5
1   José  Martínez     6.0    19.5     6.8      6
3   Pepe    Pepito     9.5     8.5     3.6      7
-----
```

Actividad 4

Para generar el top 5 tenemos que sacar del DataFrame los datos de los 5 primeros, como el DataFrame ya está ordenada no necesitamos cambiarla

```
## ACTIVIDAD 4
top_5_nombre = []; top_5_apellido = []; top_5_ranking = []; top_5_fede = []
for i in range(1,6):
    salt_puesto = tabla_ordenada[tabla_ordenada['Puesto'] == i]
    datos_salt = salt_puesto.iloc[0]
    top_5_nombre.append(str(datos_salt['Nombre']))
    top_5_apellido.append(str(datos_salt['Apellido']))
    top_5_ranking.append(int(datos_salt['Ranking']))
    for saltador in saltadores:
        if saltador[0] == top_5_nombre[i-1]:
            top_5_fede.append(saltador[2])
```

Preparamos las listas donde guardaremos los datos e iniciamos un bucle que iterará 5 veces empezando por 1

Sacamos el saltador comparando su puesto al iterador (*1-5*) y sacamos la fila con `iloc`, después guardamos los datos en la lista correspondiente

Para la federación, como no se incluyó en el DataFrame, la sacaremos comparando el nombre del saltador que estamos iterando (*i-1 para obtener el índice correctamente, ya que empezamos en 1*) con los nombres que tenemos en la lista de saltadores, si coincide, añadimos a la lista donde almacenamos las federaciones la federación del saltador correspondiente (*saltador[2]*)

Finalmente preparamos el diccionario, creamos el DataFrame y lo mostramos

```
top_5 = {
    'Puesto': [1,2,3,4,5],
    'Nombre': top_5_nombre,
    'Apellido': top_5_apellido,
    'Ranking': top_5_ranking,
    'Federación': top_5_fede,
}
tabla_top_5 = pd.DataFrame(top_5)
print("-----TOP 5-----")
print(tabla_top_5)
print("-----")
```


Obtenemos este resultado:

```
-----TOP 5-----
```

	Puesto	Nombre	Apellido	Ranking	Federación
0	1	Impac	Tum	2180	Federación PC
1	2	Raúl	Capablanca	2780	Federación Cubana
2	3	José	Martínez	2638	Federación Bética
3	4	Skibidi	Sigma	2679	Federación Rizzler
4	5	Ubuntu	Proxmox	2080	Federación Torretas

```
-----
```

Actividad 5

Eliminamos los datos de Raúl creando una copia de la lista que almacena los datos personales de todos los saltadores y los datos de los saltos de todos los saltadores, a ambas copias le eliminamos la fila de índice 0, ya que esta es la que corresponde a Raúl

```
#Eliminar Raúl
saltadores_act = saltadores.copy()
del saltadores_act[0] #eliminar raul

total_saltos_saltadores_act = total_saltos_saltadores.copy()
del total_saltos_saltadores_act[0] #eliminar raul
```

Recalculamos los datos necesarios para la creación del diccionario igual que hicimos al principio

```
#Nuevos calculos
datos_saltadores_act = datos_saltadores_para_tabla(saltadores_act) #n, a, r
datos_saltos_act = datos_saltos_para_tabla(total_saltos_saltadores_act, 3) #s1, s2, s3
total_y_saltador_act = total_por_saltador(total_saltos_saltadores_act) #[total, saltador],
```

Preparamos los datos para el DataFrame y creamos el diccionario

```
# ----- Nombre
competi_nombre_act = datos_saltadores_act[0]
# ----- Apellido
competi_apellido_act = datos_saltadores_act[1]
# ----- Ranking
competi_ranking_act = datos_saltadores_act[2]
# ----- Salto 1
todos_salto_1_act = datos_saltos_act[0]
# ----- Salto 2
todos_salto_2_act = datos_saltos_act[1]
# ----- Salto 3
todos_salto_3_act = datos_saltos_act[2]
# ----- Total
totales_act = sacar_totales_de_tps(total_y_saltador_act)
# ----- Puesto
puesto_act = ordenar_puesto(total_y_saltador_act)
```

```
datos_tabla_sin_raul = {
    'Nombre': competi_nombre_act,
    'Apellido': competi_apellido_act,
    'Ranking': competi_ranking_act,
    'Salto 1': todos_salto_1_act,
    'Salto 2': todos_salto_2_act,
    'Salto 3': todos_salto_3_act,
    'Total': totales_act,
    'Puesto': puesto_act,
}
```

Creamos el DataFrame, ordenamos y lo mostramos

```
tabla_sin_raul = pd.DataFrame(datos_tabla_sin_raul)
tabla_sin_raul_ordenada = tabla_sin_raul.sort_values(by=['Puesto'], ascending=True)
print("-----TABLA FINAL ACTUALIZADA-----")
print(tabla_sin_raul_ordenada)
print("-----")
```

Obtenemos este resultado

```
!!!!HA OCURRIDO UN IMPREVISTO!!!!
Raúl Capablanca no pasa el control anti-doping, consumió por error una sustancia prohibida y resulta descalificado
Presentamos el ranking actualizado
-----TABLA FINAL ACTUALIZADA-----
   Nombre Apellido  Ranking  Salto 1  Salto 2  Salto 3  Total  Puesto
5   Impac      Tum    2180    17.0    22.0    5.5    17.6      1
0   José    Martínez  2638     6.0    19.5    10.5    15.6      2
4  Skibidi    Sigma  2679    16.5    15.5     8.5    12.4      3
3  Ubuntu  Proxmox  2080    11.5    13.5    12.0    10.8      4
1    Iván    Gómez   2809    18.5    11.0    24.0     8.8      5
2    Pepe    Pepito  2323     9.5     8.5    12.5     6.8      6
-----
```

Finalmente nuestros ejercicios están finalizados de manera dinámica con datos generados automáticamente en cada ejecución