Fundamentos de la programación

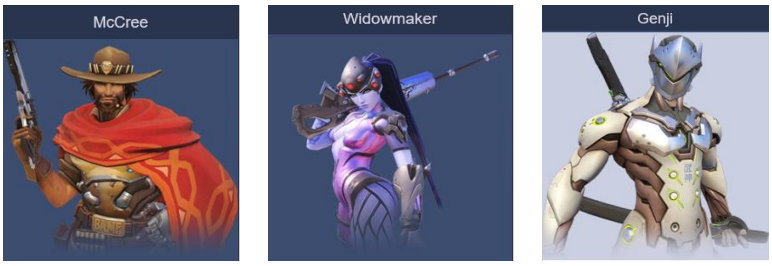
EJERCICIO RESUELTO TIPO PECL2: ESTADISTICAS OVERWATCH

Overwatch es un videojuego multijugador de disparos en primera persona (*shooter*) que se desarrolla en un escenario inspirado en el mundo real donde se enfrentan dos equipos de seis jugadores cada uno. Cada miembro de un equipo escoge uno de los varios héroes disponibles, los cuales tienen movimientos y habilidades únicas y pertenecen a una de las tres clases posibles: *Damage*, *Tank* y *Support*. A medida que juegan partidas, los jugadores progresan en el juego, por ejemplo, subiendo de nivel, recibiendo elogios de otros jugadores (en Overwatch es posible recompensar el buen comportamiento de tus compañeros de equipo) e incrementando los contadores de daños infligidos, asistencias, elementos destruidos, etc. Se desea implementar una aplicación de estadísticas de Overwatch donde se pueda consultar información actualizadas sobre los jugadores, para lo cual se debe guardar la siguiente información:

a) Información general del jugador: nombre, nivel (un entero), índice de habilidad (un entero), nivel de elogio (un entero de 0 a 5) y si su perfil es público o no.
b) Logros: partidas ganadas, partidas perdidas, tiempo total jugado (en segundos), elementos destruidos, número de asistencias y daño total infligido (un entero)
c) Héroes utilizados: una lista con los diferentes héroes con los que ha jugado (aquellos héroes que no ha usado todavía el jugador no están en su lista), que incluye la siguiente información asociada para cada héroe:
    a. Tiempo jugado por el jugador con ese héroe (en minutos)
    b. Partidas ganadas con ese héroe (un valor entero)
    c. Porcentaje de partidas ganadas
    d. Precisión con armas (en %)
    e. Promedio de eliminaciones por vida (un valor real)
    f. Precisión de golpes críticos (en %)

Se desea implementar esta aplicación de estadísticas en una versión simplificada para hacer una demo donde sólo hay información de 5 jugadores (`angiexx`, `ujfaluxi`, `r@drunner`, `filaraki` y `fcwolf`) y sólo existen 9 héroes:

- `McCree`, `WidowMaker` y `Genji` de la categoría *Damage*
- `RoadHog`, `Orisa` y `Sigma` de la categoría *Tank*
- `Ana`, `Baptiste` y `Zenyatta` de la categoría *Support*



para lo cual necesitamos tu ayuda en los siguientes puntos:

1. Diseña las estructuras de datos que soporten la información descrita en los apartados anteriores y pon un ejemplo de inicialización. No es necesario meter todos los datos de todos los jugadores ni de todos los héroes utilizados por un jugador, sólo un ejemplo donde se vea claramente el enfoque.

```
lista_jugadores = [ 'angiexx', 'ujfaluxi', 'r@drunner', 'filaraki', 'fcwolf' ]
lista_heroes = [
    'McCree', 'WidowMaker', 'Genji',
    'RoadHog', 'Orisa', 'Sigma',
    'Ana', 'Baptiste', 'Zenyatta'
]
detalle_heroe = {
            'McCree': 'Damage',
            'WidowMaker': 'Damage',
            'Genji': 'Damage',
            'RoadHog': 'Tank',
            'Orisa': 'Tank',
            'Sigma': 'Tank',
            'Ana': 'Support',
            'Baptiste': 'Support',
            'Zenyatta': 'Support'
            }

logros_jugador = {
            'angiexx': {
                    'partidas_ganadas': 100,
                    'partidas_perdidas': 24,
                    'nivel': 25,
                    'habilidad': 30,
                    'elogio': 2,
                    'perfil_publico': True
            },
            'ujfaluxi': {
                    'partidas_ganadas': 160,
                    'partidas_perdidas': 2,
                    'nivel': 47,
                    'habilidad': 38,
                    'elogio': 3,
                    'perfil_publico': True
            },
            'r@drunner': {
                    'partidas_ganadas': 60,
                    'partidas_perdidas': 23,
                    'perfil_publico': True
                    # etc.
            },
            'filaraki': {
                    'partidas_ganadas': 86,
                    'partidas_perdidas': 19,
                    'perfil_publico': False
                    #etc.
            }
            ,
            'fcwolf': {
                    'partidas_ganadas': 4,
                    'partidas_perdidas': 1,
                    'perfil_publico': True
                    #etc.
            }
}

heroes_jugador = {
            'filaraki': {
```

```
                'McCree': {
                        'tiempo_jugado': 2350,
                        'partidas_ganadas': 4,
                        'porcentaje_victorias': 0.35,
                        'precision_armas': 0.52,
                        'promedio_eliminaciones_por_vida': 4.57,
                        'precision_golpes': 0.21,
                        'asesinatos': 345
                },
                'WidowMaker': {
                        'tiempo_jugado': 3060,
                        'partidas_ganadas': 3,
                        'porcentaje_victorias': 0.15,
                        'precision_armas': 0.12,
                        'promedio_eliminaciones_por_vida': 1.35,
                        'precision_golpes': 0.11,
                        'asesinatos': 43
                },
                'RoadHog': {
                    'tiempo_jugado': 2100,
                    'partidas_ganadas': 9,
                    'precision_armas': 0.22,
                    'promedio_eliminaciones_por_vida': 1.85,
                    'precision_golpes': 0.54,
                    # rest of the data for this hero here
                },
                'Ana': {
                    'tiempo_jugado': 231,
                    'partidas_ganadas': 12,
                    'precision_armas': 0.2,
                    'promedio_eliminaciones_por_vida': 5.98,
                    'precision_golpes': 0.61,
                    # rest of the data for this hero here
                },
                'Baptiste': {
                    'tiempo_jugado': 3990,
                    'partidas_ganadas': 4,
                    'precision_armas': 0.33,
                    'promedio_eliminaciones_por_vida': 15.35,
                    'precision_golpes': 0.72,
                    # rest of the data for this hero here
                }
        },
        'ujfaluxi': {
                'WidowMaker': {
                        'tiempo_jugado': 1490,
                        'partidas_ganadas': 11,
                        'porcentaje_victorias': 0.48,
                        'precision_armas': 0.23,
                        'promedio_eliminaciones_por_vida': 9.32,
                        'precision_golpes': 0.88,
                        'asesinatos': 427
                    },
                'Orisa': {
                        'tiempo_jugado': 765,
                        'precision_armas': 0.19,
                        'promedio_eliminaciones_por_vida': 1.11,
                        'precision_golpes': 0.12,
                        # rest of data for this hero here
                    },
                'Sigma': {
                        'tiempo_jugado': 1328,
                        'precision_armas': 0.4,
                        'promedio_eliminaciones_por_vida': 7.13,
                        'precision_golpes': 0.52,
                        # rest of data for this hero here
```

```
                        },
                        'Zenyatta': {
                                'tiempo_jugado': 348,
                                'precision_armas': 0.19,
                                'promedio_eliminaciones_por_vida': 13.75,
                                'precision_golpes': 0.37,
                                # rest of data for this hero here
                        },
                        'Baptiste': {
                                'tiempo_jugado': 1855,
                                'precision_armas': 0.1,
                                'promedio_eliminaciones_por_vida': 7.37,
                                'precision_golpes': 0.28,
                                # rest of data for this hero here
                        }
                },
                'angiexx': {
                        'Genji': {
                                'tiempo_jugado': 1150,
                                'partidas_ganadas': 43,
                                'porcentaje_victorias': 0.85,
                                'precision_armas': 0.73,
                                'promedio_eliminaciones_por_vida': 7.72,
                                'precision_golpes': 0.68,
                                'asesinatos': 775
                            },
                        'Orisa': {
                                'tiempo_jugado':2370,
                                'precision_armas': 0.61,
                                'promedio_eliminaciones_por_vida': 11.37,
                                'precision_golpes': 0.34,
                                # rest of data for this hero here
                            },
                        'Sigma': {
                                'tiempo_jugado': 1610,
                                'precision_armas': 0.41,
                                'promedio_eliminaciones_por_vida': 5.97,
                                'precision_golpes': 0.32,
                                # rest of data for this hero here
                        },
                        'Ana': {
                                'tiempo_jugado': 1108,
                                'precision_armas': 0.1,
                                'promedio_eliminaciones_por_vida': 7.37,
                                'precision_golpes': 0.28,
                                # rest of data for this hero here
                        },
                        'Baptiste': {
                                'tiempo_jugado': 3419,
                                'precision_armas': 0.51,
                                'promedio_eliminaciones_por_vida': 4.17,
                                'precision_golpes': 0.44,
                                # rest of data for this hero here
                        }
                }
        # etc. (lo mismo para el resto de los jugadores)
    }
}
```

2. Implementa un procedimiento para introducir por teclado información completa de un nuevo jugador que acaba de registrarse y del que, por tanto, no se tienen datos de juego. Asegúrate de que no existe ya otro con el mismo nombre.

```
def nuevo_jugador(lista_jugadores, logros_jugador, heroes_jugador):
    """ list, dict, dict -> None
        OBJ: Permite introducir información de un nuevo jugador
    """
    nick = input('Introduce el nombre del jugador: ')
    while nick in lista_jugadores:
        nick = input('Error, el jugador ya existe. Introduce otro nombre: ')
    lista_jugadores.append(nick)
    publico = input('¿Será un perfil público o no? (S/N): ') in 'Ss'
    logros_jugador[nick] = {
                    'nivel': 0,
                    'habilidad': 0,
                    'elogio': 0,
                    'perfil_publico': publico,
                    'partidas_ganadas': 0,
                    'partidas_perdidas': 0,
    }
    heroes_jugador[nick] = {}

#prueba
nuevo_jugador(lista_jugadores, logros_jugador, heroes_jugador)
print(logros_jugador)
print(heroes_jugador)
```

3. Para un jugador de tu elección (por ejemplo, `filaraki`) extrae de los datos almacenados una lista de tuplas con la información sobre los héroes que ha utilizado, el tiempo jugado con cada uno y el número de partidas ganadas con cada uno. El resultado para `filaraki` sería, por ejemplo, así:

```
[ ('McCree', 2350, 4), ('WidowMaker', 3060, 3),
  ('RoadHog', 2100, 9), ('Ana', 231, 12),
  ('Baptiste', 3990, 4)
]
```

```
def extraer_lista_con_datos_de_heroes(heroes_jugador, jugador):
    """ dict, str -> list
        OBJ: extrae para un jugador una lista de tuplas con la información
             sobre los héroes que ha utilizado, el tiempo jugado con cada uno
             y el número de partidas ganadas.
    """
    lista = []
    for heroe in heroes_jugador[jugador]:
        lista.append( (
                        heroe,
                        heroes_jugador[jugador][heroe]['tiempo_jugado'],
                        heroes_jugador[jugador][heroe]['partidas_ganadas']
                      )
        )
    return lista

# prueba
lista_filaraki = extraer_lista_con_datos_de_heroes(heroes_jugador, 'filaraki')
print(lista_filaraki)
```

4. Adapta uno de los métodos de ordenación vistos en la asignatura (burbuja, selección o inserción) para ordenar la lista de tuplas generada en el apartado anterior en orden decreciente de tiempo jugado y a igualdad de tiempo, por número de partidas ganadas.

```
def mayor_que(t1, t2):
    """ tuple, tuple -> bool
        OBJ: determina si una tupla con información de héroes es menor que otra
        Ej: Las tuplas tienen el formato (nombre_heroe, t_jugado, part_ganadas)
    """
    return (t1[1] > t2[1]) or ((t1[1] == t2[1]) and (t1[2] > t2[2]))

def insercion(mis_heroes):
    """ list -> list
        OBJ: Ordena mediante el método de inserción una lista de tuplas de héroes
    """
    for i in range (1, len (mis_heroes)):
        clave = mis_heroes[i]
        j = i -1
        while j >=0 and mayor_que(clave,mis_heroes[j]):
            mis_heroes[j+1] = mis_heroes[j]
            j -= 1
        mis_heroes[j+1] = clave
    return mis_heroes

# prueba
lista_test = [
    ('McCree', 2350, 4), ('WidowMaker', 3060, 3),
    ('RoadHog', 2100, 9), ('Ana', 231, 12),
    ('Baptiste', 231, 4)
]
insercion(lista_test)
print(lista_test)
```

5. Implementa una función que, a partir del nombre de jugador y del tipo de juego elegido para la siguiente partida (daño, apoyo o tanque), retorne el nombre del héroe que con más probabilidades va a elegir este jugador utilizando para ello una función de predicción que debería tener en cuenta al menos 3 criterios de los detallados en el apartado (c). Trata de ser creativo cuando diseñes la función de predicción.

```
def heroe_mas_probable(jugador, tipo_juego, heroes_jugador, detalle_heroe):
    """ str, str, dict, dict -> str
        OBJ: predice en función de datos anteriores el heroe con más probablidad
             de ser elegido por el jugador para el tipo_juego elegido
    """
    probabilidad_heroes = {}
    # calculamos el tiempo total para luego ver el tiempo relativo de cada heroe
    tiempo_total_jugado = 0
    for heroe in heroes_jugador[jugador]:
        tiempo_total_jugado += heroes_jugador[jugador][heroe]['tiempo_jugado']
    probabilidad_heroes = {}
    for heroe in heroes_jugador[jugador]:
        if detalle_heroe[heroe] == tipo_juego: # solo calcula prob. para este tipo
          # la prob. p se calcula en función del % de tiempo jugado relativo y
          # el % de victorias. A igualdad, más probabilidad el que tiene mejor
          # promedio de eliminaciones por vida (p2)
          p = heroes_jugador[jugador][heroe]['tiempo_jugado'] \
                 / tiempo_total_jugado \
                 + heroes_jugador[jugador][heroe]['porcentaje_victorias']
          p2 = heroes_jugador[jugador][heroe]['promedio_eliminaciones_por_vida']
          probabilidad_heroes[heroe] = { 'p':p, 'p2':p2 }
    heroe_mas_probable = ''
    p_max = p2_max = 0
    for heroe in probabilidad_heroes:
        if probabilidad_heroes[heroe]['p'] > p_max \
           or (probabilidad_heroes[heroe]['p'] == p_max \
               and probabilidad_heroes[heroe]['p2'] > p2_max):
              p_max = probabilidad_heroes[heroe]['p']
              p2_max = probabilidad_heroes[heroe]['p2']
              heroe_mas_probable = heroe
    return heroe_mas_probable

# prueba
print(heroe_mas_probable('filaraki','Damage',heroes_jugador,detalle_heroe))
```

6. Implementa un procedimiento que a partir de un nombre de jugador y un criterio (que puede ser tiempo jugado, partidas ganadas o precisión con armas) muestre en pantalla las estadísticas completas del jugador, que incluirán la información general del jugador, sus logros y toda la información del criterio elegido sobre los héroes utilizados por el jugador.

```
import pprint

def estadisticas_jugador_por_criterio(jugador, criterio, lista_jugadores,
                                      heroes_jugador, logros_jugador):
    """ str, str, dict, dict, dict -> str
        OBJ: A partir de un nombre de jugador y un criterio (tiempo jugado,
             partidas ganadas o precisión con armas muestra estadísticas completas
             del jugador y sus héroes utilizados, centrándose en el criterio.
    """
    if jugador not in lista_jugadores:
        print('El jugador indicado no existe. No hay estadísticas para el mismo')
    else:
        print(f'Estadisticas de jugador: {jugador}')
        print('Detalle del jugador...')
        pprint.pprint(logros_jugador[jugador]) # pone “bonitos” los datos del dict
        print(f'Datos de {criterio} para el jugador {jugador}')
        for heroe in heroes_jugador[jugador]:
            print(f'{heroe}: {heroes_jugador[jugador][heroe][criterio]}')

#prueba
#estadisticas_jugador_por_criterio('angiexx',
                                   'precision_armas',
                                   lista_jugadores,
                                   heroes_jugador,
                                   logros_jugador )
```

7. Implementa una función que retorne los tres héroes (uno por cada categoría) más precisos a nivel global teniendo en cuenta tanto su precisión con armas como su precisión de golpes críticos. Esta función sólo deberá considerar los datos de aquellos jugadores con perfil público.

```
def extraer_mas_preciso_categoria(dict_heroes, categoria):
    """ dict, dict -> str
        OBJ: Retorna el héroe más preciso de una determinada categoría.
             dict_heroes es un diccionario de heroes:precision, así:
             {'Ana': 0.1, 'Genji': 0.4 ...}
    """
    max_precision = 0
    heroe_mas_preciso = None
    for heroe in dict_heroes:
        if detalle_heroe[heroe] == categoria and
           dict_heroes[heroe] > max_precision:
              max_precision = dict_heroes[heroe]
              heroe_mas_preciso = heroe
    return heroe_mas_preciso

def heroes_mas_precisos(heroes_jugador, logros_jugador, detalle_heroe):
    """ dict, dict, dict -> list
        OBJ: Retorna los tres héroes (1 por categoría) más precisos a nivel global
             teniendo en cuenta tanto su precisión con armas como su precisión de
             golpes críticos. Sólo considera jugadores con perfil público.
    """
    precision_heroes = {}
    for jugador in heroes_jugador:
        if logros_jugador[jugador]['perfil_publico']:
            for heroe in heroes_jugador[jugador]:
               if heroe not in precision_heroes: precision_heroes[heroe] = 0
               precision_heroes[heroe] =
                         heroes_jugador[jugador][heroe]['precision_golpes'] \
                         + heroes_jugador[jugador][heroe]['precision_armas']
                # corrección para comparar datos de heroes frecuentes
                precision_heroes[heroe] /= 2
    # calculados datos de precisión, extraemos el más preciso por categoría
    lista_heroes_precisos = []
    for categoria in ['Damage', 'Tank', 'Support']:
      lista_heroes_precisos.append(
           (extraer_mas_preciso_categoria(precision_heroes, categoria),categoria)
      )
    return lista_heroes_precisos

#prueba
lista_resultados = heroes_mas_precisos(heroes_jugador,
                                       logros_jugador,
                                       detalle_heroe)
print(lista_resultados)
```

8. Implementa una función que determine el héroe más utilizado (en tiempo jugado) por los 3 jugadores con mayor ratio de partidas ganadas.

```
def criterio_ordenacion(diccionario):
    return diccionario['ratio']

def heroe_mas_utilizado_por_top_players(heroes_jugador, logros_jugador):
    """ dict, dict -> str
        OBJ: héroe más frecuentemente utilizado por los 3 jugadores con > ratio
partidas ganadas
    """
    top_players = []
    for jugador in logros_jugador:
        ganadas = logros_jugador[jugador]['partidas_ganadas']
        perdidas = logros_jugador[jugador]['partidas_perdidas']
        top_players.append({ 'jugador':jugador,
                             'ratio': ganadas / (ganadas + perdidas) })
    # ordenamos por ratio descendentemente
    # se podía haber usado burbuja, inserción o selección
    top_players.sort(key=criterio_ordenacion, reverse=True)
    top3_players = top_players[:3]
    frecuencias = {}
    for top_player in top3_players:
      for heroe in heroes_jugador[top_player['jugador']]:
        if heroe not in frecuencias: frecuencias[heroe] = 0
        frecuencias[heroe] +=
                  heroes_jugador[top_player['jugador']][heroe]['tiempo_jugado']
    tiempo_maximo = max(frecuencias.values())
    posicion_mas_frecuente = list(frecuencias.values()).index(tiempo_maximo)
    return list(frecuencias)[posicion_mas_frecuente]

#prueba
top_heroe = heroe_mas_utilizado_por_top_players(heroes_jugador, logros_jugador)
print(top_heroe)
```

PROPUESTOS

Para terminar, y puesto que nadie ha aprendido nunca a programar viendo programar a otros sin hacer prácticas por sí mismo, os dejamos 3 propuestas sobre la misma estructura de datos para que podáis practicar vuestras habilidades:

9. Implementa una función que determine cuál es el héroe de tipo *Tank* menos frecuentemente utilizado a nivel global.

10. Implementa una función que determine la precisión media con armas de un cierto héroe teniendo en cuenta los datos de todos los jugadores.

11. Implementa una función que determine el número medio de asesinatos por objetivos de los héroes de tipo *Damage* según los datos de aquellos jugadores que tengan un porcentaje de partidas ganadas superior al 40%.