FUNDAMENTO DE PROGRAMACIÓN

PROYECTO: POKÉMON 6VS6

AUTOR: LUIS GIRALDO SANTIAGO *ÚLTIMA MODIFICACIÓN: 28/11/2020*

En este proyecto trabajaremos con los datos correspondientes a los Pokémon utilizados en combates 6 contra 6 en el año 2016. Los datos provienen del juego de Pokémon con los Pokémon utilizados en el competitivo de *singles 6vs6,* donde podemos encontrar datos sobre los tipos de Pokémon, sus estadísticas y sus generaciones. Representaremos la información de entrada mediante listas de tuplas, y a partir de esta estructura implementaremos una serie de funciones que nos permitirán realizar varios tipos de consultas y generar visualizaciones.

Trabajaremos con ficheros en formato CSV. Cada registro del fichero de entrada ocupa una línea y contiene 15 informaciones sobre los nombres (número de la pokedex, nombre, tipo.1, tipo.2, total de puntos, puntos de vida, puntos de ataque, puntos de defensa, puntos de ataque especial, puntos de defensa especial, puntos de velocidad, generación, legendario, maga, tier). Estas son las primeras líneas de un fichero de entrada:

	X.	Name	Type.1	Type.2	Total	HP	Attack	Defense	Sp..Atk	Sp..Def	Speed	Generation	Legendary	Mega	Tier
2	384	Mega Rayquaza	Dragon	Flying	780	105	180	100	180	100	115	3	VERDADERO	VERDADERO	AG
3	94	Mega Gengar	Ghost	Poison	600	60	65	80	170	95	130	1	FALSO	VERDADERO	Uber
4	115	Mega Kangaskhan	Normal		590	105	125	100	60	100	100	1	FALSO	VERDADERO	Uber
5	150	Mewtwo	Psychic		680	106	110	90	154	90	130	1	VERDADERO	FALSO	Uber
6	150	Mega Mewtwo X	Psychic	Fighting	780	106	190	100	154	100	130	1	VERDADERO	VERDADERO	Uber
7	150	Mega Mewtwo Y	Psychic		780	106	150	70	194	120	140	1	VERDADERO	VERDADERO	Uber
8	249	Lugia	Psychic	Flying	680	106	90	130	90	154	110	2	VERDADERO	FALSO	Uber
9	250	Ho-oh	Fire	Flying	680	106	130	90	110	154	90	2	VERDADERO	FALSO	Uber
10	257	Blaziken	Fire	Fighting	530	80	120	70	110	70	80	3	FALSO	FALSO	Uber

Figura 1: fichero de datos

Además de distintos indicadores, generaremos dos gráficas que mostrarán, respectivamente, la frecuencia de uso de los diferentes tiers (Figura 2) o las generaciones más utilizadas (Figura 3).

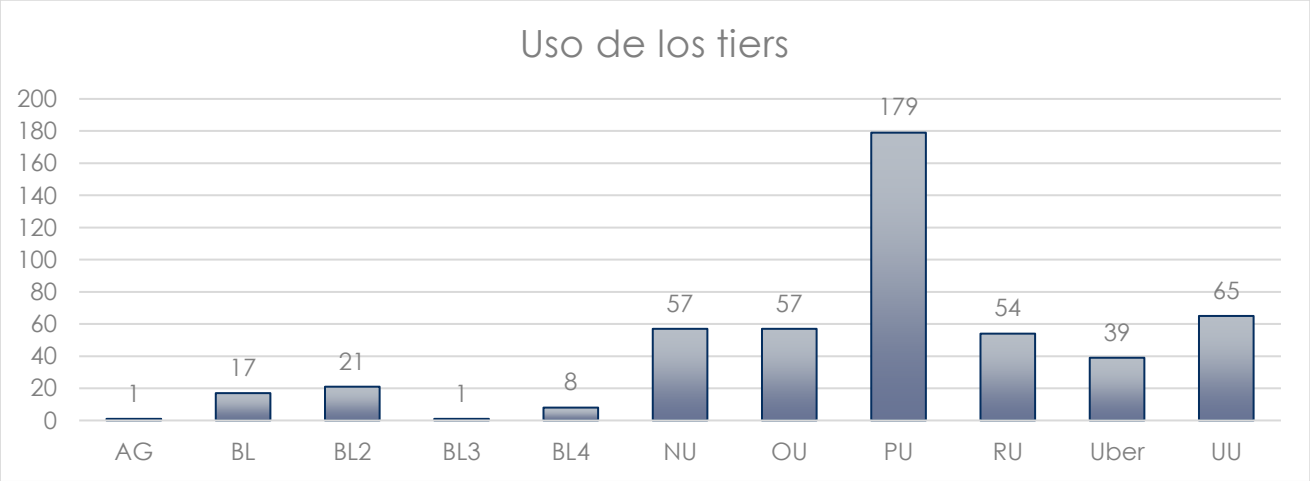

Figura 2

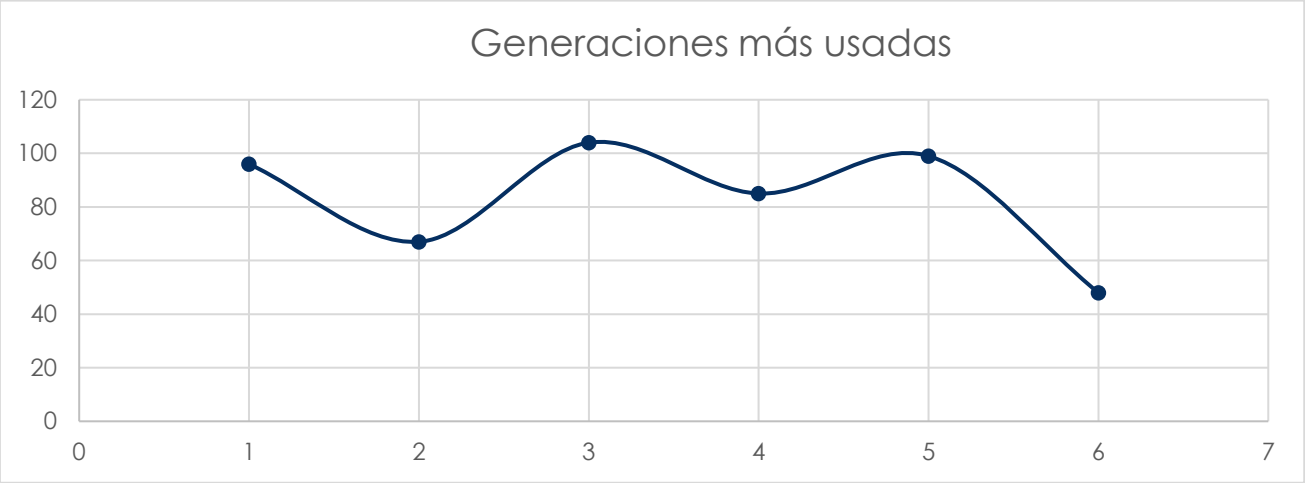

Figura 3

Para almacenar en Python la información de cada una de las líneas se usará la siguiente definición de namedtuple:

RegistroPokemon = namedtuple ('pk', 'n_pokedex, name, type_1, type_2, total, HP, atk, defe, sp_atk, sp_defe, spe, gener, legen, mega,tier')

He creado un fichero Pokemon.py que incluye en él la definición del namedtuple anterior. En este fichero también se incluyen las siguientes funciones:

1. **leer_pokemon:** recibe la ruta de un fichero CSV codificado en UTF-8, y devuelve una lista de tuplas de tipo Pokémon conteniendo todos los datos almacenados en el fichero.
2. **filtrar_por_tipos**: recibe una lista de tuplas de tipo RegistroPokemon y los tipos que por lo cuales hay que filtar en formator (str), y devuelve una lista de tuplas de tipo RegistroPokemon con los pokemon de los tipos recibidos como parámetro. Si no se elige el segundo tipo, se tomará como un valor vacío. (Elección apartado *a* del bloque 2).

3. **obtener_pokemon:** recibe una lista de tuplas de tipo RegistroPokemon, si es legendario o no y si es mega o no en formato (str), y devuelve una lista de tuplas con el nombre, su generación y su tier. (Elección apartado *d* del bloque 2).
4. **media_de_puntos:** recibe una lista de tuplas y un tipo o dos de Pokémon en formato (str) y calcula la media de cada punto de ese tipo o tipos. Si la lista no tiene elementos, devuelve una lista vacía. (Elección apartado *b* del bloque 3).
5. **media_total_de_tiers:** recibe una lista de tuplas y un tier en formato (str) y calcula la media de los puntos totales de ese tier. Si no hay media, devuelve un 0. (Elección apartado *c* del bloque 3).
6. **obtener_pokemon_peor_ataque:** recibe una lista de tuplas y un límite que nos indica el número de Pokémon que se van a mostrar por pantalla, y devuelve los Pokémon con peor ataque ordenados del peor al mejor. (Elección apartado *a* del bloque 4).
7. **obtener_pokemon_total:** recibe una lista de tuplas, un tier (str) y un parámetro n de tipo int, y devuelve una lista de n tuplas (str,int) con el nombre del Pokémon y sus puntos totales, siendo estos los más altos. La lista devuelta estará ordenada de mayor a menor. (Elección apartado *b* del bloque 5).
8. **obtener_diccionario_por_generacion:** recibe una lista de tuplas y la generación en formato (int) y devuelve un diccionario {str: int} en el que las claves son los tipos, solo primario, de los Pokémon y los valores indican cuantos Pokémon hay con esos tipos. (Elección apartado *b* del bloque 6).
9. **Obtener_diccionario_por_tier:** recibe una lista de tuplas, si es legendario o no, y devuelve un diccionario {tier;(tupla)}, donde esa tupla tiene los nombres de los Pokémon y su velocidad y además ordenadas de mayor a menor según su velocidad. (Elección apartado *c* del bloque 6).