Tomas Diaz

tdiaz@udesa.edu.ar

Prof. Alvaro Gaona

Pensamiento Computacional

27 de marzo del 2022

Trabajo Práctico I

Nota: El código fuente no lo puse aca porque se le aplica un estilo que hace que pierda la indentación y no es claro

2. OBJETIVOS DEL TRABAJO

El objetivo principal de este trabajo práctico consiste en el desarrollo de un programa interactivo que tenga la aptitud de ejecutar un marcador de puntaje para dos entradas (dos jugadores/competidores) de forma correcta. En este caso en particular, el contador de puntaje se especializa en relatar un game de tenis.

La característica principal de este programa es el desarrollo de dos procedimientos distintos para llevar a cabo el contador. En este primer método, se busca que el programa admita valores ingresados por el usuario de forma manual que determinen los ganadores de cada punto. Mientras que, en el segundo estilo de juego consiste en una simulación del desarrollo del game, donde los ganadores son elegidos aleatoriamente.

3. DESARROLLO DEL TRABAJO

En resumen, el desarrollo del programa sería así: En primer lugar, tenemos el if main que nos dirigirá a la función Main(), donde se ejecutará una bienvenida al programa y se nos dirá qué modo de juego queremos jugar. Allí se abre un input en el cual solo podemos poner dos valores válidos (“manual mode” o “simulation mode”), de lo contrario llamaremos a la función “select valid mode” que hará un while hasta que el mode sea igual a una de las dos condiciones. Una vez que el usuario decide poner una de esas opciones válidas, se abre su respectiva función, dependiendo de que haya elegido; si eligió modo manual se abrirá “Manual:mode” y se eligió modo simulación entrara a la función“Simulation Mode”. A partir de ahora el código se divide en estas dos funciones grandes, las cuales tienen otras funciones pequeñas genéricas que son

importadas y funcionan para ambas partes del código, algunas se usan en el archivo principal (ej: select valid mode).

- Cuando se elige Manual Mode, se ejecuta un string que dice “Entering Manual Mode” y luego importa la función Manual Mode(), esto es para que el usuario sepa qué modo se está por ejecutar. Cuando se entra al archivo de Manual Mode-Play Manual-se importan las funciones que están en otros archivos de la carpeta, luego se entra a la función. Lo primero que hace la función es establecer parámetros: (1) *score_p1 = 0*: almacenará los puntos del primer jugador; (2) *score_p2 = 0*: almacenará los puntos del segundo jugador; (3) *Player_won = False*: Este parámetro se establece para que el while que le sigue pueda ejecutarse indefinidamente hasta que uno de los dos jugadores gane. Luego, hay otra entrada de valores para el usuario, esta vez es para almacenar el nombre de los jugadores, mientras que el usuario no ponga dos nombres iguales, el programa los aceptara fácilmente (esto se debe gracias al select_valid_player) y los igualara a sus parámetros. Después, nos metemos en nuestro while indefinido (while == False) donde le pedimos al usuario que entre el ganador del punto, donde no puede elegir un valor distinto de los dos jugadores (esto se debe gracias al select_valid_mode), el punto se guarda en la variable ‘point winner’. Teniendo esta informacion en cuenta, llamamos a otra función, que traslada ese string-‘point winner’-a una suma para uno de los dos puntajes, (dependiendo de quién ganó) y los imprime mapeando cada valor de números con un elemento de una lista para que se pueda trasladar a formato correcto de contador de tenis, esta función se llama ‘print_score’. En print_score pedimos que Random_number = 2 (esto lo hago para que en el print_score solo haga la condición del ‘point_winner’ y se anule la de la parte

random). Eso lo igualamos a las variables score_p1 y score_p2 y repetimos el proceso, hasta que la última función del modo-'win_conditions'-analiza que existe un caso donde algún jugador gana y entonces se muestra que ganó el jugador, se rompe el while y termina el código.

- El modo simulación es muy parecido al modo Manual (porque usa casi las mismas funciones). Cuando se elige Simulation Mode, se ejecuta un string que dice "Entering Simulation Mode" y luego importa la función Simulation Mode(), esto es para que el usuario sepa qué modo se está por ejecutar. Cuando se entra al archivo de Simulation Mode-Play Random()-se importan las funciones que están en otros archivos de la carpeta, luego se entra a la función. Lo primero que hace la función es establecer parámetros: (1) *score_p1 = 0*: almacenará los puntos del primer jugador; (2) *score_p2 = 0*: almacenará los puntos del segundo jugador; (3) *Player_won = False*: Este parámetro se establece para que el while que le sigue pueda ejecutarse indefinidamente hasta que uno de los dos jugadores gane. Luego, hay otra entrada de valores para el usuario, esta vez es para almacenar el nombre de los jugadores, mientras que el usuario no ponga dos nombres iguales, el programa los aceptara fácilmente (esto se debe gracias al select_valid_player) y los igualara a sus parámetros. Le pedimos mediante otro input que elija qué modo de simulación quiere, si ver todo el desarrollo del game o solo el resultado. Acá también el usuario no puede elegir un valor distinto de los dos modos (esto se debe gracias al select_valid_mode). Después, nos metemos en nuestro while indefinido (while == False) donde pedimos que Player3 = 2 (esto lo hago para que en el print_score solo haga la condición del Random_number y se anule la de la parte manual), y también pedimos que

se genere un número aleatorio entre 0 y 1. A pesar de que haya dos modos ambos son iguales aunque uno no tenga sleep (suspensión de la función sobre un determinado tiempo): llamamos a otra función, que traslada ese número aleatorio-‘Random_number’-a una suma para uno de los dos puntajes, (dependiendo de quién ganó) y los imprime mapeando cada valor de números con un elemento de una lista para que se pueda trasladar a formato correcto de contador de tenis. Eso lo igualamos a las variables score_p1 y score_p2 y repetimos el proceso, hasta que la última función del modo-’win_conditions’-analiza que existe un caso donde algún jugador gana y entonces se muestra que ganó el jugador, se rompe el while y termina el código.

4. EXPLICACIÓN DE ALTERNATIVAS CONSIDERADAS Y ESTRATEGIAS ADOPTADAS

Para el desarrollo del código, plante para el principio un archivo principal el cual solo tenga tres funciones: (1) La función ‘main’ que sea donde se arranque la ejecución del programa; (2) La función ‘Play Manual’ donde se busca importar el modo manual; (3) La función ‘Play Simulation’ donde se busca importar el modo random/simulación. Luego, en base a eso, crear dos archivos extra donde se genera una función en cada uno que desarrolle cada modo. Y para finalizar, el resto de las funciones en archivos separados que luego se importan para poder ejecutar el ‘manual mode’ y el ‘simulation mode’, o para también importar en el archivo ‘main’ de ser necesario.

Dentro del código, la única alternativa que considere y logre aplicar sería respecto al tema de la suma de puntos y las condiciones para ganar. En un principio, decidí implementar una cadena de

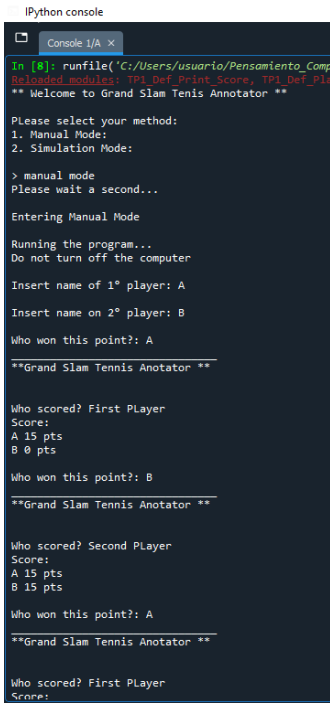
‘ifs’ y ‘el ifs’ en conjunto para determinar la suma de valores porque no encontraba manera de sumar de forma no periódica (es decir, primero y segundo sumar 15, luego sumar 10, luego convertir a string, etc), por lo tanto escribí un if por cada posible resultado existente para cada jugador. Lo mismo me ocurrió con las condiciones para poder ganar, al no poder aplicar una función general, escribí un if para cada caso, el cual no estaba bueno porque repetía demasiado código y se hacía pesado de leer (sumado a que le baja el rendimiento al programa).

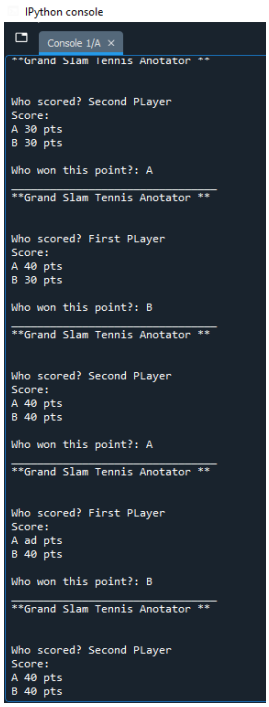
Con el paso del tiempo, fui logrando componer una función para cada respectivo problema. En el caso de los ifs, logre hacer que la función con contador que va sumando los puntajes logre mapear los números de ese contador con una lista de strings (teniendo que poner los números como strings) y no tener que preocuparme por el cambio de tipo de valor (Hace poco también me di cuenta que podría haber hecho str() e int() pero seguía usando muchos ifs). Respecto al caso de las condiciones para ganar, aplique un único if que requería que un jugador tenga 4 o más puntos y el otro jugador tenga la misma cantidad de puntos menos 2. Con esta condición se cumplen todas las posibles condiciones para ganar en una línea de código.

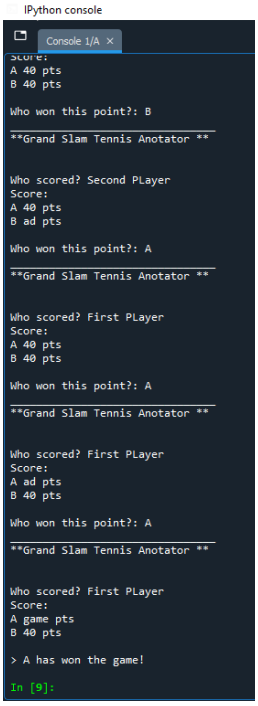
5. RESULTADOS DE EJECUCIONES

Los resultados de ejecución son los siguientes:

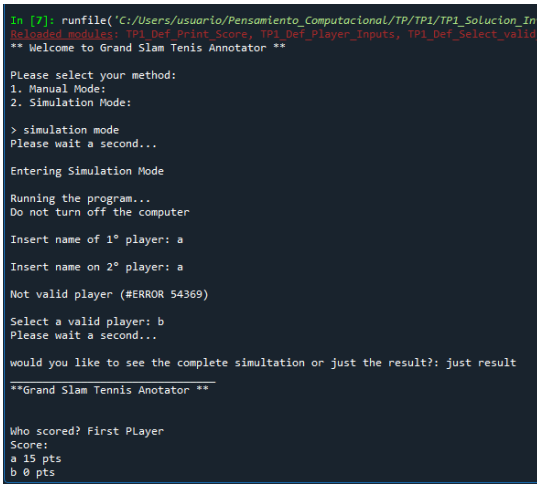
Modo Manual:



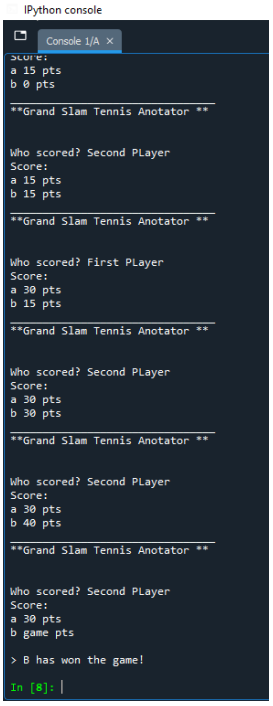




Modo Simulación:



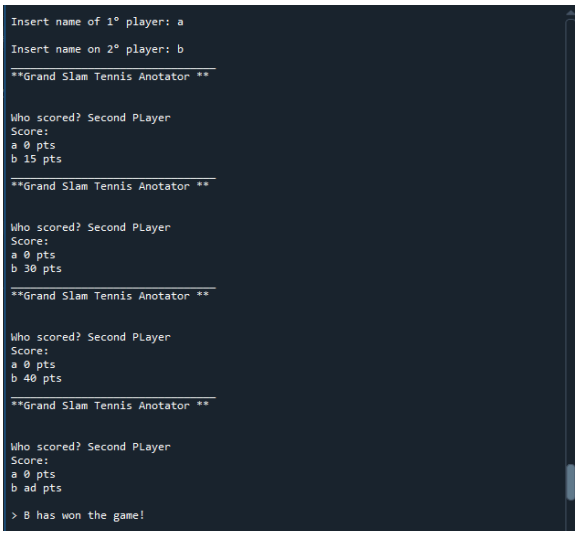
→



6. RESEÑA SOBRE PROBLEMAS ENCONTRADOS Y SOLUCIONES

Durante el desarrollo del trabajo no me encontré con errores muy complejos que lleven mucho tiempo descifrar, pero me encontré con cierto tipo de error igualmente:

1. el primero de todos fue que en un principio, en la parte de simulación solo me dejaba sumar de un lado, y cuando sumaba el otro jugador, no se imprimía en la pantalla:



Por lo tanto, al principio, pensé que era porque me faltaba un print en el archivo donde estaba importando mi función, pero no, el problema era que había puesto en ambos condicionantes del print score que si Random_number era = 0 sumaba el jugador, por lo tanto

```
if ganador_punto == Player1 or Ran_num == 0:
    score_p1 += 1
    print("")
    print("Who scored? First PLayer")
elif ganador_punto == Player2 or Ran_num == 0:
    score_p2 += 1
```

7. BIBLIOGRAFÍA

Author, Unknown. *"Importar Archivos Propios En Python - Uso De Import."* Parabyte 's Blog, 10 Nov. 2020, https://parzibyte.me.

Kapoor, Lakshay. *"Funciones De Temporizador En Python."* Delft Stack, 13 July 2021, https://www.delftstack.com.

Lozano Gómez, Juan José. *"Generar Números Aleatorios En Python. Funciones Principales."* J2LOGO, 16 Jan. 2022, https://j2logo.com.

Diaz, Tomas. "Diagrama De Flujo - Trabajo Practic #1." *Lucid Visual Collaboration Suite: Log In*, https://lucid.app/lucidchart.

8. INDICACIONES DE EJECUCIÓN DEL PROGRAMA

Como último pero no menos importante, están las indicaciones de ejecución del programa. Luego de varias pruebas y testeos, se puede afirmar que en el programa no debería haber espacio para errores (por lo menos hasta donde entiendo yo), en consecuencia, el programa se debería ejecutar correctamente. Por lo que el proceso de ejecución del programa sería el siguiente:

1. Abrir spyder(anaconda3)
2. Abrir la carpeta que contiene los archivos con terminación ".py" o que digan "Archivo PY"
3. Seleccionar el archivo "TP1 Solución Inicial"
4. Una vez dentro del archivo, presionar el botón de "play" (símbolo de flecha) o tocar el botón f5 para poder ejecutar el programa.
5. Una vez dentro del programa, se le pedirá que ingrese el modo de juego, para ingresar al modo manual escriba "manual mode" y para el modo simulación escriba "simulation mode" (puede escribir con mayúsculas o minúsculas pero asegúrese de no cambiar la sintaxis).

6. Si abre el modo simulación, le aparecerá otra entrada de datos la cual pertenece a si quiere desplegar todo el game o solo el resultado, para obtener solo el resultado escriba “just result” y para obtener el desglose completo escriba “complete” (puede escribir con mayúsculas o minúsculas pero asegúrese de no cambiar la sintaxis).
7. Por último, los nombres de los jugadores pueden ser cualquier valor que se quiera, pero, es estrictamente obligatorio poner dos nombres distintos para cada jugador para que el código se pueda ejecutar y no le niegue la entrada de valor.