Universidad de SanAndrés

Pensamiento Computacional

Trabajo Práctico No. 1

Anotador de game de tenis

Alumno: **Bruno Castagnino Rossi**
Mail: **bcastagninorossi@udesa.edu.ar**
Marzo 2022

Índice

Objetivos del trabajo

El objetivo del presente trabajo práctico es desarrollar un programa en lenguaje Python que permita al usuario manejar el indicador de puntaje de cada jugador durante un game de tenis tipo singles.

A lo largo del mismo se busca poder implementar los conocimientos de programación estudiados en la materia hasta este punto incluyendo los siguientes temas:

- Variables y tipos
- Funciones
- Condicionales
- Ciclos
- Secuencias

Desarrollo del trabajo, Alternativas consideradas y estrategias adoptadas

Inicialmente intenté confeccionar un diagrama de flujo buscando la manera de contabilizar un game de tenis, independientemente de la manera en que se realizara el input de cada ganador.

Como se puede ver en la consigna del trabajo práctico, la principal dificultad de esto reside en que la puntuación en tenis se lleva de manera no continua, es decir que cada punto ganado no se contabiliza de la misma forma (de 0 puntos se pasa a 15, luego 30, luego 40, etc.) y como desafío extra se da en modo 'ida y vuelta', queriendo decir que luego de empatar 40-40, si un jugador gana el punto siguiente pasa a estar en ventaja (Ad). Pero si posteriormente pierde el punto la puntuación vuelve a ser 40-40.

En consecuencia al no lograr descubrir un patrón uniforme para la suma de puntos intenté desarrollar un programa que fuese analizando paso a paso con múltiples "if"s el puntaje real de cada jugador y en base a los mismos definía cuánto se tenia que sumar al ganador. (Ver fig 1)

Esta estrategia resultó ser poco eficiente, con un diagrama poco abstracto que resultaba en una extension muy larga y dificultades a la hora de hacer los "idas y vueltas"

```
if win == p1:

    if score1 == 40:
        print(a, wingame)
        break

    elif score1 == 30:
        score1 = 40

    else:
        score1 += 15

elif win == p2:

    if score2 == 40:
        print(b, wingame)
        break

    elif score2 == 30:
        score2 = 40

    else:
        score2 += 15
```

fig 1

Fue entonces cuando opté por asignarle a los posibles puntajes de cada jugador un valor entero del 1 al 5, cada uno de ellos teniendo una correlación con un puntaje en la notación de tenis:

0=0

1=15

2=30

3=40

4= Ad

5= Game ganado

Esta estrategia resultó más sencilla a la hora de sumar puntos pero requirió la confección de una función capaz de transformar dichos valores a los puntos en notación de tenis.

Continuando, para avanzar en el desarrollo del game noté que debido al 'ida y vuelta' propio del tenis, es virtualmente imposible desarrollar un

programa para este objetivo sin utilizar ciclos. Esto se debe a que en teoría si ninguno de los jugadores logra ganar 2 puntos seguidos partiendo de una puntuación de 40-40 el game continuaría para siempre, y el código debería ser de una longitud infinita.

Con esto en mente desarrollé un ciclo que se perpetua siempre y cuando ninguno de los jugadores alcance una puntuación de 5 que implicaría que alguno de los 2 ha ganado.

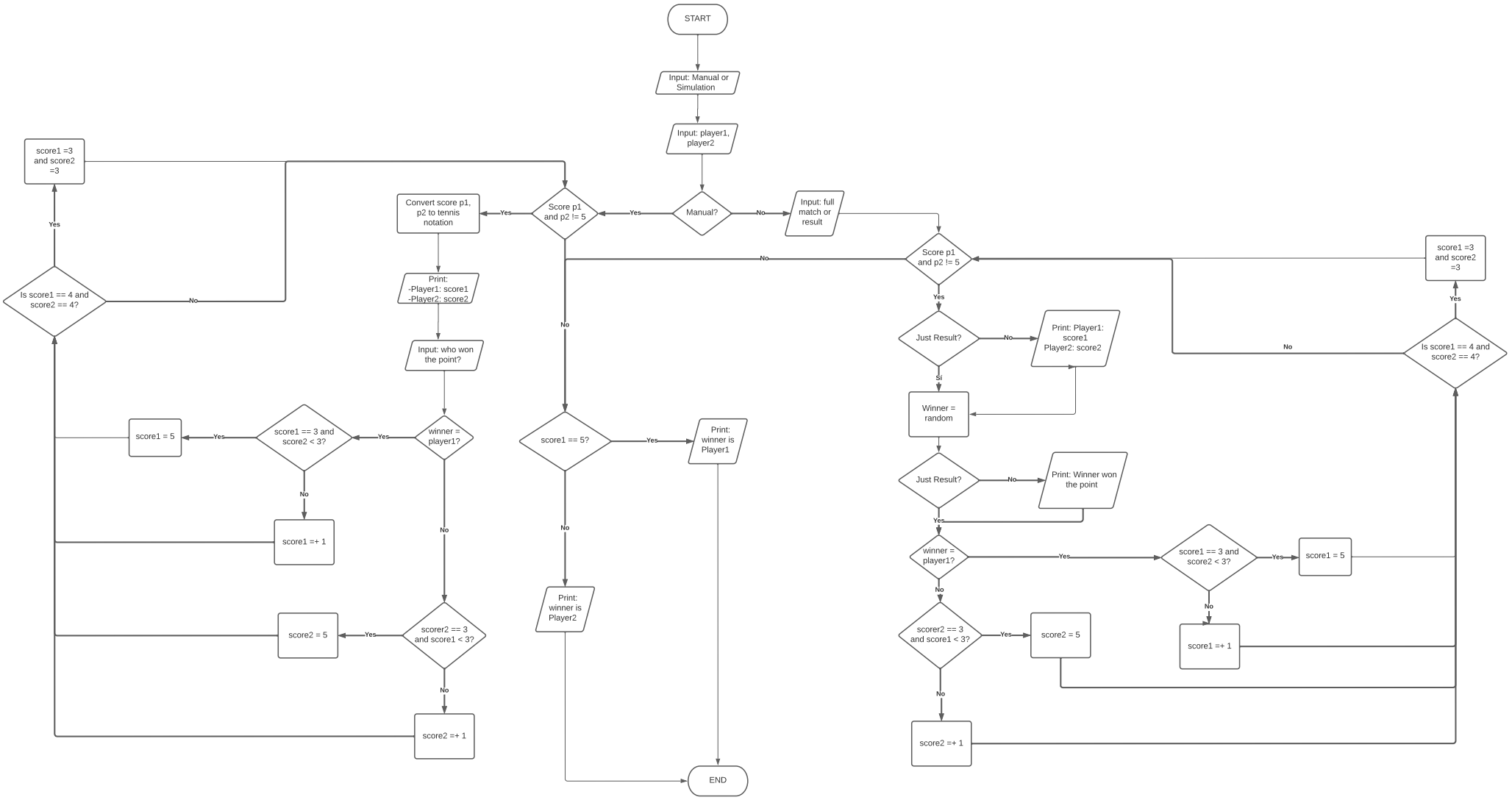
Para resolver el problema del ida y vuelta utilicé un condicional que evalúa si ambos jugadores han llegado a tener 4 puntos, lo cual implicaría que la puntuación es Ad / Ad. En caso de darse esta condición reasigna los valores de ambos puntajes a 3, que se traduce nuevamente en 40 / 40.

Una vez finalizada la lógica detrás de la cuenta de los puntos en el diagrama implementé un condicional que abría 2 ramas, una para la ejecución manual del programa y otra para la simulación. De seguirse esta última otro condicional evalúa el input del usuario para definir si se desea visualizar cada punto de la simulación o solamente el resultado final del game. Definidos estos condicionales implementé el mismo algoritmo para el modo manual y para el modo automático, salvando el hecho de que el ganador de cada punto en el modo manual sería introducido por el usuario y en el modo simulación sería definido por una función tipo random.

De este modo terminé confeccionando el siguiente diagrama de flujo:

(Ver siguiente página)

TP1
VIRTUA TENNIS GAMMA



A la hora de escribir el código, llegué a la conclusión de que las funcionalidades del mismo eran muy redundantes para los distintos modos de ejecución (lo cual puede observarse en el diagrama de flujo) y que las mismas podían resumirse en:

1- Tomar los inputs del usuario (Nombre de los jugadores, modo de ejecucion)

2- Actualizar el puntaje de cada jugador a partir del ganador de cada punto

3- Convertir los puntajes representativos (escala 1-5 mencionada anteriormente) en puntajes reales de tenis

4- Definir qué tipo de modo se esta utilizando y ejecutar los ciclos acorde a ello, ya sea pidiendo el ganador manualmente al usuario o generándolo automáticamente

Por este motivo fue que en el código definí 4 funciones, una para cada uno de estos objetivos:

-La función **main()** Pedía los inputs al usuario de los nombres de cada jugador y los modos de ejecución. También ejecuta la función del desarrollo del game que es precisamente **game()**.

-La función **showscore()** transforma los valores de la escala 1-5 en notación de tenis

-**Whowon()** actualiza los puntajes según el ganador

-**Game()** mantiene el ciclo previamente mencionado, tomando también como parámetro el modo de ejecución que desea el usuario.

De haberse elegido el modo manual, el usuario deberá ingresar quién ganó cada punto y se imprimirán los resultados de cada punto en la consola

De haberse elegido el modo simulación, se solicitará al usuario ingresar el submodo deseado (mostrar cada punto o solo el ganador) y luego se generará un ganador de manera randomizada a parir de una tupla conformada por los nombres de los 2 jugadores.

A la hora de elegir el tipo de ciclo, elegí uno de tipo indefinido ya que no se sabe de antemano cuántos ciclos se requerirán antes de que se alcance la condición deseada. Por esto implementé el mismo utilizando un condicional del tipo ‘while’.

Para randomizar la elección del ganador en el modo simulación, realicé un import de la librería random (el cual se realiza solo si se cumple la condición de que se desea ejecutar este modo) y de la misma utilicé la función .choice que selecciona aleatoriamente a uno de los dos jugadores de una tupla previamente conformada para dicho fin.

Finalmente, en cualquiera de los modos de ejecución, una vez que uno de los jugadores alcanza un puntaje de 5, el ciclo “while” de la función game() se corta, llevando al programa a imprimir por consola el nombre del ganador y finalizando la ejecución del mismo.

Una vez que había desarrollado el código y que el mismo funcionaba, tomé la decisión de no definir variables globales, las cuales hasta ese momento eran player1, player2, Mode y submode que eran ingresadas por un input del usuario dentro de main().

Para lograr este objetivo definí 3 nuevas funciones, **p1(), p2() y mod()** que cumplen la misma funcion de tomar el input del usuario y retornar un string con el nombre de cada jugador y el modo de ejecución deseado respectivamente.

Estas funciones son pasadas como parámetro a la función game() que es la única que se ejecuta dentro de main().

Resultados de ejecución

- **Entradas normales**

Modo Manual

```
 %%% VIRTUA TENNIS GAMMA %%%
Ingrese el modo de ejecución que desea:
1. MANUAL
2. SIMULACION
1
Ingrese el nombre del primer jugador: Sampras
Ingrese el nombre del segundo jugador: Willy Vilas
-Sampras: 0
-Willy Vilas: 0
Quien ha ganado el punto? Sampras
-Sampras: 15
-Willy Vilas: 0
Quien ha ganado el punto? Willy Vilas
-Sampras: 15
-Willy Vilas: 15
Quien ha ganado el punto? Willy Vilas
-Sampras: 15
-Willy Vilas: 30
Quien ha ganado el punto? Sampras
-Sampras: 30
-Willy Vilas: 30
Quien ha ganado el punto? Sampras
-Sampras: 40
-Willy Vilas: 30
Quien ha ganado el punto? Willy Vilas
-Sampras: 40
-Willy Vilas: 40
Quien ha ganado el punto? Willy Vilas
-Sampras: 40
-Willy Vilas: AD
Quien ha ganado el punto? Willy Vilas
Willy Vilas ha ganado el game!
```

Modo Simulación - Desglose completo

```
%%% VIRTUA TENNIS GAMMA %%%
Ingrese el modo de ejecución que desea:
1. MANUAL
2. SIMULACION
2
Ingrese el nombre del primer jugador: Pete Sampras
Ingrese el nombre del segundo jugador: Willy Vilas
Ingrese la modalidad de simulación:
1. DESGLOSE COMPLETO
2. SOLO MOSTRAR GANADOR
1
-Pete Sampras: 0
-Willy Vilas: 0
Quien ha ganado el punto? Pete Sampras
-Pete Sampras: 15
-Willy Vilas: 0
Quien ha ganado el punto? Pete Sampras
-Pete Sampras: 30
-Willy Vilas: 0
Quien ha ganado el punto? Willy Vilas
-Pete Sampras: 30
-Willy Vilas: 15
Quien ha ganado el punto? Willy Vilas
-Pete Sampras: 30
-Willy Vilas: 30
Quien ha ganado el punto? Willy Vilas
-Pete Sampras: 30
-Willy Vilas: 40
Quien ha ganado el punto? Willy Vilas
Willy Vilas ha ganado el game!
```

Modo Simulación - Solo resultado

```
%%% VIRTUA TENNIS GAMMA %%%
Ingrese el modo de ejecución que desea:
1. MANUAL
2. SIMULACION
2
Ingrese el nombre del primer jugador: Pete Sampras
Ingrese el nombre del segundo jugador: Willy Vilas
Ingrese la modalidad de simulación:
1. DESGLOSE COMPLETO
2. SOLO MOSTRAR GANADOR
2
Pete Sampras ha ganado el game!
```

- **Entradas anormales**

Presionar return sin haber ingresado un modo válido:

```
%%% VIRTUA TENNIS GAMMA %%%
Ingrese el modo de ejecución que desea:
1. MANUAL
2. SIMULACION
   #<-
Ha ingresado un modo inválido, ingrese el modo de ejecución
nuevamente:
```

Presionar return habiendo ingresado un modo inválido:

```
%%% VIRTUA TENNIS GAMMA %%%
Ingrese el modo de ejecución que desea:
1. MANUAL
2. SIMULACION
0   #<-
Ha ingresado un modo inválido, ingrese el modo de ejecución
nuevamente:
```

Ingresar un jugador nulo:

```
 %%% VIRTUA TENNIS GAMMA %%%
Ingrese el modo de ejecución que desea:
1. MANUAL
2. SIMULACION
2
Ingrese el nombre del primer jugador: #<-
Ha ingresado un término inválido, ingrese el nombre
nuevamente:
```

Ingresar un jugador ganador distinto de los definidos inicialmente:

```
 %%% VIRTUA TENNIS GAMMA %%%
Ingrese el modo de ejecución que desea:
1. MANUAL
2. SIMULACION
1
Ingrese el nombre del primer jugador: PEPE
Ingrese el nombre del segundo jugador: MOMO
-PEPE: 0
-MOMO: 0
Quién ha ganado el punto? Roberto #<-
Ha ingresado un término invalido, ingrese qué jugador ha
ganado el punto:
```

Ingresar un jugador ganador con distinto casing que los definidos inicialmente (el programa lo toma como input válido):

```
%%% VIRTUA TENNIS GAMMA %%%
Ingrese el modo de ejecución que desea:
1. MANUAL
2. SIMULACION
1
Ingrese el nombre del primer jugador: PEPE
Ingrese el nombre del segundo jugador: MOMO
-PEPE: 0
-MOMO: 0
Quién ha ganado el punto? pepe #<-
-PEPE: 15
-MOMO: 0
```

Problemas encontrados y soluciones

Los principales problemas que encontré fueron relacionados a cómo manejar los inputs no válidos por parte del usuario.

La manera en que solucioné esto en todos los inputs fue la misma, utilizando un ciclo indefinido tipo ‘while' cada vez que pido al usuario que haga un input. El ciclo evalúa el input del usuario, y mientras que el mismo ingrese un input no válido vuelve al inicio, imprimiendo en la consola que el mismo es efectivamente inválido y solicitando un nuevo input.

Solo al recibir una entrada válida el ciclo se corta y puede continuar la ejecución del programa.

El otro problema que tuve relacionado a los inputs del usuario fue decidir si quería darle relevancia a las diferencias en el casing de los inputs. Terminé decidiendo que no era un factor relevante, es decir, si al momento de definir el nombre de un jugador el usuario ingresa ‘ROBERTO' o ‘Roberto’, pero al momento de ingresar el ganador de un punto ingresa ‘roberto' creo que la validez del input debe ser la misma.

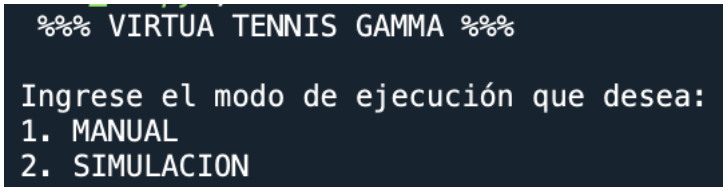
Para lograr esto, al momento de comparar el input de el ganador de un punto con los nombres implementé la función .lower() de ambos lados de la comparación, para de este modo quitarle la relevancia a las diferencias de casing.

Bibliografía

- https://www.geeksforgeeks.org/python-random-module/
- https://www.geeksforgeeks.org/isupper-islower-lower-upper-python-applications/
- https://stackoverflow.com/questions/3898572/what-are-the-most-common-python-docstring-formats
- https://problemsolvingwithpython.com/08-If-Else-Try-Except/08.06-Flowcharts/

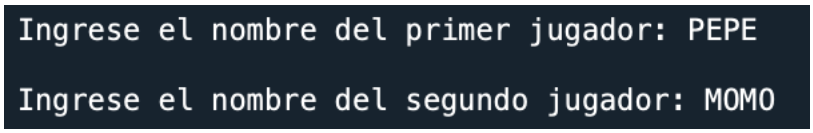
Indicaciones para la ejecución

1) Seleccionar modo de ejecución:



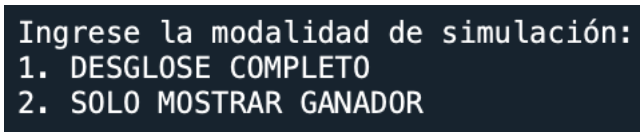
- 1 = modo manual en el que se deberá ingresar manualmente el nombre del jugador que ha ganado cada punto
- 2 = modo simulación en el que se asignará al ganador de cada punto aleatoriamente

2) Ingresar el nombre de los jugadores (los inputs válidos son todos menos un input nulo):

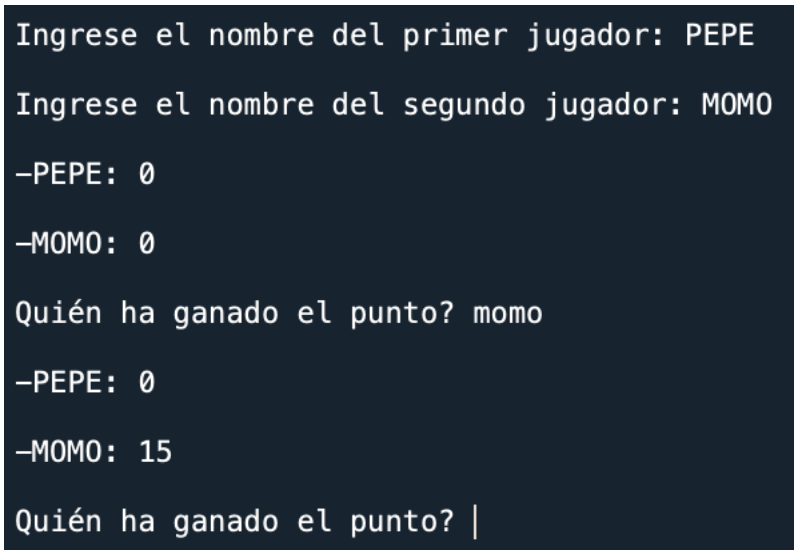


3) De haber seleccionado el modo de ejecución 2 (Simulación) deberá seleccionar el submodo deseado:

- 1 = Desglose completo. Muestra el resultado de cada punto impreso en la pantalla
- 2 = Solo mostrar ganador. Muestra en la consola solo el ganador del game



4) De haber seleccionado el modo manual deberá ingresar el nombre del jugador que ganó el último punto. El programa no diferencia entre mayúsculas y minúsculas:



5) Ya sea en el modo MANUAL o SIMULACION, al final del game el programa mostrará en la consola al ganador del mismo y finalizará su ejecución: