Trabajo Práctico N°1 – Pensamiento Computacional

Universidad de San Andrés

"Simulador de Tenis en Python"

Nombre: Pedro Santiago Gentil

E-mail: pgentil@udesa.edu.ar

Objetivo:

El objetivo de este trabajo es realizar un programa interactivo en Python donde el usuario pueda simular un *game* de tenis, de modo manual y simulado. El programa debe poder mostrar los puntajes de cada jugador en el *game*, cuyos nombres son elegidos por el usuario, mientras se juega en cualquiera de los dos modos. En el modo manual, por cada jugada, el usuario debe poder elegir quien ganará el punto y quien lo pierde. Al final, el usuario elije quien va a ganar el *game.* En el modo simulado, el usuario solo debe elegir el nombre de los jugadores, pero por cada jugada que pase, el marcador debe aparecer junto con el ganador del punto. En la creación del programa se deben utilizar funciones, variables y tipos, secuencias, condicionales y ciclos, y diagramas de flujo

Asimismo, el trabajo mostrará el desarrollo del programa, junto con explicaciones sobre el camino tomado para realizarlo. Ademas, los errores que se cometen y los problemas encontrados junto con las soluciones van a ser expuestos.

Desarrollo:

El programa hecho está compuesto por tres archivos *Python*. El principal, llamado *main*, y los secundarios, cuyos nombres son *manual* y *simulado*. Dentro de *main* se definió una función sin parámetros con el mismo nombre, y se importaron los archivos secundarios. La función *main()* contenía diferentes variables, tal como los nombres de los jugadores como entradas hechas por el usuario, los puntos de cada jugador como enteros, y una tupla con los diferentes puntajes reglamentarios del tenis. También se definió una entrada para que el usuario decida que modo de juego quería jugar.

Dentro del archivo *manual.py*, se definió una función con el mismo nombre que el archivo, con los nombres de las variables declaradas en el *main()* como parámetros (se les puso el mismo nombre para no confundir). Se hizo lo mismo en *simulado.py*. Tras hacer eso, se llamaron a las dos funciones que se crearon dentro de *main()* y se les puso los valores de las variables dentro de los argumentos de la función.

```
def main():
    print ("TeniSim")
    while True:
        while True:
            mode = int(input("""Elija en que modo quiere jugar:
- Ingrese '1' si quiere jugar el modo manual
- Ingrese '2' si quiere jugar el modo simulado
> """))
            if mode in (1, 2):
                break
            else:
                print("No ha seleccionado ningun modo de juego.")
        name1 = input("Nombre del jugador 1: ")
        name2 = input("Nombre del jugador 2: ")
        puntosp1 = 0
        puntosp2 = 0
        puntajes = (0, 15, 30, 40, "Adv.", "Win")
        if mode == 1:
            manual.manual(name1, name2, puntosp1, puntosp2, puntajes)
            break
        if mode == 2:
            simulado.simulado(name1, name2, puntosp1, puntosp2, puntajes)
            break
```

Ilustración 1 Código de la función main() al terminar de desarrollar el código y las variables dentro de esta.

En la función *manual()* dentro de *manual.py*, se encuentra la parte manual del programa, es decir con la intervención del usuario en el sistema de puntos. Con un *while loop* y diferentes condicionales, se logró basar el programa en las reglas del tenis. El usuario decide con una entrada quien gana el punto de una jugada, y dependiendo de cómo se desarrolle el partido, el programa va a seguir un camino determinado, establecido por condicionales. Si tras una jugada el programa todavía no otorga las condiciones para que se termine el partido, entonces se reinicia el ciclo y el usuario vuelve a elegir quien gana el próximo punto. Tras cada jugada se imprime el marcador del *game.* También hay una impresión al momento en que un jugador gana el *game*, indicando quien fue.

En la función *simulado()* dentro de *simulado.py*, la lógica es la misma que en *manual()*, excepto por los condicionales que determinan quien gana el punto. Como la idea de esta parte del programa es que se simule un partido de tenis automáticamente, lo único que se debe cambiar es la variable que decide que jugador gana la jugada. En la función *manual()*, era la entrada del usuario la que decidía. Pero en este caso, dentro de la función *simulado()* se importó el módulo *random*, y se llamó a la función *random()* para que genere un número entre 0 y 1 aleatoriamente dentro de la

misma variable que contenía la entrada del usuario que decidía el ganador. También se cambiaron los condicionales, que en *manual()* seguían la regla de que si el usuario elegía "1" entonces el primer jugador ganaba la jugada y lo opuesto pasaba cuando elegía "2". En cambio, si la variable ahora era un decimal entre 0.5 y 1, ganaba el primer jugador, y de lo contrario ganaba el segundo.

De esta manera hice que el programa, dividido en tres partes, pueda simular un *game* de tenis en un modo manual y otro automático.

Alternativas consideradas y estrategias tomadas:

Durante el proceso de creación del código tuve dos cambios en mi idea original. El primero fue la definición de variables y su posición en el código. Al principio, quería definir las variables que contenían a los nombres de los jugadores y a sus puntajes dentro de cada función *manual()* y *simulado()*, lo que hacía que, aunque fuesen dos archivos diferentes, el código total tenga más líneas. Esto después lo corregí en una clase tutorial que tuve y puse estas variables dentro del *main()* directamente, y utilice sus valores como los argumentos de las funciones secundarias.

El segundo cambio fue el de sistema de puntajes. No tenia claro como representar los puntajes en el programa, hasta que termine usando una tupla a la que llame "puntajes" con los puntajes reglamentarios del tenis, y otras dos variables, a las que llame "puntosp1" y "puntosp2", que marcaban la posición del elemento llamado dentro de la tupla para representar los puntos de cada jugador.

```
print (f"""El game ahora va
{name1}: {puntajes[puntosp1]} - {name2}: {puntajes[puntosp2]}""")
e;
```

Ilustración 2 Demostración de un f-string dentro de una función print() que imprime el marcador del game

Esto me ayudo, porque me vi más propicio a usar f-strings, lo que me ayudo a acortar bastante el código. En la Ilustración 2 se muestra que variables usaba para imprimir el marcador. Los puntos de los dos jugadores empezaban en 0, y cuando uno de los ganaba, se le sumaba un punto. Esto hacía que el puntaje de ese mismo jugador pase de 0 a 15, ya que se desplaza al siguiente elemento de la tupla que se puede visualizar en la Ilustración 1.

Resultados de ejecuciones:

La mayoría de las ejecuciones que se hicieron fueron al probar el modo manual, ya que fue el primero de los dos modos en crearse, y después para hacer el simulado se recicló casi todo su código, debido a la similitud en la lógica.

```
TeniSim

Elija en que modo quiere jugar:
            - Ingrese '1' si quiere jugar el modo manual
            - Ingrese '2' si quiere jugar el modo simulado
            > 1
No ha seleccionado ningun modo de juego.

Elija en que modo quiere jugar:
            - Ingrese '1' si quiere jugar el modo manual
            - Ingrese '2' si quiere jugar el modo simulado
            >
```

Ilustración 3 Primer problema en consola, al poner el numero 1 como entrada no se activaba el modo manual

El problema de la Ilustración 3 sucedió en la primera prueba del modo manual. El problema residía en el código dentro de *main()*, que estaba mal pensado.

```
while True:
    mode = int(input("""Elija en que modo quiere jugar:
    - Ingrese '1' si quiere jugar el modo manual
    - Ingrese '2' si quiere jugar el modo simulado
    > """))
    if mode != 1 or mode != 2:
        print("No ha seleccionado ningun modo de juego.")
```

Ilustración 4

```
while True:
    mode = int(input("""Elija en que modo quiere jugar:
    - Ingrese '1' si quiere jugar el modo manual
    - Ingrese '2' si quiere jugar el modo simulado
    > """))
    if mode == 1 or mode == 2:
        break
    else:
        print("No ha seleccionado ningun modo de juego.")
```

Ilustración 5

En ningún momento, con el código de la Ilustración 4, se iba a poder salir del *loop*. Entonces modifique el código al de la Ilustración 5. El funcionamiento de ese *loop* era que no se termine el programa en caso de apretar mal un número.

Después se intentó una vez más. Y esta vez el programa dio un error al correrse por los argumentos de la función *manual()*. En el momento de haber corrido el programa tenía llamada a la función, pero sin ponerle los valores de las variables de *main()*.

```
  File "C:\Users\Pedro\Desktop\UNIVERSIDAD\UDESA\Pensamiento
Computacional\PC\TPs\tp1_gentil\tp1_gentil.py", line 37, in <module>
    main()

  File "C:\Users\Pedro\Desktop\UNIVERSIDAD\UDESA\Pensamiento
Computacional\PC\TPs\tp1_gentil\tp1_gentil.py", line 28, in main
    manual.manual()

TypeError: manual() missing 5 required positional arguments: 'name1',
'name2', 'puntosp1', 'puntosp2', and 'puntajes'
```

Ilustración 6

```
def main():
    print ("TeniSim")
    while True:
        while True:
            mode = int(input("""Elija en que modo quiere jugar:
            - Ingrese '1' si quiere jugar el modo manual
            - Ingrese '2' si quiere jugar el modo simulado
            > """))
            if mode == 1 or mode == 2:
                break
            else:
                print("No ha seleccionado ningun modo de juego.")
        name1 = input("Nombre del jugador 1: ")
        name2 = input("Nombre del jugador 2: ")
        puntosp1 = 0
        puntosp2 = 0
        puntajes = (0, 15, 30, 40, "Adv.", "T")
        if mode == 1:
            manual.manual(name1, name2, puntosp1, puntosp2, puntajes)
            break
        if mode == 2:
            simulado.simulado(name1, name2, puntosp1, puntosp2, puntajes)
            break
```

Ilustración 7

La solución se puede ver en la Ilustración 7, donde se pusieron a las variables como los argumentos de las funciones *manual()* y *simulado()*.

Después otro problema que me encontré al seguir corriendo el programa en el modo manual fue la adjudicación incorrecta de puntos, ya que cuando quería darle un punto al segundo jugador, el punto iba para el primero.

```
¿Ahora quien marca? 1
No ha seleccionado a ningun jugador para que marque.
            Porfavor intentelo de nuevo.
pp ha marcado un punto
El partido ahora va
                pp: 40 - cc: 0

¿Ahora quien marca? 2
cc ha marcado un punto
El partido ahora va
                pp: Adv. - cc: 0

¿Ahora quien marca?
```

Ilustración 8

```
if u == 2:
    puntosp1 += 1
    print (f"{name2} ha marcado un punto")
    if puntosp2 == 4 and puntosp1 < 3:
        print (f"{name2} ha ganado el partido")
        break
    if puntosp2 == 5 and puntosp1 == 3:
        print(f"{name2} ha ganado el partido")
        break
    elif puntosp1 == 4 and puntosp2 == 4:
        puntosp1 -= 1
        puntosp2 -= 1
        print (f"""El partido ahora va
        {name1}: {puntajes[puntosp1]} - {name2}: {puntajes[punto
    else:
        print (f"""El partido ahora va
        {name1}: {puntajes[puntosp1]} - {name2}: {puntajes[punto
        u = int(input ("¿Ahora quien marca? "))
else:
    print ("""No ha seleccionado a ningun jugador para que marqu
    Porfavor intentelo de nuevo.""")
```

Ilustración 9

El problema se visualiza en la Ilustración 8, y se solucionó en el código dentro de *manual()*¸ donde directamente el punto aparecía adjudicado para la variable del primer jugador. Esto después lo cambie para que el punto fuese dado al segundo jugador.

Asimismo, seguían los problemas de impresión. En este caso se repetía la adjudicación de el punto para el segundo jugador al estar los dos jugadores empatados “Adv. – Adv.”, que es una situación singular ya que se le deben restar un punto a los dos para que vuelvan a un marcador de “40-40”. En este caso, al empatarlos en esas condiciones, se le restaron los puntos a cada jugador, pero posteriormente se le agrego un punto al segundo jugador sin intervención del usuario.

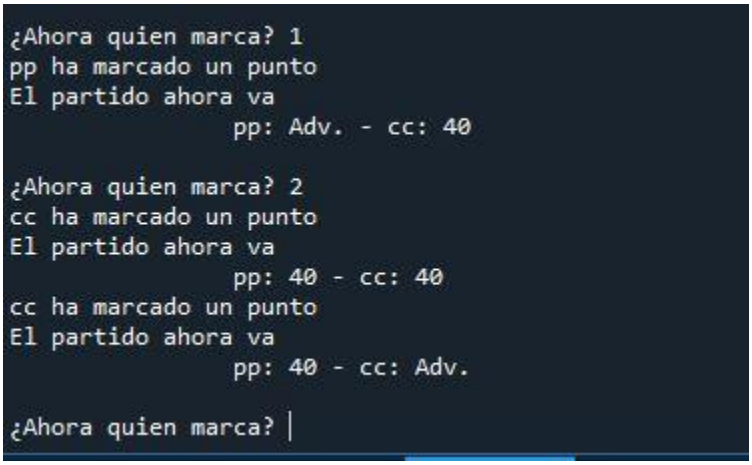

Ilustración 10

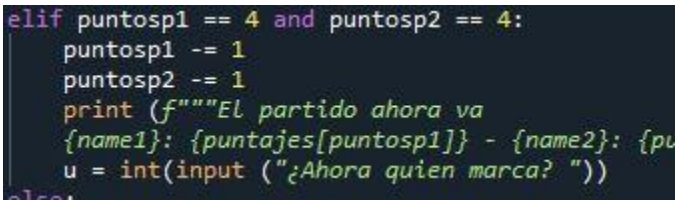

Ilustración 11

En la Ilustración 11, la solución puede no ser muy evidente. Pero el cambio que le hice al código para que este problema no surgiera mas fue cambiarle a esa condición un *if* por el *elif* que se muestra en la imagen. El *if* hacía que el programa no ciclara devuelta hasta antes no pasar por el siguiente condicional

El ultimo problema al correr el modo manual fue con otra situación de condicionales. Utilice tres condicionales para la suma de puntos para cada jugador: uno para otorgarle un punto al primer jugador, otro para otorgarle un punto al segundo, y otro por si el usuario se equivocaba de número y para que el programa no se frene cada vez que pasaba esto.

```
¿Ahora quien marca? 1
No ha seleccionado a ningun jugador para que marque.
            Porfavor intentelo de nuevo.
pp ha marcado un punto
El partido ahora va
                pp: Adv. - cc: 40

¿Ahora quien marca? 2
```

Ilustración 12

```
elif u == 2:
    puntosp2 +=
```

Ilustración 13

Al ponerle un *elif* en vez de un *if* a la segunda condición del código dentro de la función *manual()*, pude solucionar el problema, ya que el problema solo pasaba cuando se quería otorgarle un punto al primer jugador. Como la condición que le seguía ahora era un *elif*, al terminar las sentencias de la primera condición, el *loop* se reiniciaba.

Finalmente, después de crear el modo de juego automático en *simulado()*, con los mismos parámetros y la misma lógica no necesite de arreglos, ya que era código muy parecido al de *manual()*, y los errores se habían corregido al momento de correr esa parte del programa.

```
In [21]: runfile('C:/Users/Pedro/Desktop/UNIVERSIDAD/UDESA/Pensamiento
Computacional/PC/TPs/tp1_gentil/main.py', wdir='C:/Users/Pedro/Desktop/
UNIVERSIDAD/UDESA/Pensamiento Computacional/PC/TPs/tp1_gentil')
Reloaded modules: simulado, manual
TeniSim

Elija en que modo quiere jugar:
- Ingrese '1' si quiere jugar el modo manual
- Ingrese '2' si quiere jugar el modo simulado
> 2

Nombre del jugador 1: pp

Nombre del jugador 2: cc
Ha elegido el modo simulado
El game empieza 0-0
cc ha marcado un punto
El game ahora va
                pp: 0 - cc: 15
cc ha marcado un punto
El game ahora va
                pp: 0 - cc: 30
cc ha marcado un punto
El game ahora va
                pp: 0 - cc: 40
```

Ilustración 14

```
pp ha marcado un punto
El game ahora va
                pp: 15 - cc: 40
pp ha marcado un punto
El game ahora va
                pp: 30 - cc: 40
pp ha marcado un punto
El game ahora va
                pp: 40 - cc: 40
pp ha marcado un punto
El game ahora va
                pp: Adv. - cc: 40
pp ha marcado un punto
pp ha ganado el game
```

Ilustración 15

Tras haber probado unas cuantas veces más el modo manual y automático, no se encontró ningún otro error.

Problemas encontrados y soluciones:

Al momento de armar el código, me encontré con muchos problemas en los condicionales y con los *while loops* en Python. Las funciones específicas de los *ifs*, *elifs* y *else* fueron difíciles de incorporar al código. Igualmente pude solucionar al final los problemas generados por los condicionales probando repetidas veces y practicando diferentes formas de usarlos. Después lo mismo sucedió con los ciclos, pero generalmente los problemas ocurrían debido a errores en el código, como no poner *breaks* donde se debía. El sistema de puntos también fue un poco desafiante, ya que, en el tenis, los puntos en un *game* pasan del 15 al 30, pero después del 30 al 40, entonces con un algoritmo con sumas para modificar una variable directamente iba a ser complicado de realizar. Por eso utilice una tupla con los valores reglamentarios del tenis y una variable que llamase a los elementos de esta, que serían los puntajes de cada jugador, que variaban entre el 0 y el 5 como máximo. De esta manera pude solucionar el problema con el sistema de puntos.

Instrucciones de uso del programa:

- Se debe tener el modulo random.py descargado en la librería estándar de Python antes de ejecutar el programa.
- Se deben de tener los tres archivos que forman parte del programa: *main.py, simulado.py* y *manual.py*.

Bibliografía:

W3schools (2022). Refsnes Data. Recuperado de: https://www.w3schools.com/default.asp Accedido en: marzo de 2022