



Tecnológico de Monterrey

Pensamiento Computacional Para la Ingeniería
Armando Bernardo Nava Ortiz

Proyecto Integrador

Thomas Freund Paternostro A00831997
Carlos Alonso Moreno Alcantar A00831671

20 de octubre del 2020

I. Introducción

No es secreto que en la actualidad que vivimos hoy en día es común que los niños batallen con matemáticas. Esto es perfecto para resolver a través de la programación porque podemos desarrollar un juego que sea atractivo y fácil de usar, para que los niños puedan practicar matemáticas y su desempeño académico tenga mejores resultados en el área de matemáticas. Más específicamente, usamos Python para crear nuestro programa.

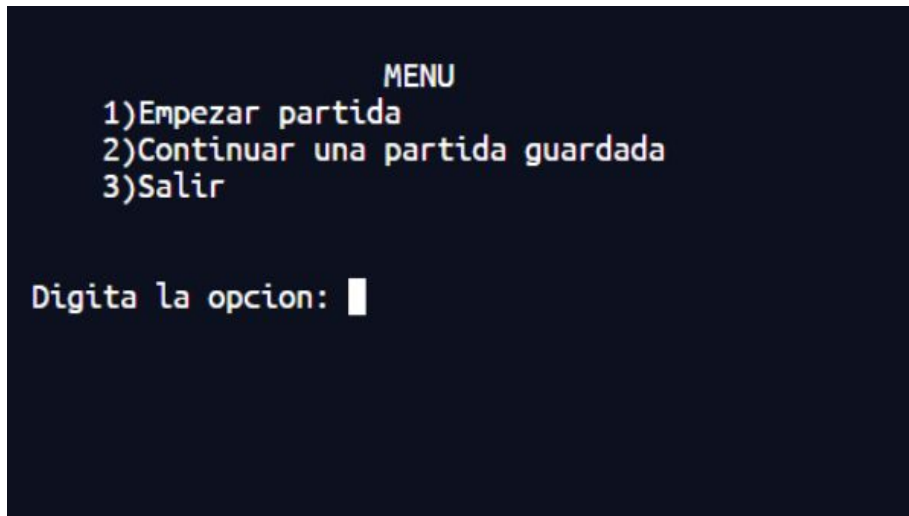
II. Descripción

El problema que se presentó, como fue previamente mencionado, es que hay niños que batallan con matemáticas en la primaria. Es posible que un factor que esté afectando el aprendizaje de los niños sea la cantidad de tiempo que invierten en practicar los temas vistos en clase. Hoy en día a la mayoría de los niños les parece aburrido hacer cosas que no involucren tecnología, o puede que no les llame la atención. La solución que aplicamos fue crear un juego que permita a los niños practicar matemáticas de primaria. Más específicamente para niños de 6to grado. Se tomó la decisión de darle énfasis a las fracciones, ya que son muy importantes no solo para la vida escolar, sino que también en la vida cotidiana. De esta manera los niños encuentran atractiva la idea de practicar matemáticas a través de un juego hecho con programación. Contamos con nuestro conocimiento adquirido en la clase “Pensamiento Computacional Para la Ingeniería”.

Se enfatizó trabajar con la sub competencia de equivalencia de fracciones, ya que durante el resto de la vida de los estudiantes el uso de fracciones va a ser imperativo tanto en su vida académica y cotidiana. Donde trabajar en entender que fracciones con diferentes números pueden valer lo mismo es una habilidad necesaria en el mundo de hoy.

III. Menú del Programa

En cuanto al menú del programa, decidimos incluir las opciones necesarias para que no solo sea bastante útil, si no que a la vez sea sencillo y fácil de usar.



IV. Marco Teórico

Utilizamos muchos elementos que vimos en clase. A continuación hablaremos a detalle sobre cada uno de ellos y cómo los incorporamos a nuestro programa.








Dado que se está trabajando para ayudar a jóvenes de primaria de sexto grado, la competencia elegida fue fracciones y por eso se destaca la historia de la misma.


Historia de fracciones

Las fracciones que conocemos hoy en día no existieron hasta el siglo XVII en Europa, antes de esto no eran consideradas como números, sino como una forma de comparar diferentes cosas. La palabra fracción proviene del latín “fractio” que significa romper (Pumfrey, 2011).

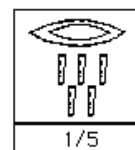
Para entender el origen de las fracciones hay que remontarnos desde los tiempos de los egipcios en los años de 1800 A.C. donde esta civilización ya comenzaba a escribir en fracciones.

El sistema numérico de los egipcios tenían una base de 10 en donde se escribía todo en dibujos que se llaman jeroglíficos.

						
1	10	100	1000	10000	100000	10^6
Egyptian numeral hieroglyphs						


276

Estos escribieron sus fracciones usando lo que hoy en día llamamos fracciones unitarias, que como sabemos la fracción unitaria es aquella que contiene un 1 como numerador. Por ejemplo esta era su forma de representar a lo que hoy conocemos como un quinto:



Existía una enorme desventaja con el sistema egipcio de fracciones, la cual era la dificultad de realizar cálculos. Los egipcios trataron de encontrar la solución a su problema con la creación de diferentes propuestas pero no se tuvo una solución concisa (Pumfrey, 2011).

Adentrándonos en la antigua Roma, podemos observar que las fracciones eran usadas con el fin de describir una parte del todo. Se basaron en la unidad de peso que se llamó “as”. Este estaba compuesto por 12 nocias, por tal motivo sus fracciones se centraban en doceavas.

Un ejemplo:

$1/12$ ---> uncia

$6/12$ ---> semis

1/24 ---> semuncia

1/144 ---> scripulum

Estas fracciones eran al igual que con los egipcios, difíciles de usar en el momento de usar cálculos.

Los babilonios fueron los primeros en crear una forma más clara de representar fracciones. Su sistema numérico se basó en el número 60. Sin embargo, también se basaron en y agruparon en torno al número 10 y por lo cual solo tenían dos símbolos, para unidad y para 10.

1	𐎶	11	𐎶𐎵
2	𐎶𐎶	12	𐎶𐎵𐎶
3	𐎶𐎶𐎶	13	𐎶𐎵𐎶𐎶
4	𐎶𐎶𐎶𐎶	14	𐎶𐎵𐎶𐎶𐎶
5	𐎶𐎶𐎶𐎶𐎶	15	𐎶𐎵𐎶𐎶𐎶𐎶
6	𐎶𐎶𐎶𐎶𐎶𐎶	16	𐎶𐎵𐎶𐎶𐎶𐎶𐎶
7	𐎶𐎶𐎶𐎶𐎶𐎶𐎶	17	𐎶𐎵𐎶𐎶𐎶𐎶𐎶𐎶
8	𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶	18	𐎶𐎵𐎶𐎶𐎶𐎶𐎶𐎶𐎶
9	𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶	19	𐎶𐎵𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶
10	𐎵	20	𐎵𐎵

El principal inconveniente con su sistema es que no contaban con cero ni algo que tuviera el papel del mundo decimal. Esto hizo que la lectura de números fuera confusa y podría llegarse a una doble interpretación entre números enteros y fracciones.

El formato conocido hoy en día fue creado por la civilización india. Su gran avance y funcionalidad de fracciones se debe sus tres principios:

1. Cada figura tiene un símbolo que no es el valor que representa.
2. El valor de la figura depende de su posición.
3. El cero es usado como forma de nada y también toma el lugar en donde faltan unidades.

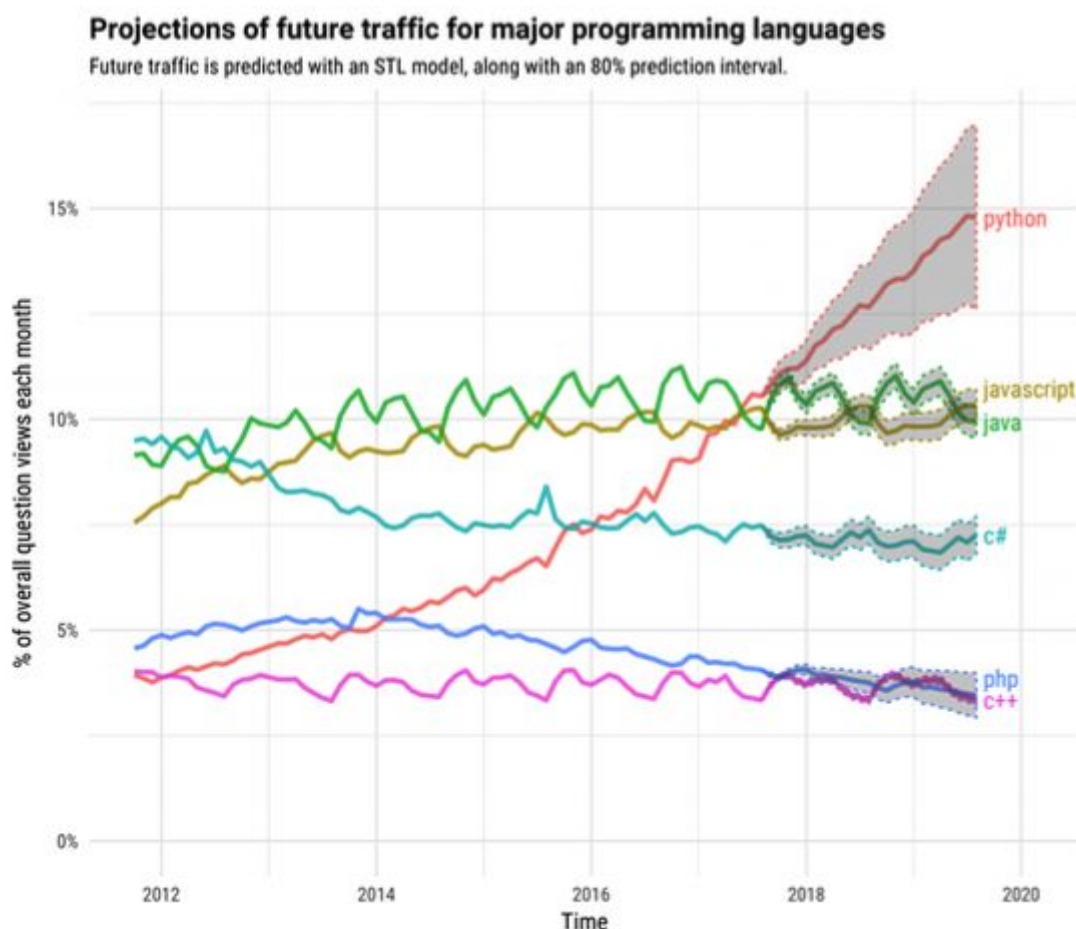
Las fracciones indias eran escritas más como las conocemos hoy, un número como numerador, arriba de otro que representaba el denominador, la única diferencia con las actuales es que no se usaba la línea entre los números(Pumfrey, 2011).

Versiones de Python desde su concepción (Unmalnick)

¿Qué es Python?:

Python es un lenguaje orientado a objetos que significa que todo en Python es un objeto. Es un lenguaje de muy alto nivel que significa que se parece mucho más al lenguaje escrito que otros programas que hablan casi directamente a la computadora como C o C++. Tiene una comunidad enorme y dispone de librerías abiertas que facilitan la programación de casi cualquier cosa imaginable

Según Stack Overflow (la página de preguntas de programación más grande del mundo, con 40 millones de usuarios mensuales) Python es el idioma con más demanda y crecimiento en el mundo dado que la mayor cantidad de preguntas se relacionan a este programa.(Soloaga, 2019)



Proyección por lenguaje de programación (2012-2020)(Soloaga)

¿Por qué usamos Python?

Python es el idioma de uso dado por el curso pero también es efectivo, ya que fue diseñado para ser fácil de leer, y su sintaxis permite a los programadores utilizar menos líneas de código sin sacrificar la calidad del programa. (Soloaga)

Funciones:

Hay dos tipos de funciones: (Sweigart,2019)

- I. **Funciones definidas:** Estas son las que fueron implementadas en el lenguaje para su uso inmediato sin llamar a ninguna librería o módulo externo
 - A. Print(): Permite imprimir en texto o una(s) variables para que el usuario realice una acción o vea lo que está sucediendo en el código.
 - B. len(): Determina la longitud de una cadena o string y le asigna un valor entero.
 - C. input(): Permite al usuario interactuar con el programa al pedirle que realice una acción específica y al mismo tiempo lee e interpreta la respuesta introducida.
 - D. range(): Determina el rango o duración que tendría algo, normalmente se lo asocia con el ciclo for para repetir un número de veces determinadas y de cierta manera .range(Donde empieza, Donde acaba -1, salto por hacer) demuestra cómo se utilizará esta función predefinida para realizar la acción necesaria.
 - E. write(): para escribir un archivo
 - F. open(): para abrir un archivo del tipo especificado, generalmente '.txt'
 - G. close(): Función par cerrar un archivo
 - H. read(): Funciona para leer todo el archivo
 - I. str(): Convierte la información en una cadena de caracteres
 - J. int():Convierte la información en en un entero
 - K. bool(): la información en un valor (False/True)
 - L. float():Conviértela en un valor real
- II. **Funciones no definidas:** Estas funciones son determinadas por el usuario de la siguiente manera: `def nombreDeLaFuncion(parámetros)`. Estas funciones se utilizan con el objetivo de hacer el programa más eficiente porque si se tiene que hacer alguna cosa solo al llamar a la función y sus parámetros puedes correrlo sin tener que copiar y pegar el código el número de veces que sea necesario.

Parámetros

Los parámetros son variables que se declaran en una función para poder utilizarlos dentro de la función.

Variables:

Las variables son espacios de memoria moldeable que pueden tener una información específica y cuando se llama a estas variables en cualquier punto del programa se regresa la información guardada dentro de esa variable. Una variable también solo puede tener un tipo de data específico sea str() o int().

Hay dos tipos de variables:

- I. **Variables locales:** Estas son las variables que se definen dentro de una función, el resultado de esta va a permanecer dentro de la no definida y se podrá usar fuera de la misma si nos la declara como una variable global.
- II. **Variables Globales:** Son definidas en todo el programa, no pertenecen a una función en específico pero todas las funciones en el programa pueden llamarlas si lo determinan.

Y hay dos clasificaciones de variables:

- I. **Variables constantes:** Son todas las que no cambian durante todo el programa como $\pi = 3.14$.
- II. **Variables no definidas:** Estas variables pueden en varios momentos del programa almacenar diferente información. y tener un uso diferente durante el mismo dependiendo del orden del algoritmo.

Tipos de datos:

Cada variable o dato solo puede tener un tipo de dato:

- I. **Booleano:** Este valor es tanto falso o verdadero que corresponde a 0 o 1.
- II. **Int:** Este tipo de dato es un numero entero (ej: 0,1,2,3,4)
- III. **Float:** Este tipo de dato es un número real (ej: 0,2.32, π , $34/3$,.25)
- IV. **None:** Este tipo de dato no tiene una input/output determinado es none
- V. **String:** Este tipo de dato es una cadena de caracteres

Condicionales:

En Python las condicionales se realizan con **if**, **elif**, **else**. Donde se establece con **if** si la condición se cumple o no que da un valor booleano, si la condición se cumple se ejecuta el código dentro del **if**. Si no se cumple va al **elif** para ver si se cumple esa igual como en **if**. Y si en ninguno de esos casos se cumplen va a hacer una tercera que sería **else** que cubre todas las otras que no fueron específicas previamente en las condicionales **if** y **elif**.

Ciclos:

Los ciclos determinan el número de veces que se tiene que realizar una acción hasta que ya no se pueda hacer o hasta que se cumpla.

Hay dos tipos de ciclos:

- I. **Ciclo while:** Los ciclos **while** son aquellos que continúan corriendo hasta que la condición no se cumpla. Se usa un ciclo **while** cuando no se sabe cuántas veces se tiene que repetir una acción y cuando se cumple se puede salir de este.
- II. **Ciclo for:** Los ciclos **for** se utilizan prácticamente para casi todo en Python, estos tienen incluido una variable definida por el usuario que determina el número de repeticiones que se ha realizado. Se lo utiliza para crear y conseguir listas y también se le puede anidar uno adentro del otro para determinar las filas y columnas en el caso de este proyecto. La estructura

Condicionales de prueba:

Las condicionales son utilizadas para determinar si hay errores en el código. Tienen una estructura **try**, **except**, **else** **finally**. Donde se determina si el funciona correctamente y si lo hace se comportaría de la manera designada en **try**. Si no lo hace se ejecuta **except** y si hay excepciones se usa el **else**. El **finally** siempre funcionaria sin importar cuál de ellos se ejecute.

Método

Los métodos son funciones que se adjuntan a listas, variables, entre otros tipos de data donde se llama a una acción específica por parte del método. Dado que Python es un lenguaje de programación de objetos casi todo se puede interpretar como un objeto y eso permite que se pueda designar, mover, adjuntar, o quitar atributos al mismo y en esencia es lo que hace un objeto. Hace una acción específica a su funcionamiento y modifica al dato/objeto con algún tipo de dato particular. Es decir el método puede variar y ser float, string, bool, None, int.

Datos Estructurados

Los datos estructurados utilizan la memoria de la computadora para designar un elemento en un cierto índice o lugar, hay varios tipos de datos estructurados como listas, diccionarios, tuplas; pero en este proyecto solo se trabajo con listas.

I. Listas: Es una estructura de datos con características especiales. Estas nos permiten almacenar cualquier tipo de valor como enteros, cadenas, etc.

A. Matrices: Las matrices son por sí solas no son un tipo de dato en python, estás simplemente son una lista de listas que pueden tomar una interpretación matemática.

Módulos y librerías

Python incluye también una serie de módulos de la librería estándar.

Cada módulo es un programa de python que se carga y contiene un grupo diferente de funciones que pueden ser incluidas en los programas.

Random: Es una librería que está diseñada para generar números aleatorios, la utilizamos para generar números aleatorios desde ciertos rangos específicos para crear las fracciones.

Unicode: Es una librería que usamos en el proyecto y esto llama a una función `unicode()` que lo que hace es tomar data Unicode y lo representa con caracteres ASCII.

Fraction: Es un módulo de Python, está diseñado para evaluar valores fraccionales con un numerador y denominador.

Archivos

Los archivos en python se pueden definir como una secuencia de caracteres los cuales son almacenados en un sitio permanente. En python se pueden realizar diferentes operaciones con los archivos como: crear, abrir, leer, escribir y cerrar.

Para abrir los archivos se utiliza el comando `open()`. Ejemplo de uso: `Archivo = open(Nombrearchivo, modo de acceso)`. Existen diferentes modos de acceso en la función `open()`, los cuales son:

Modo de acceso	Descripción
r	Abre un archivo solo para leer
w+	Abre un archivo, lo lee y sobrescribe en caso de que existe el archivo, si no lo crea
a	Abre un archivo para adjuntar contenido, en caso de que el archivo no existe, lo crea

Los métodos utilizados en el programa para abrir, leer, escribir y cerrar los archivos son: `.open()`, `.read()`, `.write()`, `.close()`

V. Ejemplo de Uso

Para poder usar todo el código por favor instalar los dos módulos adicionales:

[pip install Unidecode](#)

[pip install Fraction](#)

En caso de que no sirva en Thonny o algún otro editor, el código está en este repositorio virtual:

<https://repl.it/@CarlosMoreno12/Python-3#main.py>

(automáticamente descarga las librerías)

Menú>

El código del menú de inicio, donde usamos un try para que solo reconozca valores numéricos y ninguno otro tipo de dato:

```
n = 0
while n != 3:
    print('''
    | | | | | | | | | | MENU
    | 1)Empezar partida
    | 2)Continuar una partida guardada
    | 3)Salir
    | ''')

    while True:
        try:#Solo aceptar valores si son enteros
            n = int(input("Digita la opcion: "))
        except ValueError:#En caso de que digite cualquier otra
            cosa
            print("Digita un numero entero")
            continue
        else:
            break

    if n == 1:
        resp = ''
        parametrosjuego()

        while resp != 'fin': #Ciclo donde se une el tablero
            con el voltear cartas, permite jugar mientras no se
            escriba fin al termino de un turno

            imprimir_tablero()

            voltear_cartas()

            resp = input('¿Desea seguir jugando? (Escriba fin
            para terminar el juego)')
```

Menú en shell:

```

                                MENU
1)Empezar partida
2)Continuar una partida guardada
3)Salir

Digita la opcion: █
```

Cualquier otra opción:

```

                                MENU
1)Empezar partida
2)Continuar una partida guardada
3)Salir

Digita la opcion: 8
Digite una opcion valida

                                MENU
1)Empezar partida
2)Continuar una partida guardada
3)Salir

Digita la opcion: █
```

Opción 1:

```

                                MENU
1)Empezar partida
2)Continuar una partida guardada
3)Salir

Digita la opcion: 1
Digite el numero de filas y columnas del juego:
Digita la dimension de filas (min. 8): █
```

En caso que ponga números menores a 8 en columnas o filas:

```

Digita la opcion: 1
Digite el numero de filas y columnas del juego:
Digita la dimension de filas (min. 8): 5
Digita la dimension de columnas (min. 8): 8
Rango no valido (min. 8)
Digita la dimension de filas (min. 8): █
```

Si se pone un tablero impar también sucede lo mismo pero sale un mensaje que el tablero no puede ser impar:

```

Digita la opcion: 1
Digite el numero de filas y columnas del juego:
Digita la dimension de filas (min. 8): 5
Digita la dimension de columnas (min. 8): 3
El tablero no puede ser impar
Digita la dimension de filas (min. 8): █
```

Si se digita el tablero donde las filas y columnas son mayores a 8 y es par:

```

Digita la opcion: 1
Digite el numero de filas y columnas del juego:
Digita la dimension de filas (min. 8): 8
Digita la dimension de columnas (min. 8): 8

Guardando en archivo

      0      1      2      3      4      5      6      7
0      *      *      *      *      *      *      *      *
1      *      *      *      *      *      *      *      *
2      *      *      *      *      *      *      *      *
3      *      *      *      *      *      *      *      *
4      *      *      *      *      *      *      *      *
5      *      *      *      *      *      *      *      *
6      *      *      *      *      *      *      *      *
7      *      *      *      *      *      *      *      *

Jugador 1: 0      Jugador 2: 0

      Turno: Jugador 1

Primera Carta
Fila primera carta: █
```

Caso arriba de 8*8

Si se digita una carta que no está en el tablero (fuera de rango)

```

Digita la opcion: 1
Digite el numero de filas y columnas del juego:
Digita la dimension de filas (min. 8): 8
Digita la dimension de columnas (min. 8): 9

Guardando en archivo

      0      1      2      3      4      5      6      7      8
0      *      *      *      *      *      *      *      *      *
1      *      *      *      *      *      *      *      *      *
2      *      *      *      *      *      *      *      *      *
3      *      *      *      *      *      *      *      *      *
4      *      *      *      *      *      *      *      *      *
5      *      *      *      *      *      *      *      *      *
6      *      *      *      *      *      *      *      *      *
7      *      *      *      *      *      *      *      *      *

Jugador 1: 0      Jugador 2: 0

      Turno: Jugador 1

Primera Carta
Fila primera carta: 9
Columna primera carta: 9

Rango Invalido
```


Si se digita cartas dentro del juego guarda cada carta y prosigue a la siguiente y a la final pregunta si quieres continuar:

```

0      0      1      2      3      4      5      6      7
*      *      *      *      *      *      *      *      *
1      *      *      *      *      *      *      *      *
2      *      *      *      *      *      *      *      *
3      *      *      *      *      *      *      *      *
4      *      *      *      *      *      *      *      *
5      *      *      *      *      *      *      *      *
6      *      *      *      *      *      *      *      *
7      *      *      *      *      *      *      *      *

Jugador 1: 0      Jugador 2: 0

    Turno: Jugador 1
Primera Carta
Fila primera carta: 4
Columna primera carta: 4

Guardando en archivo

0      0      1      2      3      4      5      6      7
*      *      *      *      *      *      *      *      *
1      *      *      *      *      *      *      *      *
2      *      *      *      *      *      *      *      *
3      *      *      *      *      *      *      *      *
4      *      *      *      *      8/7      *      *      *
5      *      *      *      *      *      *      *      *
6      *      *      *      *      *      *      *      *
7      *      *      *      *      *      *      *      *

Jugador 1: 0      Jugador 2: 0

    Turno: Jugador 1
Segunda Carta
Fila segunda carta: 0
Columna segunda carta: 0

Guardando en archivo

0      0      1      2      3      4      5      6      7
49/70  *      *      *      *      *      *      *      *
1      *      *      *      *      *      *      *      *
2      *      *      *      *      *      *      *      *
3      *      *      *      *      *      *      *      *
4      *      *      *      *      8/7      *      *      *
5      *      *      *      *      *      *      *      *
6      *      *      *      *      *      *      *      *
7      *      *      *      *      *      *      *      *

Jugador 1: 0      Jugador 2: 0

    Turno: Jugador 1
No hubo par, es el turno del jugador 2
¿Desea seguir jugando? (Escriba fin para terminar el juego)

```

Si deseas continuar escribes todo menos fin y si no se tuvo un par es el turno del otro jugador y las cartas vuelven as a darse la vuelta y el puntaje sigue igual:

```

¿Desea seguir jugando? (Escriba fin para terminar el juego)si

Guardando en archivo

0      0      1      2      3      4      5      6      7
*      *      *      *      *      *      *      *      *
1      *      *      *      *      *      *      *      *
2      *      *      *      *      *      *      *      *
3      *      *      *      *      *      *      *      *
4      *      *      *      *      *      *      *      *
5      *      *      *      *      *      *      *      *
6      *      *      *      *      *      *      *      *
7      *      *      *      *      *      *      *      *

Jugador 1: 0      Jugador 2: 0

    Turno: Jugador 2

Primera Carta
Fila primera carta: 

```

Si el jugador desea dar la vuelta a la misma carta durante el mismo juego y en la segunda carta le desea dar la vuelta a la misma sale un mensaje que ya está volteada esa carta:

```
Guardando en archivo

0      0      1      2      3      4      5      6      7
0      *      *      *      *      *      *      *      *
1      *      *      *      *      *      *      *      *
2      *      *      *      *      *      *      *      *
3      *      *      *      *      *      *      *      *
4      *      *      *      *      *      *      *      *
5      *      *      *      *      *      *      *      *
6      *      *      *      *      *      *      *      *
7      *      *      *      *      *      *      *      *

Jugador 1: 0      Jugador 2: 0

    Turno: Jugador 2

Primera Carta
Fila primera carta: 1
Columna primera carta: 1

Guardando en archivo

0      0      1      2      3      4      5      6      7
0      *      *      *      *      *      *      *      *
1      *      7/10  *      *      *      *      *      *
2      *      *      *      *      *      *      *      *
3      *      *      *      *      *      *      *      *
4      *      *      *      *      *      *      *      *
5      *      *      *      *      *      *      *      *
6      *      *      *      *      *      *      *      *
7      *      *      *      *      *      *      *      *

Jugador 1: 0      Jugador 2: 0

    Turno: Jugador 2
Segunda Carta
Fila segunda carta: 1
Columna segunda carta: 1

Carta ya volteada
Segunda Carta
Fila segunda carta: 1
```

Si no es par sale un mensaje y pregunta si desea seguir jugando:

```
Guardando en archivo

0      0      1      2      3      4      5      6      7
0      *      *      *      *      *      *      *      *
1      *      *      *      *      *      *      *      *
2      *      *      *      12/6  *      *      *      *
3      *      *      *      *      *      *      *      *
4      *      *      *      *      *      48/16  *      *
5      *      *      *      *      *      *      *      *
6      *      *      *      *      *      *      *      *
7      *      *      *      *      *      *      *      *

Jugador 1: 0      Jugador 2: 0

    Turno: Jugador 2
No hubo par, es el turno del jugador 1
¿Desea seguir jugando? (Escriba fin para terminar el juego)
```

Pero si es par sale un mensaje que es par y que el jugador que hizo el par sigue siendo su turno y también los pares se quedan abiertos:

```

0      0      1      2      3      4      5      6      7
0      *      *      *      *      *      *      *      *
1      *      *      *      *      *      *      *      *
2      *      *      *      *      *      *      *      *
3      *      *      24/27  *      *      *      *      *
4      *      *      *      *      *      *      *      *
5      *      *      *      *      8/9  *      *      *
6      *      *      *      *      *      *      *      *
7      *      *      *      *      *      *      *      *

Jugador 1: 0      Jugador 2: 0

    Turno: Jugador 1
Es un par, ¡felicitaciones!!
Sigue siendo el turno del jugador 1
¿Desea seguir jugando? (Escriba fin para terminar el juego)

Guardando en archivo

0      0      1      2      3      4      5      6      7
0      *      *      *      *      *      *      *      *
1      *      *      *      *      *      *      *      *
2      *      *      *      *      *      *      *      *
3      *      *      24/27  *      *      *      *      *
4      *      *      *      *      *      *      *      *
5      *      *      *      *      8/9  *      *      *
6      *      *      *      *      *      *      *      *
7      *      *      *      *      *      *      *      *

Jugador 1: 1      Jugador 2: 0

    Turno: Jugador 1

Primera Carta
Fila primera carta: 5
Columna primera carta: 2

```

Si el jugador en cualquier punto de su turno intenta voltear una carta ya volteada sale un mensaje que esa carta ya fue volteada:

```

¿Desea seguir jugando? (Escriba fin para terminar el juego)

Guardando en archivo

0      0      1      2      3      4      5      6      7
0      *      *      *      *      *      *      *      *
1      *      *      *      *      *      *      *      *
2      *      *      *      *      *      *      *      *
3      *      *      24/27  *      *      *      *      *
4      *      *      *      *      *      *      *      *
5      *      *      *      *      8/9  *      *      *
6      *      *      *      *      *      *      *      *
7      *      *      *      *      *      *      *      *

Jugador 1: 1      Jugador 2: 0

    Turno: Jugador 1

Primera Carta
Fila primera carta: 5
Columna primera carta: 2

Guardando en archivo

0      0      1      2      3      4      5      6      7
0      *      *      *      *      *      *      *      *
1      *      *      *      *      *      *      *      *
2      *      *      *      *      *      *      *      *
3      *      *      24/27  *      *      *      *      *
4      *      *      *      *      *      *      *      *
5      *      *      *      *      8/9  *      *      *
6      *      *      18/24  *      *      *      *      *
7      *      *      *      *      *      *      *      *

Jugador 1: 1      Jugador 2: 0

    Turno: Jugador 1
Segunda Carta
Fila segunda carta: 3
Columna segunda carta: 2

Carta ya volteada
Segunda Carta
Fila segunda carta:

```

En caso que el jugador desee salir de la partida al escribir ‘fin’, sale otra pregunta para guardar la partida hasta ese turno. Si escribe si, en ese caso la partida fue guardada y regresa al menú principal.

```

0      0      1      2      3      4      5      6      7
0      *      *      *      *      *      *      *      *
1      *      *      *      *      *      *      *      *
2      *      *      *      *      *      *      *      *
3      *      *      *      *      70/63      *      *      *
4      *      *      *      *      *      *      *      *
5      *      *      *      *      *      *      *      *
6      *      *      *      *      *      *      *      *
7      *      *      *      *      *      *      *      *

Jugador 1: 0      Jugador 2: 0

Turno: Jugador 1
Segunda Carta
Fila segunda carta: 6
Columna segunda carta: 4

Guardando en archivo

0      0      1      2      3      4      5      6      7
0      *      *      *      *      *      *      *      *
1      *      *      *      *      *      *      *      *
2      *      *      *      *      *      *      *      *
3      *      *      *      *      70/63      *      *      *
4      *      *      *      *      *      *      *      *
5      *      *      *      *      *      *      *      *
6      *      *      *      *      2/3      *      *      *
7      *      *      *      *      *      *      *      *

Jugador 1: 0      Jugador 2: 0

Turno: Jugador 1
No hubo par, es el turno del jugador 2
¿Desea seguir jugando? (Escriba fin para terminar el juego? fin
¿Desea guardar partida? si

Guardando en archivo

MENU
1)Empezar partida
2)Continuar una partida guardada
3)Salir

Digita la opcion: 

```

Si el jugador mas tarde quiere jugar desde el punto guardado, digita la opcion 2:

```

MENU
1)Empezar partida
2)Continuar una partida guardada
3)Salir

Digita la opcion: 2
Leyendo contenido de archivo ...

Guardando en archivo

0      0      1      2      3      4      5      6      7
0      *      *      *      *      *      *      *      *
1      *      *      *      *      *      *      *      *
2      *      *      *      *      *      *      *      *
3      *      *      *      *      *      *      *      *
4      *      *      *      *      *      *      *      *
5      *      *      *      *      *      *      *      *
6      *      *      *      *      *      *      *      *
7      *      *      *      *      *      *      *      *

Jugador 1: 0      Jugador 2: 0

Turno: Jugador 2

Primera Carta
Fila primera carta: 

```

Para demostrar los ganadores y los 3 diferentes escenarios cambiamos el código a permitir columnas y filas de mínimo 2 para poder ejemplificar lo que pasaría cuando todas las cartas se voltean:

Caso 1 gana el jugador 2 :

```
Turno: Jugador 2
Segunda Carta
Fila segunda carta: 1
Columna segunda carta: 1
```

Guardando en archivo

	0	1
0	8/5	16/16
1	56/35	8/8

Jugador 1: 0 Jugador 2: 1

```
Turno: Jugador 2
Es un par, ¡felicitaciones!!
Sigue siendo el turno del jugador 1
```

```
      ^                               ^
  (____)-----(____)
  |  /  |       |  \  |
  |  /  |       |  \  |
  |____|       |____|
  (____)-----(____)

      Felicidades Jugador 2
```

```
El jugador 2 tuvo 2 puntos
el jugador 1 tuvo 0 puntos
¿Desea seguir jugando? (Escriba fin para terminar el juego? )
```

```

0      0      1
0      2/2    1/1
1      7/6    28/24

Jugador 1: 1          Jugador 2: 0

Turno: Jugador 1
Es un par, ¡felicitaciones!!
Sigue siendo el turno del jugador 1


      ^           ^
(____)------(____)
| / |         |\ |
| / |         \ \|
|___|         ___|
(____)------(____)

Felicidades Jugador 1


El jugador 1 tuvo 2 puntos
el jugador 2 tuvo 0 puntos
¿Desea seguir jugando? (Escriba fin para terminar el juego?)
```

```
Guardando en archivo

      0      1      2      3
0      7/8      24/32      3/4      36/4
1      9/1      28/32      50/20      10/4

Jugador 1: 2      Jugador 2: 1

Turno: Jugador 2
Es un par, ¡felicitaciones!!
Sigue siendo el turno del jugador 1
¡Termino el Juego es un empate!
El jugador 1 y 2 tuvieron 2 puntos
EMPATE
```


VI. Conclusiones

Thomas Freund: Durante este proyecto integrador tuvimos la oportunidad de aprender aún más sobre programación. Por un lado pudimos crear un videojuego desde cero donde estaba muy bien definido los parámetros y limitaciones que se deben tener para que se pueda completar el memorama. Frente a la ejecución de las necesidades pudimos como grupo completar todos los escenarios dados como se ilustra en los casos de uso.

Por medio de este proyecto integrador los conceptos aprendidos durante el curso realmente fueron implementados y se demostró un entendimiento de la materia por medio de la ejecución. Personalmente trabajar con funciones anidadas y externas junto con muchas variables y datos estructurados fue un desafío nuevo que presentó muchas dificultades pero terminó siendo una experiencia enriquecedora.

Frente al objetivo del proyecto de crear un memorama que pueda ayudar a estudiantes de sexto grado, también creo que se cumplió. Ya que el tablero puede tener cualquier dimensión par, donde las filas y columnas sean mayor o igual a 8 donde se utilizan varios aspectos del cerebro utilizando la memorización, la concentración, y un entendimiento intrínseco de la sub competencia de equivalencia de fracciones de una manera competitiva y divertida.

Carlos Moreno

A lo largo de este proyecto fue necesario aplicar bastantes conocimientos para la creación de nuestro juego. Considero que tuve una gran experiencia de aprendizaje de programación durante la creación del código de nuestro memorama. Fue un gran trabajo en equipo en donde mi compañero y yo tuvimos que coordinarnos para lograr nuestros objetivos en cuanto a nuestro programa, fue un trabajo largo pero al final logramos hacer que el código funcionará por completo.

El proyecto integrador logró que integrará todos mis conocimientos vistos en la materia de pensamiento computacional, además de desde mi persona seguir investigando para dominar más aquellas competencias que eran necesarias para programar dicho memorama de manera exitosa.

El propósito principal del juego se cumplió, el cual era la creación de un memorama didactico en donde los estudiantes de sexto grado pudieran practicar fracciones así dominar sus conocimientos en este tema de las matemáticas.

Me quedo con la satisfacción de haber aprendido lo básico de programación de Python y espero en un futuro seguir aprendiendo más y así dominar más este lenguaje de programación.

VII. Bibliografía

Pramanick, S. (mayo 6 del 2019). History of Python. Recuperado el 10 de octubre del 2020, de <https://www.geeksforgeeks.org/history-of-python/>

Pumfrey, L. (2011, febrero). History of Fractions. NRICH. <https://nrich.maths.org/2515>

Soloaga, Ana.(2019.) “Python, Los 5 Usos Más Importantes De Este Lenguaje De Programación.” *El Blog De Akademus,,* www.akademus.es/blog/programacion/principales-usos-python/.

Sweigart, A. (2019). Automate the Boring Stuff with Python, 2nd Edition: Practical Programming for Total Beginners (English Edition) (2.a ed.). No Starch Press. <https://automatetheboringstuff.com/>

Unmalnick.(2018). “La Historia De Python.” *Platzi*, Platzi.
platzi.com/blog/historia-python/.