**PROYECTO DE OPERACIONES CON MATRICES: MATRIZMATE**

**INTEGRANTES DEL PROYECTO:**

**SAMUEL EDUARDO FAJARDO QUINTERO**

**MANUEL FELIPE TORRES GAMBOA**

**SANTIAGO GAMBOA MARTÍNEZ**

**UNIVERSIDAD NACIONAL DE COLOMBIA**

**SEDE ORINOQUÍA**

**2024**

**Nombre del Proyecto: MatrizMate**

En el mundo de las ciencias computacionales y matemáticas, el concepto de matrices desempeña un papel fundamental. Desde la resolución de sistemas de ecuaciones hasta la optimización de algoritmos, las matrices son una herramienta poderosa. En este proyecto, buscamos crear un programa en Python que nos permita realizar diversas operaciones con matrices, todo ello según la preferencia del usuario.

**Objetivo del Proyecto**

El objetivo principal de este proyecto es desarrollar un programa interactivo que pueda manipular matrices de acuerdo con las necesidades y preferencias del usuario. Esto incluirá la capacidad de realizar operaciones básicas como la suma, resta, multiplicación y división de matrices, así como también funcionalidades más avanzadas como la transposición, el cálculo de determinantes y la resolución de sistemas lineales.

**Herramientas Utilizadas**

Para la implementación de este proyecto, nos valdremos del lenguaje de programación Python, más específico Python 3.9, debido a la compatibilidad de versiones con las librerías. Además, haremos uso de la biblioteca NumPy, la cual proporciona un conjunto de funciones y métodos optimizados para el manejo de matrices y operaciones numéricas en Python. También haremos uso de la biblioteca gráfica PyQt5, la cual nos permitirá desarrollar una interfaz gráfica de manera sencilla y rápida.

**Estructura del Proyecto**

El proyecto estará organizado en diferentes fases, comenzando por una introducción a las operaciones básicas con matrices y la implementación de estas operaciones en Python. Luego, nos adentraremos en la interacción con el usuario, creando un programa que permita ingresar matrices y realizar operaciones con ellas de manera más interactiva. Finalmente, exploraremos funcionalidades más avanzadas y proporcionaremos ejemplos de uso y pruebas para demostrar la eficacia y funcionalidad de nuestro programa.

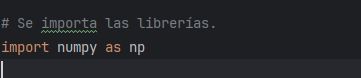
Con este proyecto, no solo buscaremos fortalecer nuestros conocimientos en programación y manipulación de matrices, sino también brindar una herramienta útil y educativa para aquellos interesados en el fascinante mundo de las matemáticas computacionales.

**Fase I: Funcionalidades Básicas del Programa**

En esta primera fase, nos enfocaremos en desarrollar las funcionalidades esenciales de nuestro programa para operar con matrices. Estas funcionalidades incluirán una interacción básica con el usuario, donde este podrá especificar el número de filas y columnas de las matrices que desea utilizar. Además, implementaremos las operaciones fundamentales de suma, resta, multiplicación y división de matrices.

1. **Importación de librerías**

Para comenzar con nuestro proyecto, necesitaremos importar la biblioteca NumPy, una herramienta fundamental que nos permitirá realizar la creación y manipulación de matrices de manera eficiente.

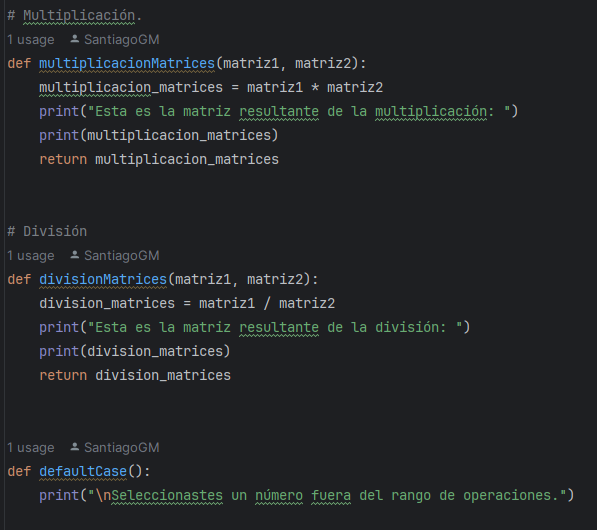


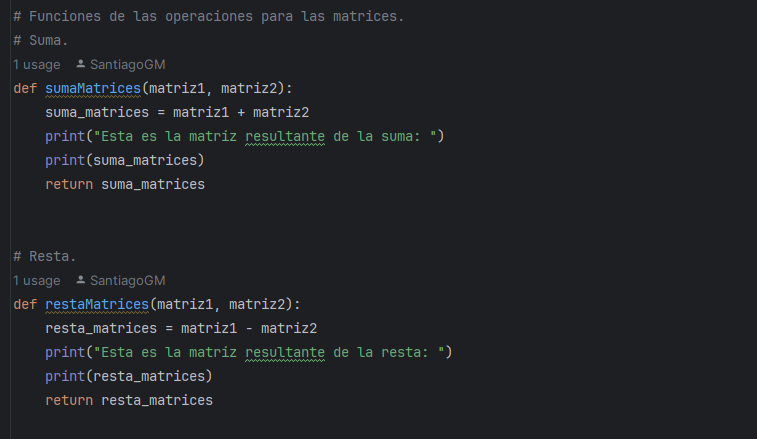
La librería NumPy nos proporciona una amplia gama de funciones y métodos optimizados para trabajar con matrices y operaciones numéricas en Python. Al importarla con el alias **np**, simplificamos su uso en nuestro código y seguimos una convención comúnmente aceptada en la comunidad de programadores Python.

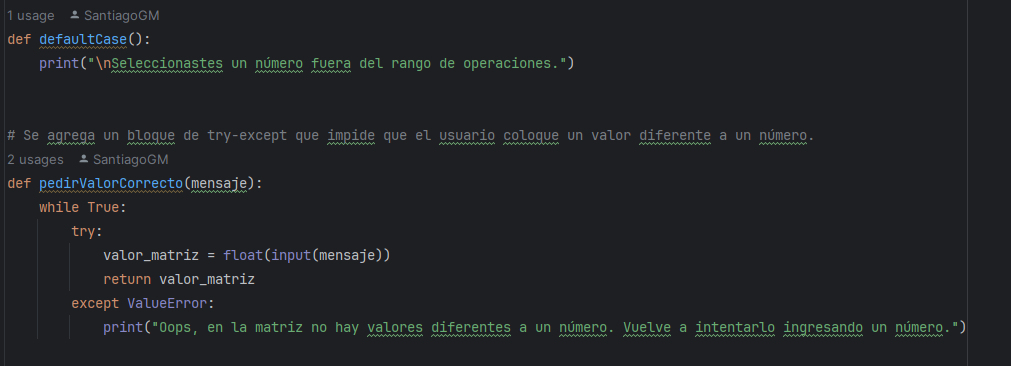
Con la librería NumPy a nuestra disposición, estamos listos para comenzar a desarrollar las funcionalidades básicas de nuestro programa para operar con matrices.

1. **Definición de funciones de Operaciones Básicas con Matrices, Default para seleccionador de Operaciones y Manejo de errores**

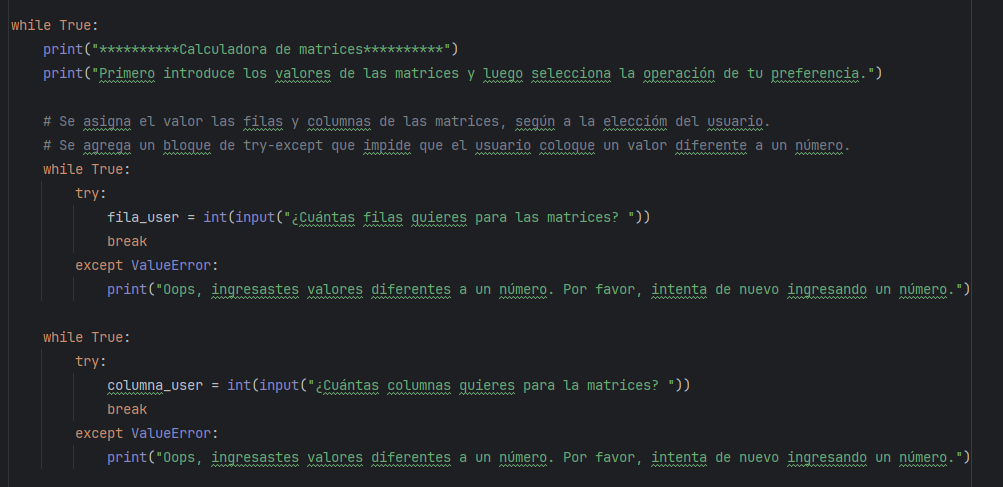
Una vez que tengamos las matrices definidas, implementaremos las operaciones básicas de suma, resta, multiplicación y división. Utilizaremos la biblioteca NumPy para llevar a cabo estas operaciones de manera eficiente y precisa. También para brindar una mejor experiencia al usuario, se agrega una función default que se llama **defaultCase** que se utilizará en la Sección 4 en el caso en que el usuario seleccione una opción que no se encuentre en el rango de operaciones. Por último, se crea una función llamada **pedirValorCorrecto** que permitirá manejar los errores de tipos de valores en la entrada de la definición de valores de las matrices, en otras palabras, el programa se asegurará de que el usuario ingreso un valor de tipo numérico; lo anterior se utilizará en la Sección 3.2.





****

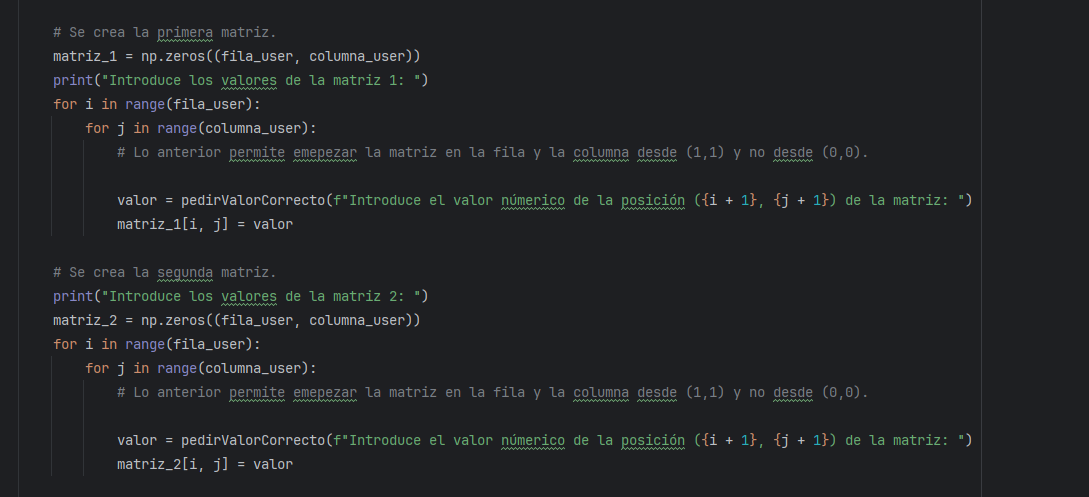
1. **Interacción con el Usuario**
   1. **Asignación de las dimensiones de las matrices**

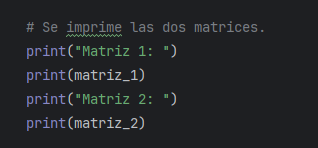
Permitiremos que el usuario defina las dimensiones de las matrices con las que desea trabajar. Esto incluirá solicitar al usuario el número de filas y columnas para cada matriz. Utilizaremos la función **input()** para recopilar esta información del usuario. También cada **input()** tendrá su propia estructura de manejo de errores (Try-Except), la cuál asegurará que el usuario ingrese un valor entero.

* 1. **Asignación de los valores de las matrices**

Se define los valores de las matrices y se utiliza la función de manejo de errores creada anteriormente. Para esto haremos uso en cada matriz dos **for anidados** en el que se encuentra un

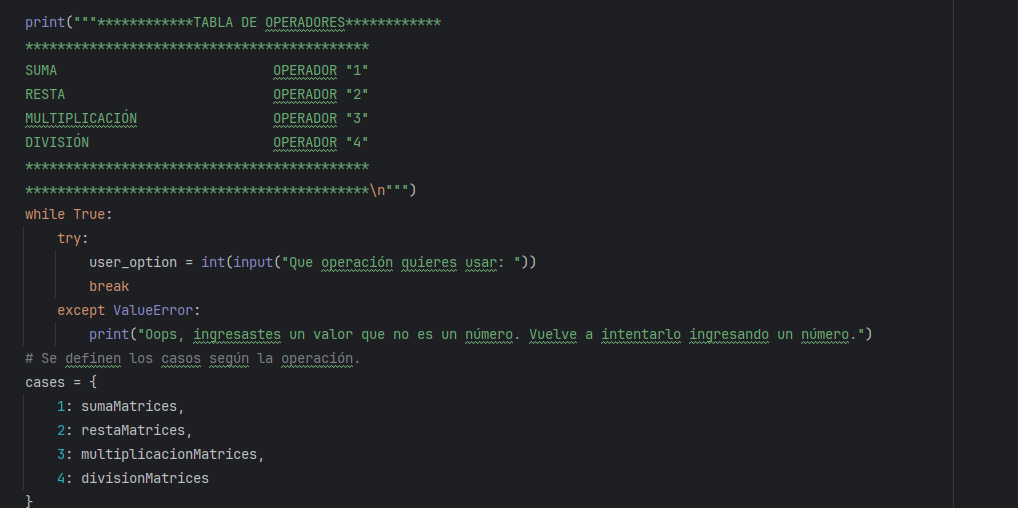
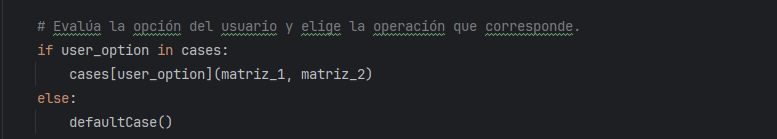
**input()**, lo cual se usa para que el usuario ingrese los valores de cada posición de las matrices. Luego de esto se imprimen las dos matrices.



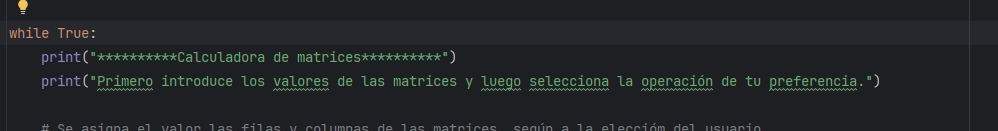


1. **Selección de Operaciones**

Para brindar al usuario una experiencia fluida, creamos una estructura de control que se conforma de un diccionario, en la cual se definió el número asignado a cada operación, luego con la función **input()** y la función **if-else** le permitirán al usuario seleccionar la operación que desea realizar, y si este ingresa una opción fuera del rango se utiliza la función **defaultCase** que se creó anteriormente en la sección 2, el cual imprime un mensaje de error.



* **Importante:**

Para dar una mejor experiencia al usuario se creó un control de flujo con un **while** el cuál funciona si el usuario quiere volver a utilizar la calculadora.

