

TRANSFORMACIONES DE MATRICES

Introducción:

Este código implementa una aplicación web interactiva utilizando la biblioteca streamlit, que nos permite a nosotros los usuarios cargar una matriz desde un archivo de texto y realizar diferentes transformaciones sobre ella, como obtener la diagonal principal, las partes triangulares superior e inferior, y la transpuesta. Las matrices transformadas se muestran en la interfaz de usuario y también se guardan en archivos de texto separados.

Código en Python:

```
import streamlit as st
import pandas as pd

class Matriz:
    def __init__(self, datos):
        self.__datos = pd.DataFrame(datos)

    def save_and_show(self, data, title, filename):
        st.write(title)
        st.write(data)
        data.to_csv(f'{filename}.txt', index=False, sep='\t')

    def show_matriz(self):
        self.save_and_show(self.__datos, "Matriz Original:", 'matriz_original')

    def diagonal(self):
        diag = pd.DataFrame([[self.__datos.iat[i, j]
                               if i == j else 0 for j in range(len(self.__datos.columns))
                               for i in range(len(self.__datos))])
        self.save_and_show(diag, "Matriz Diagonal:", 'matriz_diagonal')

    def triangular_superior(self):
        tri_sup = pd.DataFrame([[self.__datos.iat[i, j]
                                 if i <= j else 0 for j in range(len(self.__datos.columns))
                                 for i in range(len(self.__datos))])
        self.save_and_show(tri_sup, "Matriz Triangular Superior:", 'matriz_triangular_superior')

    def triangular_inferior(self):
        tri_inf = pd.DataFrame([[self.__datos.iat[i, j]
                                 if i >= j else 0 for j in range(len(self.__datos.columns))
                                 for i in range(len(self.__datos))])
        self.save_and_show(tri_inf, "Matriz Triangular Inferior:", 'matriz_triangular_inferior')

    def transpuesta(self):
        transp = self.__datos.transpose()
        self.save_and_show(transp, "Matriz Transpuesta:", 'matriz_transpuesta')

def main():
    st.title("Transformaciones de Matrices")
```

```
uploaded_file = st.file_uploader("Cargar archivo de texto", type=["txt"])
if uploaded_file is not None:
    content = uploaded_file.read().decode("utf-8")
    data = [list(map(int, row.split())) for row in content.strip().split('\n')]
    m = Matriz(data)
    buttons = {
        "Mostrar Matriz Original": m.show_matriz,
        "Mostrar Diagonal": m.diagonal,
        "Mostrar Triangular Superior": m.triangular_superior,
        "Mostrar Triangular Inferior": m.triangular_inferior,
        "Mostrar Transpuesta": m.transpuesta
    }
    for label, action in buttons.items():
        if st.button(label):
            action()

if __name__ == "__main__":
    main()
```

Salida:



Mostrar Matriz Original

Matriz Original:

	0	1	2
0	8	1	3
1	5	4	9
2	2	6	7

Mostrar Diagonal

Matriz Diagonal:

	0	1	2
0	8	0	0
1	0	4	0
2	0	0	7

Mostrar Triangular Superior

Matriz Triangular Superior:

	0	1	2
0	8	1	3
1	0	4	9
2	0	0	7

Mostrar Triangular Inferior

Matriz Triangular Inferior:

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

Mostrar Transpuesta

Matriz Transpuesta:

	0	1	2
0	8	5	2
1	1	4	6
2	3	9	7

Link del Repositorio y Código QR

https://github.com/Lucc4z/Programming-Language-II/blob/main/Python/Matriz_pd.py



Explicación del código:

```
import streamlit as st
import pandas as pd
```

Aquí estamos importando 2 librerías: **streamlit** para crear la interfaz de usuario y **pandas** para trabajar con datos en formato de DataFrame.

```
class Matriz:
    def __init__(self, datos):
        self.__datos = pd.DataFrame(datos)
```

Definimos una clase llamada **Matriz**. El método `__init__` es el constructor de la clase y recibe los **datos** de la matriz como argumento. Estos datos se convierten en un DataFrame de Pandas y se almacenan en el atributo `__datos`.

```
    def save_and_show(self, data, title, filename):
        st.write(title)
        st.write(data)
        data.to_csv(f'{filename}.txt', index=False, sep='\t')
```

Luego este método recibe tres argumentos: **data** (la matriz a mostrar y guardar), **title** (el título a mostrar en la interfaz) y **filename** (el nombre de archivo donde se guardará la matriz). Dentro del método, se muestra el título y la matriz utilizando `st.write`, y luego se guarda la matriz en un archivo de texto utilizando el método `to_csv` de Pandas.

```
    def show_matriz(self):
        self.save_and_show(self.__datos, "Matriz Original:", 'matriz_original')
```

Este método simplemente llama al método `save_and_show` con la matriz original (`self.__datos`), el título "Matriz Original:" y el nombre de archivo "matriz_{original}".

```
    def diagonal(self):
        diag = pd.DataFrame([[self.__datos.iat[i, j]
                               if i == j else 0 for j in range(len(self.__datos.columns))]
                              for i in range(len(self.__datos))])
        self.save_and_show(diag, "Matriz Diagonal:", 'matriz_diagonal')
```

Este método crea una nueva matriz **diag** que contiene los elementos de la diagonal (`self.__datos`). Luego estamos utilizando unas listas anidadas para construir la matriz: si los índices *i* y *j* son iguales, se toma el elemento correspondiente de la matriz original; sino, se asigna un 0. Luego, se llama al método `save_and_show` con la matriz **diag**, el título "Matriz Diagonal:" y el nombre de archivo "matriz_{diagonal}".

```
    def triangular_superior(self):
        tri_sup = pd.DataFrame([[self.__datos.iat[i, j]
                                 if i <= j else 0 for j in range(len(self.__datos.columns))]
                                for i in range(len(self.__datos))])
        self.save_and_show(tri_sup, "Matriz Triangular Superior:", 'matriz_triangular_superior')
```

Este método es similar al `diagonal`, pero crea una matriz **tri_sup** que contiene los elementos de la parte triangular superior de la matriz original (`self.__datos`). Si el índice *i* es menor o igual que *j*, se toma el elemento correspondiente de la matriz original; sino, se asigna un 0. Después llamamos al método `save_and_show` con la matriz **tri_sup**, el título "Matriz Triangular Superior:" y el nombre de archivo "matriz_{triangular_superior}".

```
def triangular_inferior(self):
    tri_inf = pd.DataFrame([[self.__datos.iat[i, j]
        if i >= j else 0 for j in range(len(self.__datos.columns))]
        for i in range(len(self.__datos))])
    self.save_and_show(tri_inf, "Matriz Triangular Inferior:", 'matriz_triangular_inferior')
```

Este método es similar al `triangular_superior`, pero crea una matriz `tri_inf` que contiene los elementos de la parte triangular inferior de la matriz original (`self.__datos`). Si el índice `i` es mayor o igual que `j`, se toma el elemento correspondiente de la matriz original; sino, se asigna un 0. Después, llamamos al método `save_and_show` con la matriz `tri_inf`, el título "Matriz Triangular Inferior:" y el nombre de archivo "matriz_{triangular_inferior}".

```
def transpuesta(self):
    transp = self.__datos.transpose()
    self.save_and_show(transp, "Matriz Transpuesta:", 'matriz_transpuesta')
```

En este método se crea una nueva matriz `transp` que es la transpuesta de la matriz original (`self.__datos`). Se utiliza el método `transpose` de Pandas para calcular la transpuesta. Después llamamos al método `save_and_show` con la matriz `transp`, el título "Matriz Transpuesta:" y el nombre de archivo "matriz_{transpuesta}".

```
def main():
    st.title("Transformaciones de Matrices")
    uploaded_file = st.file_uploader("Cargar archivo de texto", type=["txt"])
    if uploaded_file is not None:
        content = uploaded_file.read().decode("utf-8")
        data = [list(map(int, row.split())) for row in content.strip().split('\n')]
        m = Matriz(data)
        buttons = {
            "Mostrar Matriz Original": m.show_matriz,
            "Mostrar Diagonal": m.diagonal,
            "Mostrar Triangular Superior": m.triangular_superior,
            "Mostrar Triangular Inferior": m.triangular_inferior,
            "Mostrar Transpuesta": m.transpuesta
        }
        for label, action in buttons.items():
            if st.button(label):
                action()
```

Y pues esta parte es la función principal del programa.