F210 Programacion Aplicada a Finanzas
Matrices

1. Se le pide construir una clase *myarray* para operar con matrices usando herramientas **exclusivamente de Python base** que tenga las siguientes características:

   - Debe contener **exclusivamente** los siguientes atributos de instancia de clase:

     (a) una lista conteniendo los elementos de la matriz (llamelo *elems*),

     (b) un indicador de la cantidad de filas (r)

     (c) un indicador de la cantidad de columnas (c)

     (d) un indicador (*by_ row,* de tipo boolean) para indicar la correspondencia entre los elementos de la lista y los elementos de la matriz (saber si al movernos por la lista estamos atravesando la matriz por filas – *by_ row = True* – o por columnas – *by_ row=False* –)

     NOTA: la estructura de datos no debe ser de lista de listas, sino que debe ser una lista de elementos. Cada elemento m_i corresponde a un elemento de la matriz.

   - Debe tener al menos los siguientes métodos:

     – Creador de instancia de clase *_ _ init_ _ (self,lista,r,c,by_ row)* , con la lista de elementos, las dimensiones de la matriz y el orden de navegacion. Si *by_ row* = True, entonces se interpretará que los elementos se encuentran almacenados fila por fila, caso contrario, se interpretará que están almacenados por columna.

     – funciones *get_ pos(self,j,k)* y *get_ coords(self,m).* La primera toma las coordenadas *j,k* en la matriz del elemento $x_{j,k}$ y devuelve la posicion i asociada al elemento m_i en la lista *elems*. La segunda, toma una posición i en la lista y devuelve en forma de tupla las coordenadas *j,k* correspondientes en la matriz.

     – Una función *switch(self)* que modifique el objeto y devuelva la misma matriz, pero alterando la lista *elems* y cambiando el valor de verdad de *by_ row.* Verifique que (A.switch()).switch() = A. Notar que la cantidad de filas y columnas no cambian porque la matriz es la misma. Solamente cambia la forma de almacenamiento.

     – Funciones $get\_row(self, j)$, $get\_col(self, k)$ y *get_ elem(self,(j,k))* que devuelvan el contenido de la columna *j*, de la columna k y el elemento *(j,k)* de la matriz respectivamente. Construya dos variantes de cada una. La primera que sea explícita, buscando los elementos posicionalmente. La segunda que sea valiendose de la multiplicacion a izquierda o a derecha por un vector (se valdrá de una funcion *prod* que debe construir más abajo).

     – Una función $get\_submatrix(self, \mathrm{row}\, list, \mathrm{col}\, list)$ que reciba una lista de filas y columnas y devuelvan la submatriz correspondiente

     – funciones *del_ row(*$self, j$*), del_ col(self,k)* que devuelvan un objeto de la clase habiendo eliminando la fila j o la columna k. Debe presentar dos versiones de estas funciones siguiendo la lógica expuesta en $get\_row(self, j)$ y $get\_col(self, k)$

- funciones *swap_rows(self,j,k)* y *swap_cols(self,l,m)* que devuelven un objeto de la clase con las filas (j,k) y columnas (l,m) intercambiadas. Aqui se le piden 2 versiones: una explicita barriendo los elementos, y otro valiendose de la pre o postmultiplicación por un operador de permutación $P_{j,k}$ que definiremos en clase.
- funciones *scale_row(self,j,x), scale_col(self,k,y)* que tomen el objeto y devuelvan otro del mismo tipo, pero con la fila j multiplicada por el factor x en el primer caso y con la columna k multiplicada por y en el segundo. Nuevamente, construya dos versiones: la explicita y otra pre o postmultiplicando por una matriz de escala S_j o S_k que definiremos en clase.
- Operador *transpose(self)* que devuelve un elemento de la clase, pero con la matriz transpuesta. Verifique que (A.transpose()).transpose() = A.
- Operador *flip_cols(self)* y *flip_rows(self)* que devuelven una copia del elemento de la clase, pero reflejado especularmente en sus columnas o filas.
- funcion *det(self)* que devuelva el determinante de la matriz, si es cuadrada. La función determinante se computa recursivamente de la forma

  $$\det(M) = \sum_{j=1}^{N} m_{i,j} \left(-1\right)^{i+j} \det\left(M_{-i,-j}\right)$$

  donde M es una matriz de NxN, i es una fila arbitraria de M que elegimos para computar el determinante y $M_{-i,-j}$ es la submatriz de M que resulta de eliminar la fila i y la columna j. La recursión termina cuando $M_{-i,-j}$ es una matriz de 1x1 en cuyo caso $det(x) = x$.
- funciones *add(self,B)*, *sub(self,B)*, *rprod(self,B)*, *lprod(self,B)* y *pow(self,n)* que efectuen operaciones de suma, resta, producto a derecha y a izquierda del objeto matriz por otro de su clase **o por un escalar (debe evaluar primero el type de B y operar segun el caso),** y potenciación por un escalar.