Universidad de Concepción
Facultad de Ingeniería
Departamento de Ingeniería Industrial

Programación Aplicada a la Ingeniería Industrial

Tarea 1

Profesor: Carlos Contreras Bolton **Fecha:** 20 de abril de 2023

1. Entrega

Lunes 1 de mayo de 2023 hasta las 23:59 hrs en Canvas.

2. Objetivos de la tarea

a) Desarrollar su capacidad para llevar a cabo un programa de pequeña escala en el lenguaje de programación Python.

b) Aplicando de matrices y cadenas de caracteres.

c) Aplicando el paradigma de orientación a objetos.

d) Aplicando control de excepciones.

3. Enunciado

Desarrollar un mini programa de matlab/octave en Python que se base en el paradigma de orientación a objetos. El programa debe facilitar la manipulación de las matrices, y debe ser capaz de trabajar con matrices de cualquier tamaño que contengan números y/o términos algebraicos.

Entre las operaciones que se pueden utilizar en el programa se encuentran la creación de matrices a partir de un archivo, la suma, resta y multiplicación de matrices, la multiplicación escalar de matrices, la transpuesta, la inversa y la determinante de una matriz. Además, la ordenación de los elementos de una matriz y la búsqueda de elementos en la misma.

Para garantizar una experiencia de usuario satisfactoria, el programa debe manejar excepciones en caso de que se introduzcan datos no válidos o se intenten realizar operaciones con matrices de tamaños incompatibles o valores que no sean numéricos o algebraicos. Además, el programa debe implementar la sobrecarga de operadores para simplificar la sintaxis (en el código) y permitir al usuario utilizar la notación matemática tradicional. Por ejemplo, el usuario puede sumar dos matrices utilizando la expresión $A + B$, restarlas con $A - B$, multiplicarlas con $A * B$ y calcular la transpuesta con $A **T$, calcular la matriz inversa con $A **-1$, determinante de una matriz con `det` A, ordenar una matriz con `sort` A, y buscar en una matriz con b `in` A.

El programa también debe permitir guardar el resultado (solo cuando el resultado es una matriz) de una operación en una nueva matriz. Por ejemplo, si se realiza la operación $C = A + B$, el resultado es guardado en la matriz C, y luego se podría realizar la operación $2 * C$. Además, el usuario tiene la opción de sobrescribir matrices, utilizando cualquiera de las operaciones permitidas. Por ejemplo, se puede realizar la operación $A = A **T$ o $A = C - B$. Se incluye una operación para eliminar matrices (`clear` A), lo que permite que el usuario pueda eliminar cualquier matriz cargada o nueva en el programa. Además, se cuenta con una operación para mostrar una o todas las matrices (`show`). Al utilizar `show` sin

una matriz, se deben mostrar todas las matrices cargadas por pantalla, mientras que al utilizar `show` A, se debe mostrar solo la matriz A por pantalla.

Es importante destacar que el programa no soporta matrices vacías y antes de calcular/guardar el resultado de una operación en una matriz, el programa verifica que las dimensiones de las matrices sean compatibles y que se puedan realizar las operaciones correspondientes. Si no se cumple con estas condiciones, se debe informar al usuario que la operación no es posible. Además, para simplificar el programa, las operaciones solo permiten hasta una sola operación como elemento, es decir, se podría realizar una operación $A + B$, donde A contuviera elementos hasta con una sola operación, por ejemplo, "$2+a$" o "a". Pero, no estaría permitido si es "$2+a*d$" o "abc". Sin embargo, el resultado de $C = A+B$, sí podría obtener elementos con más de dos operaciones (luego no se podría ocupar para una futura operación).

Notar que cuando se utiliza la operación de ordenación, se debe seguir el formato fila 1, fila 2, y así sucesivamente. Como las matrices pueden tener número y/o términos algebraicos, entonces, las funciones de ordenamiento deben comparar el primer carácter (en caso de empate, seguir con el segundo, y así sucesivamente) de los elementos en orden ascendente (de menor a mayor) según su valor y se utiliza el valor `Unicode` de los caracteres para los términos algebraicos. Por ejemplo, si se ordena una secuencia que contiene los siguientes elementos: a, 3, b+2, 2, c, 1+a, el orden debe ser: 1+a, 2, 3, a, b+2, c, es decir, primero se ordenan los números de menor a mayor, y luego se ordenan los términos algebraicos en orden alfabético.

4. Entrada y Salida

Existe un archivo de entrada que se llama "`entrada.in`". Tiene el siguiente formato: n (tamaño de filas), espacio, m (tamaño de columnas), espacio, $n \times m$ números y/o términos algebraicos (siempre separados por un espacio). Todo en una sola línea. Cada línea corresponde a una matriz A, B, C, ..., Z. Notar que también se permiten los operadores: $*$ y $+$, siempre y cuando cumplan la sintaxis de su operación matemática correspondiente, es decir, no podría ocurrir: "$+a$" o "$1 + *a$", etc. Tampoco terminos con más de una operación. Por tanto, debe controlar esas excepciones.

Tabla 1: Ejemplo de entrada y salida de archivos con las matrices.

Entrada	Salida

`3 3 1 2 3 4 5 6 7 8 9`
`3 3 a b c d e f g h i`
`2 2 a b c 2 1 2`
	`    | 1 2 3 |`
	`A = | 4 5 6 |`
	`    | 7 8 9 |`
	
	`    | a b c |`
	`B = | d e f |`
	`    | g h i |`
	
	`    | a b |`
	`C = | c 2 |`
	`    | 1 2 |`

5. Ejemplos

El programa debe permitir al usuario ingresar las operaciones permitidas que desea realizar. Si el usuario ingresa una operación válida, como una suma (como se muestra en la Tabla 2) o una multiplicación de matrices (como se muestra en la Tabla 3), siempre y cuando las propiedades de la operación sean satisfechas, se debe mostrar la salida correspondiente. De lo contrario, se debe informar al usuario que la operación no está permitida. Es importante destacar que para cada operación hay reglas y/o propiedades, como por ejemplo en la multiplicación de matrices, se deben verificar las dimensiones de las matrices

para asegurar que sean compatibles y se puedan multiplicar. Por tanto, se debe verificar para todas las operaciones permitidas. Finalmente, un último ejemplo, en la Tabla 4 se observa el resultado de ordenar la matriz B obtenida en la multiplicación anterior.

Tabla 2: Ejemplo de entrada y salida una suma.

Entrada	Salida

`E = A + B`	`E = | 1+a 2+b 3+c |`
	`| 4+d 5+e 6+f |`
	`| 7+g 8+h 9+i |`

Tabla 3: Ejemplo de entrada y salida una multiplicación de matrices.

Entrada	Salida

`B = A * C`	`B = | a+2c+3 b+10 |`
	`| 4a+5c+6 4b+22 |`
	`| 7a+2c+3 7b+34 |`

Tabla 4: Ejemplo de entrada y salida de ordenar una matriz y mostrar.

Entrada	Salida

`sort B`
`F = sort B`
`show F`
	`B = | 4a+5c+6 4b+22 |`
	`| 7a+2c+3 7b+34 |`
	`| a+2c+3 b+10 |`
	
	`F = | 4a+5c+6 4b+22 |`
	`| 7a+2c+3 7b+34 |`
	`| a+2c+3 b+10 |`
	
	`F = | 4a+5c+6 4b+22 |`
	`| 7a+2c+3 7b+34 |`
	`| a+2c+3 b+10 |`

6. Evaluación

A continuación, se detalla como se evalúan los requerimentos de la tarea.

- Implementar de manera correcta los requerimientos (3,0 ptos).
- Considerar un menú con: 1) Lectura (leer archivo "`entrada.in`"), 2) Operaciones (solo operaciones permitidas), 3) Guardar (guardar todas las matrices cargadas en un archivo "`salida.in`"), 4) Salir (0,5 ptos).
- La implementación debe estar completamente orientada a objetos. (1,5 ptos).
- Realizar control de excepciones (1,0 ptos).
- Bonificación por funcionalidad extra o novedosa (0,5 ptos).

7. Condiciones de la tarea

- La tarea es individual.
- Está estrictamente prohibido utilizar otra estructura de datos diferente (como `numpy` u otras) a las listas de Python para realizar las operaciones que se necesitan.
- Las dudas respecto al trabajo, se realizan de manera personal (vía Teams) o vía correo electrónico.
- El lenguaje a utilizar es Python 3.x de manera estándar. No se aceptarán Bibliotecas/Módulos no vistos en clases para la lógica de la tarea. Pero si se aceptarán otras bibliotecas para poder realizar la bonificación.
- Cree las bibliotecas o utilice las bibliotecas dadas por el profesor.
- El enunciado podría sufrir modificaciones que serán publicadas en Canvas.
- Formato de entrega:
  - El sistema debe ser robusto, se penalizarán las caídas de cualquier tipo.
  - Debe estar bien documentado.
  - La entrega se hace mediante la plataforma Canvas, en el link disponible para subir la tarea.
  - En caso de detectarse copia, los estudiantes involucrados tendrán la nota mínima sin apelación.
  - El archivo final subido a la plataforma debe ser un archivo comprimido con el siguiente formato:

    `ApellidoPaterno_ApellidoMaterno_letraInicialNombre.extensión`
    Ejemplo: Juan Pérez Valdivia `Pérez_Valdivia_J.zip`
  - El código o los códigos, y otros archivos deben estar comprimidos (zip o rar) en un carpeta que debe tener el formato antes mencionado.
  - El código debe estar en un archivo de extensión ***.py*** y debe ser nombrado igual que el archivo comprimido.
  - Si considera necesario, se puede agregar un archivo de texto plano, llamado `LEEME.txt`, donde se pueden agregar instrucciones para que el profesor pueda ejecutar de manera correcta su tarea. Sin embargo, no se permiten instrucciones en la cuales se deba intervenir el código para el buen funcionamiento.
  - En caso de nos seguir alguna de las condiciones o instrucciones será penalizado con **1.0 punto** por cada infracción.