Ejercicios para Resolver y Algunos Ejemplos

Programación II - Parte II

(Trabajos Prácticos que el alumno resuelve, incluye en la carpeta y/o explica en clase)

Índice

IMPORTANTE: Creación y Uso de tus Prompt. Aprender a escribir tus propios **prompt** tal como lo hicimos en clase, y crear los archivos "txt" que leerás con tus programas, es muy importante. Aprenderás a usar tu IA como herramienta de trabajo, ya que durante la corrección de carpeta, podrás usar con tu programa el archivo "txt" que traes de tu casa o crear uno nuevo con el promp que tienes.

Recuerda que tu profesor puede solicitar que crees un nuevo archivo de texto para que tu programa use, o que escribas un nuevo **prompt** y crees otra vez el archivo TXT para usarlo con tu programa durante la corrección.

Por este motivo deberás guardar tu prompt usado en un archivo de texto, que deberás nombrar "Promp_NomArchivo.txt", donde "**NomArchivo**" es el Nombre que tiene el archivo que usará tu programa. **Por ejemplo**, (levanta tus antenas y presta atención): si tu Prompt crea (genera los datos) para el archivo "**Numeros.txt**", el archivo de texto que contendrá tu prompt, deberá llamarse "Prompt_Numeros.txt" y encontrarse junto al archivo "**Numeros.txt**", que usará tu programa.

Nro	Más Enunciados con Lecto/Escritura de Archivos	Finaliza
1)	Continuamos usando el Archivo "**Alumnos_01.txt**" que usamos en anteriores programas, búscalo, descárgalo de la Nube o créalo nuevamente. Recuerda que la estructura del registro es: Nombre | Nota_01 | Nota_02 | Nota_03 **a)-** Ahora, debes cargar en la lista "**AA**", el nombre de cada alumno y en una segunda lista llamada "**AP**" guardar el promedio de las tres notas de cada alumno. Tener en cuenta en la posición Genérica N de la Lista "AA" encontraremos el nombre de un alumno y en la misma posición N de la lista "AP" el Promedio de ese alumno. **Segmento Código Ejemplo** C+=1 AA[C] = Nombre AP[C] = float((N1+N2+N3)/3) Una vez terminada la carga y cerrado el archivo, pero antes de finalizar el programa, debes mostrar en el monitor cada nombre y la nota promedio correspondiente a ese alumno. En el archivo se usa como separador de campos la coma ",". **Puedes Bajar Archivo desde:** Archivos/Alumnos_01.txt **b)-** Recrea el archivo "**Alumnos_01.txt**" con la IA de tu elección (mínimo 30 alumnos). En tu carpeta adjunta el **prompt** que hayas usado (debajo del enunciado), además, crea un archivo txt que contenga el promp y guárdalo junto con tu programa. Lo usaremos durante la corrección para crear el archivo y evaluar tu trabajo.	
2)	Leer el Archivo "**Alumnos_01.txt**" y carga en memoria: En una lista, el nombre de cada alumno. En otra lista, guardar (en cada posición) el promedio de las notas del alumno. Recuerda que la estructura del registro del archivo "Alumnos_01.txt" es: Nombre | Nota_01 | Nota_02 | Nota_03 **Parte A:** Crear el archivo "**Alumnos_02.txt**" con el contenido de Ambas listas. Ten presente que cada registro debe contener: Nombre | Promedio	

Nro	Más Enunciados con Lecto/Escritura de Archivos	Finaliza

Parte B: Antes de finalizar el programa, recorrer e imprimir el contenido de ambas listas, mostrando en cada renglón del monitor, los siguientes datos:

Nombre	Promedio

En la parte del programa que corresponda, debes usar las funciones:
- CalculaPromedio( ), que recibe las tres notas, calcula y retorna el promedio. El promedio deberá tener tantos decimales como se le indique a la función, y en cuando no se le pase este dato como parámetro, la función deberá retornar el promedio con estrictamente dos decimales. Deberás usar un **Parámetro Omitido**.
- MuestraNombre_y_Promedio( ), que debe recibir el nombre y promedio e imprimirlos según formato.

Como separador de campos usamos la coma ",".

Descarga Archivo desde:	Archivos/Alumnos_01.txt

Vectores con Más de un Elemento (campo) en Cada Posición

Clase Teórica xx/xx/2026

Comencemos con una breve descripción: En cada posición de una lista (vector), puedes guardar más de un campo o elemento, tantos como te resulten necesarios. Esto lo podremos definir e incluso inicializarlo en el momento de la definición del vector o al Insertar una elemento o grupo de elementos.

IMPORTANTE: A continuación te dejo la definición de un vector de 5 (cinco) posiciones, en el que cada posición almacena dos elementos: El primero (como siempre en la posición cero), será una cadena de caracteres (puede ser un nombre o lo que prefieras) y el segundo (en la posición uno), un número, o lo que prefieras. **Analiza el Código**:

Definición:	**Cantidad = 5** Vector = [[**' ', 0**] for i in range(Cantidad)]

Función Carga:	def CargaVector(V, C):
	  for F in range(C):
	  V[F][0] = input("Ingresar texto: ")
	  V[F][1] = input("Ingresar Numero: ")
	  print("\a")

Función Muestra:	def MuestraVector(V, C):
	  for F in range(C):
	  print(V[F][0], end=" ") **# No Baja Renglón**
	  print(V[F][1]) **# Si Baja Renglón**
	  print("\a\n")

Programa Principal:	CargaVector(Vector, Cantidad) MuestraVector (Vector, Cantidad)

Bajar código Completo:	Codigo/07_Listas_Vectores_020_Campos_01_Crea_Carga_Muestra_01.txt

Acá te dejo varias formas en que puedes realizar la Generación de los datos que puedes cargar en el vector.

Bajar código Completo:	Codigo/07_Listas_Vectores_000_Declara_Define_C_Genera_Datos_Inicializacion_01.txt
Bajar código Completo:	Codigo/07_Listas_Vectores_020_Campos_01_Crea_Carga_Muestra_02.txt

---

Trabajamos con un vector dinámico. Analiza el código o pregunta y lo vemos juntos. Este enfoque es especialmente útil cuando no sabes de antemano cuántos elementos necesitarás.

Definición:	**Cantidad = 5** # Solamente para limitar la carga
	Vector = [] # Vector vacío

Función Carga:	def CargaVector(V, C):
	  for F in range(C):
	  Texto = input("Ingresar texto: ")
	  Numero = input("Ingresar Numero: ")
	  V.append([Texto, Numero]) **# Inserto Elementos**
	  print("==========\a")

Función Muestra:	def MuestraVector(V, C):
	  for F in range(C):
	  print(V[F][0], end=" ") **# No Baja Renglón**
	  print(V[F][1]) **# Si Baja Renglón**
	  print("\a\n")

Programa Principal:	CargaVector(Vector, Cantidad) MuestraVector (Vector, Cantidad)

Bajar código Completo:	Codigo/07_Listas_Vectores_020_Campos_02_Crea_Carga_Muestra_01.txt

Otra forma en la que puedes usar este tipo de listas (vectores) dinámicas.

Bajar código Completo:	Codigo/07_Listas_Vectores_020_Campos_02_Crea_Carga_Muestra_02.txt

Enunciados con Vectores que tienen varios elementos en cada Posición

3) Repetiremos un programa anterior, pero ahora usando una única lista, en la que cada posición tenga dos campos. Presta mucha atención.

Leer el Archivo "**Alumnos_01.txt**" y carga en una única lista, el nombre y el promedio de las notas tres notas, de cada uno de los alumnos. Recuerda que la estructura del registro es:

Nombre	Nota_01	Nota_02	Nota_03

Parte A:
Crear el archivo "**Alumnos_02.txt**" con el contenido de la lista. Ten presente que cada registro (de la lista y del archivo) debe contener:

Nombre	Promedio

Parte B:
Antes de finalizar el programa, recorrer e imprime el contenido de la lista, mostrando en cada renglón del monitor, los siguientes datos:

Nombre	Promedio

Puedes Bajar Archivo desde:	Archivos/Alumnos_01.txt

Parte C:
Recrea el archivo "**Alumnos_01.txt**" con la IA de tu elección (mínimo 45 alumnos). En tu carpeta adjunta el **prompt** que hayas usado (debajo del enunciado), además, crea un archivo txt que contenga el promp y guárdalo junto con tu programa. Lo usaremos durante la corrección para crear el archivo y evaluar tu trabajo.

4) **Parte A:** En el Archivo Peliculas.txt, en cada línea (registro) contiene los siguientes campos:

Campo	Descripción
Genero	Los géneros, identifican el tipo de película, y es un número entero, que oscila entre 0 (cero) y 50 (cincuenta). Pueden faltar algunos valores.
Código de Película	Es un número entero, no correlativo que identifica cada película.
Titulo	Nombre de la película, que puede contener una o varias palabras
Año Lanzamiento	El año en que se estreno la película, y es un numero entero siempre de 4 dígitos, como todo año calendario

En el registro, se usa como separador de campo una coma ",".

Hacer un programa que Cargue los datos en memoria (lista o vector) y luego a continuación, según se **seleccione en un menú**, cualquiera de las siguientes opciones:

1- Genere y muestre listado de todas las películas.
2- Preguntar al Usuario que genero quiere listar, y mostrar todos las películas (código y Titulo) pertenecientes a ese genero.
3- Preguntar al Usuario un Año de lanzamiento, y mostrar todas las películas (Género, Código y Titulo) Con el año de lanzamiento mayor o igual al solicitado.
9- Terminar el proceso.

Parte B: Modificar la Parte A, agregando las Opciones:

4- Agregar los datos de una película a las ya existentes en memoria.
5- Guardar los datos cargados en memoria en el archivo original (reemplazándolo)

Sugerencia: Carga todo en un vector, en donde cada posición contiene una película y

recórrelo buscando lo que se te solicite en cada opción del menú.
Parte C: Crea al archivo con la IA de tu elección. En tu carpeta adjunta el **prompt** que hayas usado (debajo del enunciado), además, crea un archivo txt que lo contenga y guárdalo junto con tu programa. Lo usaremos para crear el archivo y evaluar tu programa.

5) **Consulta a Registros de Pacientes**: Procesa el archivo "**Pacientes.txt**". Los registros del archivo usan como separador de campo el Punto y Coma ";" y contienen los siguientes campos:

NroDocumento	NombrePaciente	Edad	Diagnóstico

Descripción de los campos:

NroDocumento:	Número de documento, que este ejemplo, para simplificar, solo será un número mayor que 0 (cero) y menor o igual que 1000 (mil), y no se permiten repeticiones.
NombrePaciente:	Este campo, contiene el nombre completo del paciente: Solo primer apellido (en caso de tener más de uno), y el primer nombre (en caso de tener más), ambos separado por una coma
Edad:	Es un numero entero, y debe estar comprendida entre 0 y 125 años
Diagnostico:	Es un texto que puede contener letras, números, puntos, guiones, comas y espacios en blanco. Con una longitud máxima de 200 caracteres en total.

Requerimientos de tu programa. Crear un menú de selección, con el que puedas elegir una o varias de las siguientes opciones:

Primera Parte:

- **a)** Cargar en memoria los datos del archivo "**Pacientes.txt**". Si por error seleccionas otra opción y los datos todavía no fueron cargados, informa del error y pide al usuario otra opción.
- **b)** Ingresar un número de documento y mostrar los datos del paciente.
- **c)** Ingresar Número de documento y la letra "E" separados por "/" (Barra diagonal o Slash), y el sistema mostrara el nombre y edad del Paciente.
- **d)** Ingresar Número de documento y la letra "D" separados por "/" (Barra diagonal o Slash), y el sistema mostrara el nombre y Diagnóstico del Paciente.
- **e)** Hacer Nada.
- **f)** Hacer Nada.
- **g)** Termina el programa

Segunda Parte:

Modificar la primera parte, agregando al menú las siguientes opciones:

- **e)** Permitir cargar los datos de un nuevo paciente.
- **f)** Borrar y crear nuevamente el archivo "**Pacientes.txt**" con los datos existentes en memoria, respetando sus números de documentos origina.

Tercera Parte:

Crea nuevamente el archivo "**Pacientes.txt**" con la I.A. de tu elección. En tu carpeta, adjunta el **prompt** que hayas usado (debajo del enunciado), además, crea un archivo "txt" que lo contenga y guárdalo junto con tu programa. Posiblemente usaremos tu promp para crear nuevamente el archivo y evaluar tu programa.

6) Acá Próximamente nuevo Enunciado

7) OPCIONAL

Enunciado OPCIONAL, pero te lo recomiendo:

Se solicita desarrollar un programa en Python para gestionar y analizar ventas de productos. El programa debe realizar las siguientes tareas:

A). Carga de los Datos de Productos (vector para consultas).

Tu programa debe inicializar un vector llamado "Vpro" (vector de productos o artículos) para almacenar los datos de los productos comercializados por un negocio. Estos datos se encuentran guardados en un archivo de texto llamado "**Precios_01.txt**". Cada línea del archivo (registro) representa un producto y contiene tres campos separados por comas: Código, Nombre, y Precio.

OPCIONAL OPCIONAL OPCIONAL OPCIONAL OPCIONAL

Descripción y uso de los campos de cada registro

El Código del artículo, que es un número entero comprendido entre 1 y 100, se utilizará como índice para cargar los datos en el vector "Vpro". Por ejemplo, los datos del artículo con código 5 deberán ser almacenados en la posición 4 del vector (considerando que los índices de los vectores en Python comienzan en 0).

El nombre de cada artículo, puede contener una o varias palabras (por ejemplo " Carne Vacuna")

Precio, es un numero float (real) con 2 decimales. En el archivo la coma decimal esta representada con un punto.

Puedes descargar Archivo desde:	Archivos/Precios_01.txt

B). Lectura de las ventas realizadas (desde el teclado).

Una vez que el vector de productos esté cargado (cuando se termine la lectura del archivo), el programa debe entrar en un bucle que permita al usuario ingresar por teclado el código del artículo, cantidad vendida y entregada a cada cliente que realice una compra.

El bucle para ingresar las ventas realizadas, debe terminar cuando el cajero ingrese un código igual a 0 (cero).

El programa debe validar que el código de Artículo ingresado durante el registro de ventas sea válido, es decir: que el código exista en el vector de artículos "**Vpro**" y además que el código sea mayor que cero y menor o igual que 100. En caso de que no lo sea, el programa debe mostrar un mensaje de error y solicitar el ingreso nuevamente.

C). Generación de Reporte por monitor.

Al finalizar el ingreso de todas las ventas (cuando algún código de artículo sea 0), el programa debe calcular y mostrar por pantalla 3 (tres) reportes detallados. Estos reportes deben incluir la siguiente información:

- Primer reporte: Nombre y Cantidad de unidades vendidas por cada artículo.
- Segundo reporte: Nombre e Importe total recaudado por cada artículo.
- Tercer reporte: Importe total general recaudado de todas las ventas.

Requisitos Adicionales.

Recuerda incluir en la carpeta:

- El Diagrama de Flujo (que represente la lógica del programa).
- Codificación Python completa. Recuerda que el programa será testeado en la PC.
- El diagrama de flujo deberá corresponder al código, si hay diferencias el programa será rechazado.

IMPORTANTE: La carga del "Vector para consultas" deberá ser realizada por la Función "Lee_Archivo()" la que debe recibir por parámetro el nombre y ruta del archivo, y al terminar la lectura, retorna como parámetro el vector cargado (aunque con Python no sea necesario el retorno, deberás realizarlo - consulta con tu profesor).

Y para el Cálculo del importe Total vendido, debes usar/crear la función "CalculaElTotal()" que recibirá como parámetro el vector que hayas usado para contener el total por artículo)

Sugerencia: (solo para entendidos) para calcular el total por artículo, puedes agregar un campo extra en el vector "**Vpro**" donde sumarás todos los "**Precio * Cantidad**" de cada venta.

Otra alternativa puede ser: Generar o definir un segundo vector (paralelo) con dos campos (Código y Total).

D). Genera un nuevo archivo de texto "**Precios_01.txt**" usando la I.A. de tu preferencia, e

OPCIONAL	inclúyelo en tu Pendrive junto con tu programa. Debes insertar en tu carpeta el "prompt" que usaste, y recuerda guardarlo también en un archivo txt en tu Pendrive, te lo pediré y usaremos nuevamente durante la corrección de tu programa.	

Nro	Cosas Interesantes (Parte IV)	Finaliza
	Colores en el Texto (Parte 2): Si en tus programas debes resaltar algunas palabras o textos, acá te dejo otra forma simple y rápida de hacerlo. Los códigos ANSI, conocidos como **Secuencias de Escape ANSI**, son un conjunto de instrucciones que permiten modificar (entre otras cosas) el formato y el color del texto en la terminal. Estos códigos son especialmente útiles para crear y mejorar la visualización de interfaces de usuario y la interactividad de las aplicaciones en consola, volviéndolas más atractivas y dinámicas. Puedes bajar el instructivo de la nube con el link que te dejo (ya sabes como usarlo). Bájalo desde: ANSI_Colores_en_el_Texto.pdf	
COMO BORRAR	Limpiar la Pantalla o Consola de Trabajo: Esto se verá nuevamente, pero Ahora tienes el adelanto. Si has querido limpiar la Pantalla o consola de trabajo de **Spyder**, acá te dejo tres formas distintas para hacerlo. Puedes Bajarlo desde: Codigo/90_Interesante_005_Limpio_Pantalla_Spyder_01.txt	COMO BORRAR
	Arduino y Python Comunicándose (Parte 1): A continuación dispones de los códigos necesarios para comunicarte con Arduino. Analízalos y.... "**Al Infinito y Más Allá!**" ☺. a)- Python Lee datos en la PC y envía. Arduino Recibe los Datos. Descargá el Código desde: Codigo/40_Arduino_001_Arduino_Leyendo_PC_01_Inicio_A.txt b)- Arduino Envía Datos y con Python los Leo en la PC. Descargá el Código desde: Codigo/40_Arduino_001_PC_Leyendo_Arduino_01_Inicio_A.txt c)- Este programa adicional, permite que verifiques si el puerto donde esta conectando Arduino esta activo (o con alguien ya conectado) y si no lo está, avisa en que puertos ya hay algo conectado. Descargá el Código desde: Codigo/40_Arduino_001_Verifico_Puerto_01_Si_Arduino_Esta_Conectado_A.txt	

Nro	Enunciados donde Interactuamos con Arduino (Opcional - Este grupo lo puedes omitir)	Finaliza
8)	Hacer un programa Python, en el que se visualice un menú con el que se pueda ordenar a la placa Arduino, prender o apagar tres led (Rojo, Verde, Amarillo). Al elegir un led en el menú, si esta apagado se prende y si esta prendido se apaga. Agregar una cuarta opción al menú, para terminar el programa. **Nota:** Si realizaste el programa opcional anterior, podes Saltarte este ejercicio.	
9)	Acá Próximamente nuevo Enunciado	
10)	Acá Próximamente nuevo Enunciado	

Nro	Más Enunciados con Lecto/Escritura de Archivos	Finaliza
11)	Este ejercicio tiene Tres partes, Elige si resolverás la opción A o la opción B. Leyendo el Teclado. **a)-** Hacer un programa que lea por teclado una serie de pares ordenados (X,Y), que representan el numero de Sensor y la Temperatura registrada por ese Sensor. Grabar en el archivo "Temperaturas_01.txt" todos los pares ordenados. Fin de Carga cuando se lea el par ordenado (0,0). Leyendo un Sensor Arduino. **b)-** Leer por el puerto serie de tu PC pares ordenados	

Nro	Más Enunciados con Lecto/Escritura de Archivos	Finaliza
	(X,Y) enviados por un Sensor de temperaturas controlado por Arduino. El Primer valor representa el número de Sensor y el segundo la temperatura registrada. Grabar en el archivo "Temperaturas_01.txt" todos los pares ordenados. Fin de Carga cuando se lea el par ordenado (0,0), o Arduino detenga el envío de datos por 60 segundos. Cuidado con esta condición. Para los dos primeros puntos "**a**" o "**b**". **Controlar que** los datos grabados cumplan con las siguientes condiciones: - El número de Sensor sea mayor o igual que 1 (uno) y menor o igual que 99 (noventa y nueve). - Las temperaturas son números reales (float), todas positivas y menores o iguales que 70 Grados. - Si algún dato no cumpliera con las condiciones estipuladas, ignorar el par (no lo grabes). y esperar el próximo. - Al grabar el archivo, debes usar como separador de campo una coma. **c)-** Hacer un programa que lea el archivo "Temperaturas_01.txt" e informe, cuantas lecturas realizo cada sensor, cuantas lecturas correctas se grabaron en total en el archivo.	
12)	Este ejercicio tiene varias partes, Parte A para repasar y las otras partes, hay que resolverlas. **a)-** Para comenzar, sugiero repases como "Reconocer los Tipos de Datos" que estudiamos anteriormente, para esto puedes bajar de la nube dos ejemplos que ya habíamos utilizado. Código: Codigo/90_Interesante_001_Reconoce_Caracter_a_que_tipo_Pertenece.txt Código: Codigo/90_Interesante_001_Reconociendo_Tipos_de_Numeros.txt **b)-** Hacer un programa que cree el archivo "**Numeros_01.txt**" y contenga un número en cada registro, los números deben ser enteros positivos y leídos desde el teclado. Se indica la finalización de la carga de datos con un número estrictamente menor a 1 (uno). Debes cargar mínimamente 10 registros. **c)-** Usando como modelo la parte anterior de este ejercicio, tu programa deberá controlar que los números leídos sean realmente **Números. D**escarta los números que no sean enteros, caracteres y palabras, informa del error y lee nuevamente el valor erróneo. En esta parte, usar la función "**leeUnNúmero( )**" que debe leer el número y hacer el control solicitado. Como separador de campos usamos la coma "**,**". **d)-** Crea al archivo con la IA de tu elección. En tu carpeta, adjunta el **prompt** que hayas usado (debajo del enunciado), además, crea un archivo "txt" que lo contenga y guárdalo junto con tu programa. Lo usaremos para crear el archivo y evaluar tu programa.	

Eliminar Caracteres en Blanco, y Algo Más.

Esto ya se analizo anteriormente, pero siempre es bueno repasarlo y Profundizar un poco más. Es muy útil y Práctico. Como limpiar caracteres indeseados de los renglones o registros leídos por teclado o de los archivos.

lstrip( ) devuelve una copia de la cadena con los caracteres **iniciales** en blanco (no visibles) eliminados. También tiene en cuenta los tabuladores y saltos de línea.

rstrip( ) devuelve una copia de la cadena con los caracteres **finales** en blanco eliminados. También tiene en cuenta los tabuladores y saltos de línea.

Recuerda que: Cualquier objeto de tipo **string** puede implementar los métodos **lstrip( )** y **rstrip( )**.

Código	En la Pantalla Veremos
cadena = ' \t\t\n hola \t\t\n ' print(cadena.**rstrip( )**) print(cadena.**lstrip( )**)	Hola Hola

-o-

Recuerda que también dispones del comandos **strip( ),** que hace el solo, lo que hacen los métodos **rstrip( )** y **lstrip( )** juntos.
Baja los códigos de la nube y analízalos. Te servirán de referencia.

Ejemplo:	Codigo/08_Archivos_004_TXT_000_Limpio_Datos_Leidos.txt
Ejemplo:	Codigo/08_Archivos_004_TXT_000_Separo_Datos_Leidos.txt

Y revisa este otro ejemplo, que te dará muchas opciones para trabajar:

Ejemplo:	Codigo/08_Archivos_004_TXT_000_Limpio_Datos_Leidos_con_DOS_Separadores.txt

Nro	Más Enunciados con Lecto/Escritura de Archivos	Finaliza
13)	La empresa "**RompeTodo S.R.L.**" se dedica a transportar ladrillos desde los cortaderos (fábrica de ladrillos) hasta el lugar donde serán utilizados (construcción). La empresa dispone de 99 camiones, numerados correlativamente, comenzando por el camión numero 01 hasta el camión número 99 (ambos números incluidos).  **Parte A)** Hacer un programa que escriba en el archivo "**Ladrillos_01.txt**" en el que cada registro debe contener el número de camión y cantidad de ladrillos que el camión trasportó en ese viaje. Debes comprender y recordar que, cada registro representa un viaje de un camión, y si algún camión realiza varios viajes, entonces en el listado y en el archivo, encontraremos varios registros (un registro por cada viaje realizado por ese camión). Es importante realizar las siguientes operaciones y controles: • Los datos que cargarás en el archivo, se encuentran anotados en el listado (escritos en hoja de papel) y lo deberás leer renglón a renglón. Los datos, están separados por un espacio en blanco, lee el renglón y separa los datos para verificarlos. • Durante la carga verificar que los números de camión se encuentren comprendidos entre 1 (uno) y 99 (noventa y nueve). Esto deberá ser controlado por la función ControlaCamiones( ). • Siempre, la cantidad de ladrillos en cada viaje, debe ser mayor que 0 (cero). Esto será controlados por la función ControlaLadrillos( ). • En todos los casos, si se encontrara algún error durante la carga, avisar del error, ignorar ese dato y leerlo nuevamente. **Parte B)** Hacer un programa en el que leyendo el archivo "**Ladrillos_01.txt**", creado en el punto **A** Calcule: • Cantidad de Viajes realizó Cada camión. • Cantidad de Ladrillos (en total) que transporto cada camión. • Cantidad total de ladrillos transportados. **Usar un vector con dos elementos en cada posición (Cantidad de Viajes/Cantidad de ladrillos) y Como separador de campos del registro usamos la coma ","** **Parte C)** Crea al archivo "**Ladrillos_01.txt**" con la IA de tu elección. En tu carpeta adjunta el **prompt** que hayas usado (debajo del enunciado), además, crea un archivo "txt" que lo contenga y guárdalo junto con tu programa. Posiblemente lo usaremos para crear nuevamente el archivo y evaluar tu programa.	
14)	**Pregunta:** Es posible desarrollar una función que retorne una lista, en la que se cargaron los datos leídos desde un archivo que contiene registros de un solo campo (solo un numero por registro)?. Si esto es posible, hazlo junto con el programa que use la función. Y asegúrate que funcione. Intégralo en tu carpeta.	

Nro	Más Enunciados con Lecto/Escritura de Archivos	Finaliza
15)	**Practico Adicional E:** Desde hoy, deberás tener insertado en la carpeta el Cálculo de cualquier Raíz de Cualquier Número. Recuerda que si se presentó un error en el calculo, Informar del error y causa del error. ***IMPORTANTE: Este trabajo se debe realizar consultando con Alguna IA. Incluyendo el prompt en la carpeta, debajo del enunciado y en un archivo "txt".*** Agrega adicionalmente un comentario, en donde indicas que problemas se presentaron con la IA, durante la solicitud hasta lograr tu objetivo.	

Ordenamiento de Listas/Vectores

Clase Teórica

Acá encontrarás la teoría y un ejercicio que contiene dos partes, resuelve ambas. 21/10

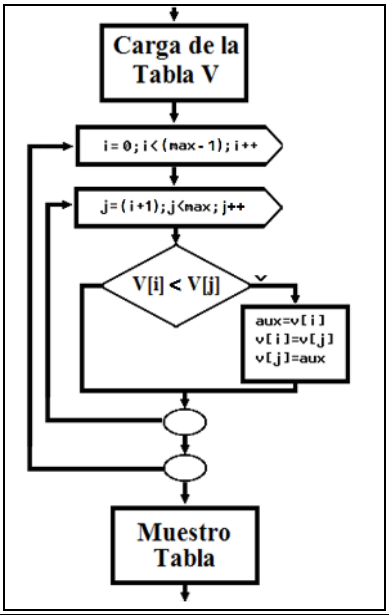
Existen muchos algoritmos para ordenar los elementos de un vector (ascendente o descendente). Estos algoritmos permiten ordenar vectores con componentes de tipo int, float, string o cadenas de caracteres, etc. En este último caso el ordenamiento es alfabético.

Uno de los primeros algoritmos que surge, consiste en comparar el primero con todos los siguientes, y si encuentro alguno mayor que el primero, intercambio posiciones, de tal forma que al terminar el recorrido, el elemento mayor queda en la primera posición.

A continuación, me posiciono en el segundo elemento y (como con el primero) lo comparo con todos los siguientes, quedando ahora en la segunda posición, el mas grande de los que quedaron.

Nuevamente repito, pero ahora posicionado el tercero, etc. **(Este es el puedes ver en el diagrama de flujo)**.

Con el tiempo, surgieron mejoras de este método, por ejemplo "La Ordenación de burbuja" (que funciona **revisando cada elemento de la lista que va a ser ordenada con el siguiente, intercambiándolos de posición si están en el orden equivocado**. Es necesario revisar varias veces toda la lista hasta que no se necesiten más intercambios, lo cual significa que la lista está ordenada).



Ver acá como trabaja	img/Ordena_01.gif
Ver acá el código	Codigo/07_Listas_Vectores_013_Ordenar_Sobre_SI_Misma_(Numeros)_(Bubble_sort).txt
Ver acá otra forma:	Img/Bubble_sort_example.gif

Ahora a Trabajar.

Diagrama - Formato 01	Diagramas/Ordena_Vector_01.png
Diagrama - Formato 02	Diagramas/Ordena_Vector_02.png

Aclaración Importante: A continuación encontrarás dos formatos de diagramar, pero ambos hacen lo mismo. "**Max**" es la variable que al realizar la carga de los elementos en la tabla, se usó para contar la cantidad de números leídos y cargados. Podríamos haber escrito: **max = len(V)**

Clase Teórica

a). Analiza (haciendo la prueba de escritorio) y explica que Acciones realiza este segmento del diagrama de flujo.

b). Ahora, en esta parte del problema, deberás confeccionar el diagrama, prueba de escritorio y codificar en PC para el siguiente enunciado: Crear un programa que Cargue en memoria (en una Tabla o vector) los datos contenidos en el archivo "**Numeros_01.txt**" (El archivo ya debe estar creado y contener un número en cada registro), **realiza la acción mostrada en el diagrama** y antes de finalizar, graba los datos, en el mismo archivo que usaste para cargar la tabla (borra el original y escríbelo nuevamente). Para dejar claro, se pide que recrees o sobre-escribas el archivo original. Acá te dejo el archivo que deberás leer, ya creado, bájalo de la nube.

Puedes Bajar Archivo desde:	Archivos/Numeros_01.txt

(Asegúrate de entender y manejar bien este ejercicio. Será parte de la próxima evaluación)

Mira este código, solo si **no te sientes capáz de analizar** el diagrama planteado y adaptalo al enunciado.

Ordenamiento de Números	Codigo/07_Listas_Vectores_013_Ordenar_Sobre_SI_Misma_(Numeros)_(Metodo_Burbuja).txt	
Ordenamiento de Texto	Codigo/07_Listas_Vectores_015_Ordenar_Sobre_SI_Misma_(Texto)_(Metodo_Burbuja).txt	
Métodos Python (de Todo)	Codigo/07_Listas_Vectores_015_Ordenar_(Metodo_Python).txt	

16) 21/10

a) Desarrolla una función que ordene una lista en forma ascendentes (la lista podrá contener palabras o contener números, cualquiera de los dos). Confecciona un programa que use esta función. Pruébalo y asegúrate que funcione.

b) Si crearas una función con el diagrama que acá te doy, que hará esa función? Explica el funcionamiento. Codifica el programa completo (imagina todo lo que falta), Puedes usarlo en la Primara Parte de este ejercicio?. Crea un enunciado y agrégalo a tu carpeta.

Diagrama:	Diagramas/Vector_Parte_Verif_Orden_01.png

17) 21/10

TRABAJO DE INVESTIGACIÓN PARA INCORPORAR EN LA CARPETA.

a)- En Python, cual es la diferencia entre una **Función** y un **Método**?

b)- Investiga, amplía, que hace el comando y parámetros puede usar?. Explica y confecciona un pequeño ejemplo (Código y prueba de escritorio) mostrando como se usa cada una de las funciones/métodos que a continuación se detallan. Identifica en cada caso, si se trata de una función, o un método asociado a una variable y/o tipo de dato. **Asegúrate de que este todo bien y completo**.

capitalize( )	Devuelve una copia de la cadena con la primera letra en mayúsculas
count()	**count()** es una función incorporada en Python que devuelve el recuento de cuántas veces aparece un objeto dado en la lista. El objeto a buscar dentro de la lista pueden ser letras, palabras, etc.
False	¿?
isdigit( )	**isdigit( )** es un método integrado que se utiliza para el manejo de cadenas. El método isdigit( ) devuelve "***Verdadero***" si todos los caracteres de la cadena son dígitos. De lo contrario, devuelve "***False***".
islower( )	Este Método retorna **True** si todos los caracteres de una cadena están en mayúsculas, **False** en caso contrario.
isupper( )	Este Método retorna **True** si todos los caracteres de una cadena están en mayúsculas, **False** en caso contrario.
join( )	**Join( )** formará una cadena de caracteres (nueva) con los elementos de una lista. Los elementos se guardarán se copian en la cadena, separados por el carácter que especifiquemos como separador. Ver ejemplos en este apartado del uso de **split( )**
len( )	Investiga que hace, aunque seguro ya lo sabes.
lower( )	Devuelve una copia de la cadena texto toda en Minúscula.
None	**None:** es un tipo de dato. Python incorpora este tipo de dato, que estrictamente hablando se llama NoneType y cuyo único valor posible es "**None**". Este valor, puede ser asignado a cualquier variable o alguna funciones lo retornan cuando quieren decir que no se logro lo que se debía hacer o no se encontró lo que se buscaba. Por lo tanto es muy simple preguntar por "None". Acá dejo algunos ejemplos, investiga si tiene otros usos:

a = None
if a == None:
  print("**a** es None.")
else:
  print("**a** no es None.")
	Dado que se trata de un tipo de dato como cualquier otro, con la peculiaridad de que tiene un único valor posible, podemos realizar las comparaciones habituales.

a = 1
b = 2
if not **a** is None:
  print("**a** no es None.")
if **b** is not None:
  print("**b** no es None.")
	Dos formas de preguntar si es distinto de None, una más legible que la otra

a = None
if a is None:
  print("a es None.")
else:
  print("a no es None.")
	Método recomendado para realizar la comparación.

def imprimir(lista=None):
  if lista is None:
  with open("Ejemplo.txt") as Archi:
  print(Archi.read())
  else:
  for elemento in lista:
  print(elemento)
	Como parámetro por defecto, podríamos mostrar elementos de una lista o de un archivo.

c)- Investiga, amplía, explica y confecciona pequeño ejemplo. (igual que el punto anterior)

random	Generación de números aleatorios o al azahar. Para poder usar estas funciones/métodos hay que importar el **modulo random** a tu programa, y te brindará varios métodos, Investiga. Una Ayuda en tu investigación: A continuación te dejo un ejemplo con algunas cosas que por ahora te pueden resultar útiles. Codigo/90_Interesante_003_Generacion_Numeros_Aleatorios_B.txt
split( )	El método split(sep = None, maxsplit = -1) retorna una lista de palabras o tokens usando sep como cadena de separación. Básicamente, se utiliza para dividir o separar un string en partes. IMPORTANTE: Si no se pasa el argumento sep o este es **None**, eliminará cualquier espacio en blanco, incluyendo los del comienzo y fin de la cadena y cualquier carácter que se imprima en blanco, como **\n \t o \r**. Ver ejemplos en este apartado del uso de **join( )**
strip( )	Este método devuelve una copia de la cadena con los caracteres iniciales y finales en blanco eliminados. También tiene en cuenta los tabuladores y saltos de línea. Cualquier objeto de tipo **string** implementa el método **strip( )**. Muy práctico para la grabación y recuperación de los datos en archivos de texto con formato o estructura.
swapcase( )	Devuelve una copia de la cadena con los caracteres en mayúsculas cambiados por minúsculas y viceversa.
True	¿? Para que se utilizaría?
title( )	Devuelve una copia de la cadena con la primera letra de cada palabra de una cadena convertida en mayúsculas.
upper( )	Devuelve una copia de la cadena texto toda en Mayúscula.

18) En este ejercicio, trabajaremos con un programa que ya hemos usado antes (búscalo y encuéntralo), crea y carga el archivo "**Alumnos_01.txt**", que contiene el siguiente registro:

Nombre	Nota_01	Nota_02	Nota_03

Acá te dejo el archivo ya creado, bájalo de la nube o créalo nuevamente

Puedes Bajar Archivo desde:	Archivos/Alumnos_01.txt

El programa funciona bien, pero tiene un inconveniente, cuando se cargan las notas, no controla apropiadamente y puedes ingresar letras o palabras en vez de solamente las notas, guardando todo en el archivo y eso esta mal.

Baja el código de la nube, pruébalo, verifica el problema informado y corrígelo. Para esto debes crear una función de control, que asegure la correcta carga de las notas y posterior grabación en el archivo.

A tener en cuenta:
- Cuando se ingrese el valor de una nota incorrecto, avisar al usuario que se ingreso un valor erróneo, y leer otro valor en su lugar.
- Debes mantener la condición para finalizar la carga de datos.

Puedes Bajarlo desde:	Codigo/08_Archivos_003_TXT_Plano_Z_Escritura_001_Registro_Varios_Campos.txt

Cosas Interesantes (Parte V)

Versión Python Instalada: Antes de comenzar esta etapa, te recuerdo como puedes saber que versión de Python tienes instalado en la PC que estás trabajando.

Baja el código desde: Codigo/01_Secuenciales_000_Muestra_Version_Python.txt

Borrar la Pantalla: Si has querido BORRAR la Pantalla o Consola de trabajo de **Spyder**, acá te dejo varias formas de hacerlo. Importante para muchos programas.

Bájalo desde: Codigo/90_Interesante_005_Limpio_Pantalla_Spyder_01.txt

Cosas Interesantes (Parte V)

Sonidos con Python: Solo SI Tienes Audífonos o Parlantes conectados a Tu PC y quieres escuchar un FUERTE BEEP o Música durante la ejecución de tu programa. Pero antes de ejecutar los programas, descarga los sonidos, guárdalos en la carpeta "Sonidos" y colocas la carpeta "Sonidos" en la misma carpeta donde estés guardando los programas.

Bájalo desde: Codigo/Sonidos_mp3.rar
Bájalo desde: Codigo/90_Interesante_006_Sonido_con_Windows_01_Beep.txt
Bájalo desde: Codigo/90_Interesante_006_Sonido_con_Windows_05_mp3.txt

Sobrescribe la Línea en la Pantalla: Te serviría poder escribir varias veces en el mismo renglón, sobre escribiendo lo que había primero. Acá te dejo unos ejemplos, analízalos que seguro te servirán muy pronto.

Bájalo desde: Codigo/90_Interesante_007_Reimprime_en_Monitor_01_print.txt
Bájalo desde: Codigo/90_Interesante_007_Reimprime_en_Monitor_02_sys.txt

Rápido Menú con Python: **Te gustaría un MENU** con el que no debas estar luchando para que funcione bien?

Bájalo desde: Codigo/55_Menu_de_Seleccion_01_INICIAL_001_Basico_A.txt
Bájalo desde: Codigo/55_Menu_de_Seleccion_01_INICIAL_001_Basico_B.txt
Bájalo desde: Codigo/55_Menu_de_Seleccion_01_INICIAL_001_Basico_C.txt

Para visualizar apropiadamente los códigos ejemplo, Guárdalos en tu PC como **txt** (y luego los renombras), o directamente guárdalos como **py ,** donde grabas tus programas (para ejecutarlos).
Si te gustara hacer un Menú de selección un poco mas elaborado, acá te muestro como manejar los textos del menú:

Bájalo desde: Codigo/55_Menu_de_Seleccion_01_INICIAL_001_Idea_A.txt

Detecta Teclas Presionadas e ingresa los valores desde teclado sin usar "ENTER". Varios ejemplos distintos.

Codigo/85_Reconocimiento_01_Teclas_001_Reconozco_y_Hago_01_Comenzamos.txt

Rápido Ejemplo: Resumen de lo anterior. Que lo disfrutes. Pero si no puedes hacerlo funcionar, consulta con tu profesor. Recuerda que debes ya tener la Carpeta de sonidos dentro de la carpeta donde Guardaste el programa.

Bájalo desde: Codigo/Sonidos_mp3.rar
Bájalo desde: Codigo/90_Interesante_006_Sonido_con_Windows_07_MI.txt

Debes corregir en el programa la ruta o dirección donde se encuentre la carpeta de sonidos. :)

Inicia con: Codigo/Sonidos/Esto_es.mp3

A resolver ejercicios, primero en el pizarrón y luego en la PC.

Nro	Enunciados donde Controlamos el Tiempo	Finaliza
19)	Investiga y Resuelve: Medir el tiempo que tardas en presionar la Tecla **ENTER** y luego detener el programa por un tiempo determinado. Te dejo una secuencia de ejemplos, los que te recomiendo analices en orden, ya que cada vez se pone más interesante. Codigo/80_Tiempo_01_INICIAL_001_Controla_Intervalo_01.txt	
20)	Medir un intervalo de tiempo, sin detener el programa. Te dejo una secuencia de ejemplos, los que te recomiendo analices en orden, ya que cada vez se pone más interesante. Codigo/80_Tiempo_01_INICIAL_001_Controla_Intervalo_02.txt Codigo/80_Tiempo_01_INICIAL_001_Controla_Intervalo_03_A.txt Codigo/80_Tiempo_01_INICIAL_001_Controla_Intervalo_03_B.txt	
21)	Alterna 20 veces entre los caracteres "x" y "-" a cada intervalos regulares de tiempo. Pero siempre en el mismo lugar del monitor. Codigo/80_Tiempo_01_INICIAL_001_Controla_Intervalo_04_Alterna_Caracteres.txt	
22)	Un Reloj: Notaste como el tiempo se escurre, cuando estas sentado en la máquina	

Nro	Enunciados donde Controlamos el Tiempo	Finaliza

programando?

Deberás realizar dos programas en donde medirás el tiempo a como un Reloj y mostrando: **Días, Horas, Minutos, Segundos** Transcurridos desde el momento que inicie el Programa. En Ambos casos, actualizar el monitor (con el tiempo transcurrido) cada un segundo.

a)- Proceso_Esperado: Iniciando en cero, incrementa cuando corresponda los contadores de segundos, minutos, horas, y días.

b)- Proceso_Esperado: Toma del sistema un momento inicial, y desde ese momento, calcula el tiempo transcurrido (segundos, minutos, etc).

IMPORTANTE: Cuida y esmérate en este trabajo, ya que regresaremos en futuras actividades para mejorar este reloj. **Como sugerencia final,** trata de **NO USAR** sentencias que detengan la ejecución del programa, porque luego, lo adaptaremos y usaremos para medir tiempos que ocupan (tardan) algunos procesos determinados dentro de algunos programas.

Para visualizar apropiadamente los códigos ejemplo, Grábalos en tu PC, como **txt** (y luego los renombrarlos), o directamente como **py** (para ejecutarlos).

Analiza estos códigos, seguro te ayudarán.

Ver código **01**: Codigo/80_Tiempo_01_INICIAL_001_Controla_Intervalo_04_Alterna_Caracteres.txt
Ver código **02**: Codigo/85_Reconocimiento_01_Teclas_001_Reconozco_y_Hago_01_Comenzamos.txt

No Robes el código, Toma ideas. Aprende.

Ver código **03**: Codigo/80_Tiempo_01_Reloj_001_Inicial_01_Dias_Horas_Minutos_Segundos.txt

Nro	Enunciados con Problemas Variados	Finaliza

23) Mientras Lee el Archivo "**Alumnos_01.txt**", que contiene el siguiente registro:

Nombre	Nota_01	Nota_02	Nota_03

Puedes Bajar Archivo desde: Archivos/Alumnos_01.txt

a)- Debes cargar sus datos en cuatro listas o vectores (un campo del archivo en cada vector) y a continuación ordenar alfabéticamente las listas (por nombres), usando el método explicado anteriormente. Una vez dispongas de las cuatro listas ordenadas, debes generar (escribir) el archivo "**Alumnos_03.txt**", en el que a cada registro contendrá los siguientes campos:

NroOrden	Nombre	Nota_01	Nota_02	Nota_03

Donde "**Numero de orden**" estará representado por el numero de posición o índice que ya tiene cada nombre en la tabla (vector) ordenada. Recuerda que en ambos archivos, usaremos la coma como separador de campo.

b)- Modifica el programa Anterior, para que solo uses un vector, y en cada posición hayas guardado los cuatro campos que has leído del archivo.

Crear un Archivo de texto si no Existe, o Agrega Líneas al final, si YA Existe

Nro	Enunciados con Creación Y Modificación/Extensión de Archivos	Finaliza

24) Este ejercicio tiene tres partes, para estar completo, deberás realizar todas las partes.

a) Crear un programa que, al ingresar una serie de números, calcule de cada uno su factorial y guarde el número ingresado en una lista y el factorial en otra lista. Al finalizar deberá mostrar por el monitor ambas listas, respetando el siguiente formato:

El final de la lectura y carga de números (cantidad desconocida para el programa) queda a cargo del alumno.

Número	Factorial
xx	Xxx
..	...
xx	Xxx

Por último, antes de terminar el programa, escribir el

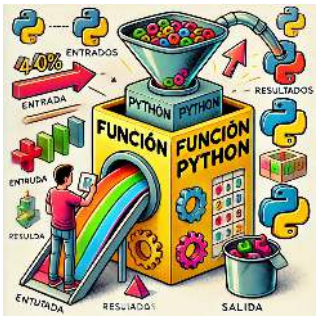
archivo "**Factoriales_01.txt**" el que en cada registro tendrá el número leído y su factorial. El archivo deberá contener un mínimo de 10 registros, usar como separador de campo la barra "/". El programa debe usar las funciones: **LeeUnNumero(), CalculoFactorial(), MuestraDosNumeros()**.

b) Como agregar Líneas en un archivo de texto, y si no existe crearlo: Cuando necesitas agregar líneas en un archivo de texto, pero si no existiera, lo creas y continúas insertando líneas, es en este caso en que solo debes aplicar este código. Bájalo de la nube, analízalo y adáptalo a tus necesidades.

Código Completo:	Codigo/08_Archivos_010_TXT_Crea_y_Escribe_o_Argrega_Lineas_Al_Final.txt

c) Crear un nuevo en el que continúes leyendo números, calculas el factorial y agregas el par ordenado al archivo ya existente "**Factoriales_01.txt**". Asegúrate de grabar al menos 10 nuevos registros y que cumplan con todos los requerimientos solicitados originalmente.

Las Funciones También se Pueden Asignar



Para Python, las funciones son objetos, por lo tanto, se las puede asignar a una variable para luego usar esa variable como una función.

Esto es simple y en ocasiones muy práctico, además relativamente común encontrar este método de programar (en video juegos y otros productos), por lo tanto se incluye en este apunte.

Por lo menos para que conozcas la existencia.

Ver Código en la Nube:	Codigo/09_Funciones_006_Asignacion_001_Ejemplo_01.txt	Veremos en Pantalla

```
# -*- coding: utf-8 -*-
"""
Creado 24/04/2022
Funciones Objetos - Asignación de funciones a Variables
@author: Horacio
"""
print("-------------------------------------------------")
print("Funciones Objetos - Asignación de funciones a Variables")
print("-------------------------------------------------\n")

# Definición de funciones
def Lee_Valor( ):
    Nro = int(input("Ingresar un número: "))
    return Nro

def Calcula_Cuadrado( Nro ):
    return Nro**2

def Muestra_Valor( Nro ):
    print("El Cuadrado es: ", Nro)

print("------------ Comienza el programa ----------\n")

print("\nEjemplo 01 - Uso Nombre Originales")
Numero = Lee_Valor( )
Cuadrado = Calcula_Cuadrado( Numero )
Muestra_Valor(Cuadrado)

# Asigno las funciones a otros nombres
L = Lee_Valor         # Asigno a la Variable L
C = Calcula_Cuadrado  # Asigno a la Variable C
M = Muestra_Valor     # Asigno a la Variable M

print("\nEjemplo 02 - Uso Nombre Cambiados.")
Numero = L( )
Cuadrado = C(Numero)
M(Cuadrado)

print("\nEjemplo 03 - llamado Críptico.")
print("Cada función Toma lo que la otra retorna.")
Muestra_Valor( Calcula_Cuadrado( Lee_Valor( ) ) )

print("\nEjemplo 04 - Críptico - Nombres Cambiados.")
print("Cada función Toma lo que la otra retorna.")
M(C(L( )))
# Termina Programa
print("\n-------------------------------------------------")
print("Programa Terminado.")
```

```
-------------------------------------------------
Funciones Objetos - Asignación de funciones a Variables
-------------------------------------------------

------------ Comienza el programa ----------

Ejemplo 01 - Uso Nombre Originales
Ingresar un número: 2
El Cuadrado es:  4

Ejemplo 02 - Uso Nombre Cambiados.
Ingresar un número: 3
El Cuadrado es:  9

Ejemplo 03 - llamado Críptico.
Cada función Toma lo que la otra retorna.
Ingresar un número: 4
El Cuadrado es:  16

Ejemplo 04 - Críptico - Nombres Cambiados.
Cada función Toma lo que la otra retorna.
Ingresar un número: 5
El Cuadrado es:  25

-------------------------------------------------
Programa Terminado.
```

May the Force be with you

UN POCO DE MAGIA

La asignación de funciones se puede extender a vectores, en donde cada elemento del vector puede contener una función y cada función hacer cosas distintas de las otras. Acá te dejo un código que puedes ver y analizar. Espero te aventures en este mágico remolino del universo de la programación.

Código Completo:	Codigo/09_Funciones_006_Asignacion_002_Ejemplo_01.txt
Código Completo:	Codigo/09_Funciones_006_Asignacion_003_Ejemplo_01.txt

Listas o Vectores, Repasemos y Aprendamos un Poco Más

Repasemos un poco y durante el repaso, llamaremos a estas estructuras por sus tres nombres (los más usados) para que realmente los identifiques cuando te los nombran. **En programación**, un vector es una estructura de datos que almacena una colección (conjunto) de elementos del mismo tipo, organizados en una secuencia lineal. Se puede pensar en un vector como un listado que permite acceder a sus elementos mediante un índice numérico (número de orden: primero, segundo, etc). Por ejemplo, en un arreglo unidimensional, cada elemento se puede acceder utilizando su posición en la lista, comenzando desde cero. Los vectores son útiles para almacenar y gestionar grandes cantidades de datos de manera eficiente, ya que permiten realizar operaciones como la búsqueda, ordenamiento y algunos leguajes con el Python, la inserción y la eliminación de elementos de forma rápida, etc. Además, su tamaño puede ser fijo (estáticos) y/o dinámico, **dependiendo** del lenguaje de programación utilizado.

Cuando usamos arreglos (vectores o tablas) en nuestros programas, según el leguaje usado, tenemos dos tipos o formas de usarlos: estáticos (Pascal, C, etc.) y dinámicos (Pascal, C, Pythom, etc).

a) Vectores Estáticos: Se pueden usar cuando de antemano, sabemos cuantos elementos tendrá. Por lo tanto mantienen su

tamaño (cantidad de datos) a lo largo de todo el programa.

b) Vectores Dinámicos: Se usan cuando desconocemos la cantidad de elementos que deberemos usar. Por lo tanto comienzan vacíos (sin elementos) y se le agregan, eliminan o cambian elementos, cada vez que sea necesario. Dicho de otra forma, crecen, cambian o decrecen de tamaño durante la ejecución del programa.

Ambos tipos de estructuras (estáticas y dinámicas) se recorren de igual manera, pero hay casos en los que se debe o quiere acceder a las posiciones o datos en forma directa (sin recorrer o búsqueda previa) y esto, para los temas que ahora estudiaremos (posicionamiento directo), resulta muy fácil con los vectores estáticos. Finalmente, la definición y en el tratamiento que se les da al momento de la carga de datos, los arreglos estáticos y dinámicos, son diferentes.
Es importante mencionar, que aun cuando sepamos de ante mano cuantos elementos deberemos guardar en un vector, **casi siempre** podremos (si eso preferimos) definirlo y usarlo dinámicamente.

A continuación, encontraras algunos segmentos de código donde podrás diferenciar como definir con Python vectores que podrías usar como estáticos y/o dinámicos.

IMPORTANTE: Cuando definimos un elemento del tipo caractér o grupo de caracteres (palabras, nombres, etc) que pertenecen o guardaremos en un Vector, podremos usar comillas simples y/o comillas dobles. Ambas formas son válidas y se pueden utilizar indistintamente, aunque hay una diferencia: Las comillas dobles permiten incluir caracteres como apóstrofes dentro de la cadena de texto, mientras que con las comillas simples será necesario utilizar el carácter de escape '\'. **En la práctica**, la elección entre uno u otro tipo de comillas suele ser una cuestión de preferencia personal o de consistencia en el estilo de codificación.

Ver Ejemplos con Python

Definición de Vectores Estáticos. Baja los códigos de la Nube y Analízalos.

Estáticos:	Codigo/07_Listas_Vectores_000_Declara_Define_A_Estaticos_01.txt

Definición de Vectores Dinámicos. Baja los códigos de la Nube y Analízalos.

Dinámicos:	Codigo/07_Listas_Vectores_000_Declara_Define_B_Dinamicos_01.txt

Nro	Enunciados con Vectores y Archivos	Finaliza
25)	Una empresa dedicada a la venta de materiales para la construcción, tiene un cuerpo de 10 vendedores (codificados del 1 al 10), que realizan su actividad telefónicamente en TODA Argentina. Por otro lado, dispone de 25 sucursales (codificadas del 1 al 25) ubicadas estratégicamente en el territorio nacional, dedicadas a la entrega de los productos vendidos. Cuando un vendedor realiza una venta, el sistema notifica a la sucursal de la zona (la más cercana al cliente) para que realice la entrega en forma inmediata. Durante un periodo determinado, todos los datos referentes a las ventas/entregas, fueron grabadas en el archivo "**Distrubucion_01.txt**", el que contiene el siguiente formato de registro: NroVend | NroSuc | NroCliente | NroFactura | Importe Adicionalmente debes saber que el separador de campo usado es "/", y que el archivo que debes usar lo puedes encontrar en la nube, bájalo. Hacer un programa, en el que usando dos vectores (uno para vendedores y otro para sucursales) calcule y muestre el total vendido por cada vendedor (sin importan que sucursal hace la entrega) y el importe total entregado por cada sucursal (sin importar que vendedor realizo la venta) en concepto de mercadería. Usar este Archivo: Archivos/Distribucion_01.txt	

Nro	Enunciados con Vectores y Archivos	Finaliza

26)

Resolver cada parte como un ejercicio independiente.

Parte A: Hacer un programa que cree y cargue el archivo "**Datos_01.txt**" en el que cada registro deberá contener:

Nombre	Antigüedad	Categoría	Sueldo

La finalización de la carga de datos y cantidad de registros, quedan a criterio del alumno.
IMPORTANTE: El programa deberá contener y usar las siguientes funciones.

- Lee un nombre	Lee el nombre que el usuario ingresara desde el teclado
- Lee numero entero	Podrás usarla para leer antigüedad y categoría desde el teclado. Recuerda entregar como parámetro de la función un texto descriptivo que le indica al usuario que debe ingresar
- Lee numero real	La deberás usar para leer desde el teclado, el sueldo. Por las dudas, recuerda que un número real, es un número o valor que puede o no tener una parte decimal.

Puedes usar este programa como un modelo de lógica.
Codigo/08_Archivos_004_TXT_004_Alumnos_01_Grabo_Alumnos_02_con_Promedio.txt

Parte B: Usando el archivo "**Datos_01.txt**" recién creado, hacer un programa en el que se *use como interfase con el usuario, un menú de selección*, que según sea la opción elegida, el programa realice alguna de las siguientes acciones:

a) Opción 0: Cargar en 4 (cuatro) listas los campos de cada registro. Contempla la situación en que si ya estaban cargadas las listas, sean borradas y se carguen nuevamente con los datos del archivo.
b) Opción 1: Ordenar las listas, alfabéticamente (por nombres), orden ascendente (de la **A** a la **Z**.
c) Opción 2: Ordenar las listas, alfabéticamente (por nombres), orden descendente (de la **Z** a la **A**.
d) Opción 3: Mostrar en el monitor los datos contenidos en listas, estén o no ordenados.
e) Opción 4: Buscar y mostrar el nombre y antigüedad del empleado con mayor sueldo.
f) Opción 5: Buscar y mostrar por monitor, el nombre y sueldo de todos aquellos empleados que tengan un sueldo mayor o igual al sueldo promedio abonado por la empresa.
g) Opción 6: Buscar y mostrar por monitor, los nombres y categorías de los empleados con una categoría superior o igual a 5.
h) Opción 7: Buscar y mostrar por monitor, los datos correspondientes a un empleado especifico (el nombre se ingresa por teclado).
i) Opción 8: Eliminar todos los datos (de las 4 listas), correspondientes a un empleado específico. El nombre se ingresa por teclado. Es posible eliminar un elemento cualquiera de una lista? Explica bien este procedimiento.
Recuerda que debes usar mínimamente las funciones:

- Lee un nombre	Lee el nombre que el usuario ingresara desde el teclado
- Muestra datos	Trata de usar una sola función que muestre el o los datos necesarios. Debes omitir parámetros.
- Ordenar	Ambas funciones. Orden Ascendente y Descendente.
- Busca	

Cosas Interesantes (Parte VI) - Uso de la Impresora

Como Imprimir: El uso de la impresora desde Python, tiene muchos niveles de complejidad, dependiendo de que y como quieras imprimir. La más simple, que en este momento veremos, consiste enviar a la impresora predeterminada de Windows, un documento (Archivo), y decirle que lo imprima. Este archivo debe contener los datos con el formato que desees ver impreso en la hoja de papel.
Más adelante estudiarás otras formas, que ampliarán mucho tu campo de acción.

Cosas Interesantes (Parte VI) - Uso de la Impresora

A continuación de dejo algunos ejemplos que te permitirán imprimir archivos. Analízalos bien, así podrás incluir estas funcionalidades en próximos programas. Recuerda que debes tener una impresora conectada a tu PC, o no podrás probarlo

Imprimir Archivos de Texto (Parte 01): Con estas rutinas podrás imprimir archivos de texto y archivos con imágenes. Solo debes especificar que imprimirás.

Bájalo desde:	Codigo/90_Interesante_008_Impresora_01_Archivo_Texto_01.txt
Bájalo desde:	Codigo/90_Interesante_008_Impresora_01_Archivo_Texto_02_Listado.txt
Bájalo desde:	Codigo/90_Interesante_008_Impresora_01_Archivo_Texto_03_Carpeta.txt

Instalación de Librerías para PDF: Para imprimir documentos PDF, instala la librería "fpdf2". Luego analiza y ejecuta los ejemplos.

```
pip install fpdf2
```

Si te quedan dudas o quieres leer más, accede a esta dirección (puedes traducir la Página) y sigue las instrucciones.

https://pypi.org/project/fpdf2/

Imprimir Archivos PDF (Parte 01): En este caso Podrás imprimir un Archivo PDF o todos los que se encuentren en una carpeta específica.

Bájalo desde:	Codigo/90_Interesante_008_Impresora_02_Archivo_PDF_01.txt
Bájalo desde:	Codigo/90_Interesante_008_Impresora_02_Archivo_PDF_02_Listado.txt
Bájalo desde:	Codigo/90_Interesante_008_Impresora_02_Archivo_PDF_03_Carpeta.txt

Archivos de Texto (Parte 02): Crea un Archivo Especial para la Impresora

Esta parte, es solo para mejorar el aspecto del archivo impreso, y te puede llevar un tiempo analizar, comprender y construir tu propio método. Acá encontrarás una forma de trabajar y construirlo; hay otras, pero puedes comenzar por esta.

Como ya se ha mencionado anteriormente, el método que usaremos para imprimir, es simplemente enviar un archivo de texto a la Impresora. Acá veremos como crear ese archivo, lo mejor y más bonito posible.

El archivo usado para impresión, deberá ser construido línea a línea. Para crear estas líneas, usaremos una cadena de caracteres patrón, a la que incorporaremos (cargando/insertando) los datos (sub cadenas de caracteres) con formatos y tamaños (largo) PREestablecidos, en los lugares o posiciones adecuadas, para que al grabar ese renglón, quede alineado con todos los otros renglones (encolumnados), aportando prolijidad y legibilidad al listado, luego, una vez grabado en el archivo, se limpian las cadenas usadas y comenzamos con los datos del próximo renglón.
Lo primero que debemos hacer, es dar un formato común a cada tipo de datos que estemos usando. Por ejemplo:

- Números Enteros: deberán tener un signo (los positivos podemos omitirlos, pero respetar su lugar), y un máximo de 5 dígitos (esto debe ser configurable para cada impresión).
- Números Reales: (float) Igual cantidad de dígitos enteros que los números enteros, el punto decimal y dos dígitos decimales.
- Las Cadenas de Caracteres: (descripciones y textos) una longitud predeterminada.

Y por demás esta decir que, todos esos valores y tamaños, deberán ser configurables para cada situación requerida. LO QUE NO COMPRENDAS, SOLO PREGÚNTALO AL PROFESOR.

COMENCEMOS.

a) Cada uno de los datos que usaremos para ensamblar la línea del archivo a imprimir, hay que transformarlo a cadena de caracteres y asegurarnos que tiene el tamaño adecuado para poder encolumnar todas las líneas. Acá te dejo las funciones para Transformar y adecuar los datos (tamaño de números enteros, reales y cadenas).

Bájalo desde: Codigo/15_Caracteres_10_Cadenas_11_Datos_para_Linea_de_Archivo_01.txt

b) En este paso, agregaremos al programa anterior, la parte en que armamos (construimos) la línea, que luego guardaremos (escribiremos) en el archivo imprimible (sugiero que repaces el tema de variables locales o preguntes al profesor).

Bájalo desde: Codigo/15_Caracteres_10_Cadenas_12_Ensambla_Datos_en_Linea_01.txt

Cosas Interesantes (Parte VI) - Uso de la Impresora

c) Acá, luego del ensamblado de datos en la cadena de caracteres, agregaremos los procesos de grabación en el archivo imprimible, repitiéndolo todo para cada registro o renglón. Dando por resultado un archivo listo para ser impreso.

Bájalo desde: Codigo/15_Caracteres_10_Cadenas_13_Graba_Linea_en_Archivo_01.txt

d) Y en teste punto, estén o no los Títulos de cada columna de datos, imprimiremos **(al fin...!)**.

Bájalo desde: Codigo/15_Caracteres_10_Cadenas_14_Imprime_Archivo_01.txt

e) Ya casi hemos terminado. Si has concluido con en proceso anterior, ya puedes ver el archivo imprimible y hasta lo puedes imprimir. Pero... Y los Títulos de cada columna de datos?.

Bájalo desde: Codigo/15_Caracteres_10_Cadenas_15_Titulos_01.txt

f) Ahora algo muy importante, de donde sacaremos los datos para todo este proceso de impresión?

Bájalo desde: Img/Tu_Cerebro_Con_Estudio_e_imaginacion.jpg

Archivos PDF (Parte 02) - Creación desde Python: Crea un Archivo PDF y úsalo para Presentar o Imprimir tu Trabajo.

Si este tema te interesa, puedes acceder al documento, copiando su nombre en el lugar acostumbrado (el documento se actualizará próximamente):

Bájalo desde: Archivos_PDF_Creacion_y_Uso.pdf

Esto se complementa con la Generación de Gráficos desde Python. Lo que veremos mas adelante y antes de terminar este documento. Puedes bajar de la Nube un rápido ejemplo.

Bajar ejemplo de la Nube: Codigo/60_Graficos_000_Ejemplo_Guia_01_Basico_con_PDF_A.txt

Nro	Más Enunciados con Vectores y Archivos	Finaliza

27) Hacer un programa que cree (escriba) el archivo "**VentasDiarias.txt**" que contendrá las ventas realizadas durante todo el año pasado, en un comercio de la zona. El archivo deberá contener el siguiente registro:

Año	Mes	Día	NroTicket	CodArtículo	PrecUnit	Cant

Los campos del archivo deberán estar separados por una coma "," y con el siguiente formato:

Año	Año en que se realizo la venta. Es un número entero Positivo de 4 dígitos.
Mes	Mes en que se realizo la venta. Es un número entero Positivo, con 2 dígitos y además debe ser mayor que cero y menor que 13.
Día	Año en que se realizo la venta. Es un número entero Positivo, con 2 dígitos y además debe ser mayor que cero y menor que 32.
NroTicket	Número de Ticket en el que se registro la venta. Siempre es un numero Positivo con 3 dígitos (mayor que cero y menor que 1000)
CodArtículo	Es el número que identifica a cada artículo. Siempre es un numero Positivo con 3 dígitos (mayor que cero y menor que 1000)
PrecUnit	Es el precio unitario del artículo. Es un numero real (float) Positivo, con 3 dígitos enteros, el punto decimal y dos dígitos decimales.
Cant	Es la cantidad de unidades vendidas. Es un número entero Positivo, con 2 dígitos y además debe ser mayor que cero y menor que 99.

IMPORTANTE:

a) Verificar que cada campo leído cumpla con las condiciones dadas. Y en caso de encontrarse un error, ignorar el campo y solicitarlo nuevamente.

b) Terminar el proceso de carga cuando el Usuario ingrese el año 1111 (cuatro unos)

c) Recuerda usar las funciones "**lee( )**" y "**controla( )**" para cada uno de los campos.

d) No uses Listas (vectores) en este programa.

e) Carga en el archivo no menos de 2 registros para cada mes del año.

Nro	Más Enunciados con Vectores y Archivos	Finaliza
28)	Crea un programa que lea el archivo **"VentasDiarias.txt"** (creado en un programa anterior) que: **Punto A:** Imprima un listado **(Impresora)** con el contenido del archivo. Cuando incorpores el programa en tu Carpeta de clases, incluye un comentarios indicando que cosa deberías mejorar (en el archivo que imprimes) al momento de crearlo, o si prefieres crear un archivo nuevo para que al imprimir, quede más prolijo y sobre todo entendible. **Punto B:** Mientras lees el archivo, muestra en el monitor, cada registro leído, con los campos ya separados. El listado que muestres en el monitor, deberá tener el siguiente formato:	

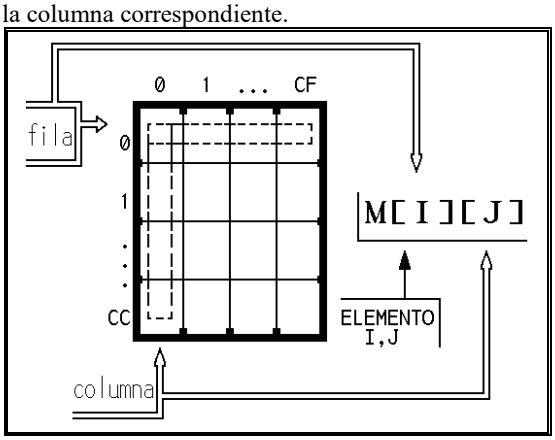
Fecha	NroTicket	CodArtículo	PrecUnit	Cant
AAAA/MMM/DDD	999	999	999.99	99
.....	...	...	.....	..
AAAA/MMM/DDD	999	999	999.99	99

Matrices o Listas Bi-Dimensionales

Imaginemos que queremos crear un programa con el cual podamos, de algún modo, almacenar los nombres y edades de diferentes personas. Esto lo podríamos resolver usando dos listas simples (vectores) o una lista que contenga dos listas en su interior (Matriz). Para cualquiera de las dos formas de escribir el programa, tendríamos una lista con los nombres y otra con los números de documento. Pero todo esto lo analizaremos más adelante.

Usando Tablas (Matrices): Las tablas o matrices se crearon para ayudarnos en múltiples circunstancias. Dado que una lista es capaz de almacenar múltiples listas en su interior, la podemos pensar como una tabla (con filas y columnas, igual que en el juego de la guerra naval) y asignamos valores en cada casilla según sea necesario, indicando la fila y la columna correspondiente.

A tu derecha, puedes ver una matriz dibujada, en la se muestran las filas y columnas y como identificar un elemento a través de los índices.



Debes saber que (igual que en los vectores), un elemento es el valor contenido en una posición determinada (un cuadradito) de la Matriz y en el dibujo, los valores de la variable (índice) "**I**" representa cada una de las filas de la matriz. Y los valores del índice (variable) "**J**" representa cada una de las columnas de la Matriz.

Cuando usamos tablas en nuestros programas, tenemos dos tipos: las estáticas y las dinámicas.

a) **Matrices Estáticas:** son aquellas que de antemano sabemos cuantas finas y cuantas columnas tendrá. Por lo tanto mantienen su tamaño a lo largo del programa.

b) **Matrices Dinámicas:** Son aquellas en la que desconocemos la cantidad de elementos que deberemos cargar. Por lo tanto comienzan vacías y sin elementos, y crecen en tamaños durante la ejecución del programa.

Ambos tipos de estructuras (estáticas y dinámicas) se recorren y utilizan de igual manera, pero cambia el tratamiento que se les da al momento de realizar la carga de datos.

Nro	Enunciados con Tablas Bidimensionales (Matrices)	Finaliza
29)	**Estructuras de datos en Memoria** **Dado en Clase**: Debes explicar línea por línea que hace cada uno de los códigos que acá encontraras, recuerda incluirlos en la carpeta.	

Ejemplo A y B: Cantidad y Contenido pre definido

```
tabla_matriz = [ ['Jose', 'Daniela'] , [18, 21] ]
```

Ejemplo C: Cantidad Predefinida, y hay que cargar datos

```
filas = 3
columnas = 3
valor_inicial = 0
tabla = [[valor_inicial for j in range(columnas)] for i in range(filas)]
```

Ejemplo D: Cantidad No definida, y hay que cargar datos

```
Empleados = [[ ], #Fila 1
             [ ], #Fila 2
             [ ]] #Fila 3 etc
```

Es muy importante destacar que:

- ✓ En cualquiera de los tres cuatro casos (A, B, C y D) siempre es posible agregar mas posiciones y mas datos si fuera necesario.

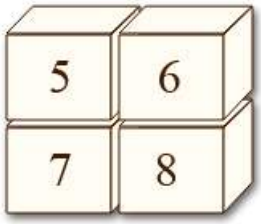
Nro	Enunciados con Tablas Bidimensionales (Matrices)	Finaliza

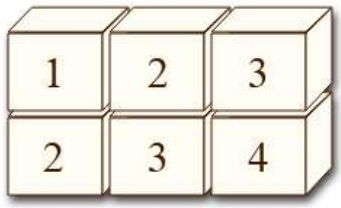
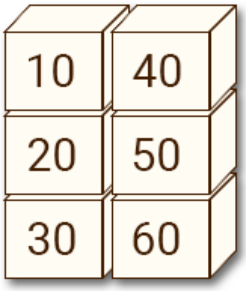
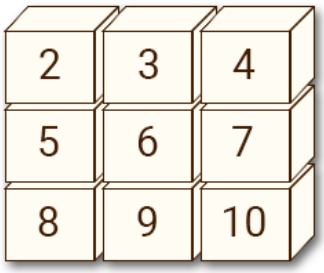
- ✓ Siempre, en las matrices que sus filas tienen siempre igual cantidad de columnas, no importa como se hayan definido, SE RECORREN DE IGUAL FORMA.
- ✓ Todas las Matrices pueden ser recorridas por Filas o por Columnas.
- ✓ RECOMENDACIÓN: Siempre dibuja la matriz que imaginas, así entenderás mejor la lógica que aplicas

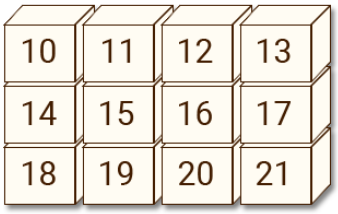
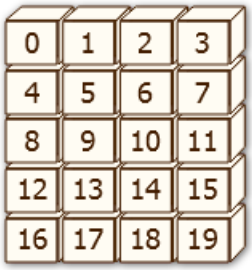
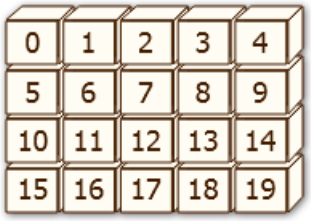
Baja los códigos completos desde la nube, analízalos y realiza la Prueba de escritorio.

A)-	Codigo/10_Listas_Matriz_Estatica_001_A_Definicion_y_Carga.txt
B)-	Codigo/10_Listas_Matriz_Estatica_001_B_Definicion_y_Carga.txt
C)-	Codigo/10_Listas_Matriz_Estatica_001_C_Definicion_y_Carga.txt
D)-	Codigo/10_Listas_Matriz_Estatica_001_D_Definicion_y_Carga.txt

30) Analiza detalladamente cada una de las matrices, y en tu carpeta (pegas o dibujas cada Matriz) junto a cada imagen, detallas cual es la posición y contenido de cada elemento, identificándolo de las forma: "A[**Fil**][**Col**] = x". Finalmente realiza el segmento de código donde quedarían definidas en un programa.

Matriz A	**El Contenido en Cada Posición de la Matriz A es:**
	A[0][0] = 5 | A[0][1] = 6 A[1][0] = 7 | A[1][1] = ¿?

Matriz B	Matriz C	Matriz D
		

Matriz E	Matriz F	Matriz G
		

Nro	Enunciados con Tablas Bidimensionales (Matrices)	Finaliza

31) Analiza detalladamente cada uno de los ejercicios (A, B, C y D). A continuación, realiza el diagrama de flujo, confecciona un enunciado valido y finalmente dibuja la matriz original y la matriz resultante para cada caso. Recuerda incluir todo en tu carpeta, incluyendo los enunciados.

Ejercicio A

```
# La Matriz tiene Números Reales
Matriz = [ [1.1,   23.45,  4.67],
           [56.78,  7.0,  89.99],
           [12.34, 34.56,  5.12] ]

Fil = len(Matriz)
Col = len(Matriz[0])

for i in range(Fil):
    for j in range(Col):
        print(f"{Matriz[i][j]:6.2f}", end=" ")
    print()
```

Ejercicio B

```
Matriz = [ [5 , 10 ,15],
           [20, 25, 30],
           [35, 40, 45] ]

Fil = len(Matriz)
Col = len(Matriz[0])

for i in range(Fil):
    Matriz[i][i] = 0

for i in range(Fil):
    for j in range(Col):
        print(f"{Matriz[i][j]:3}", end="  ")
    print()
```

Ejercicio C

```
Fil = 3
Col = 3
Val_Ini = 0
Matriz = [[Val_Ini for j in range(Col)] for i in range(Fil)]

Matriz[0][0] =  5
Matriz[0][1] = 10
Matriz[0][2] = 15
Matriz[1][0] = 20
Matriz[1][1] = 25
Matriz[1][2] = 30
Matriz[2][0] = 35
Matriz[2][1] = 40
Matriz[2][2] = 45

for i in range(Fil):
    Matriz[i][Fil - i - 1] = 0

for i in range(Fil):
    for j in range(Col):
        print(f"{Matriz[i][j]:3}", end="  ")
    print()
```

Ejercicio D

```
Fil = 3
Col = 3
Val_Ini = 0
Matriz = [[Val_Ini for j in range(Col)] for i in range(Fil)]

Matriz[0][0] =  5
Matriz[0][1] = 10
Matriz[0][2] = 15
Matriz[1][0] = 20
Matriz[1][1] = 25
Matriz[1][2] = 30
Matriz[2][0] = 35
Matriz[2][1] = 40
Matriz[2][2] = 45

for i in range(Fil):
    Aux = Matriz[i][i]
    Matriz[i][i] = Matriz[i][Fil - i - 1]
    Matriz[i][Fil - i - 1] = Aux

for i in range(Fil):
    for j in range(Col):
        print(f"{Matriz[i][j]:3}", end="  ")
    print()
```

32) Hacer un programa que cargue una matriz (Según lo explicado en clase) de 5x6 (5 filas con 6 columnas cada fila) con números enteros, y al finalizar la carga, calcule y muestre en el monitor la **suma de cada fila**.

33) Hacer un programa que cargue una matriz de 3x3 (3 filas con 3 columnas cada fila) con números enteros, y al finalizar la carga realice:

$$\begin{bmatrix} 2 & 4 & 3 \\ 1 & 9 & 7 \\ 6 & 5 & 3 \end{bmatrix} \leftrightarrow \begin{bmatrix} 6 & 5 & 3 \\ 1 & 9 & 7 \\ 2 & 4 & 3 \end{bmatrix}$$

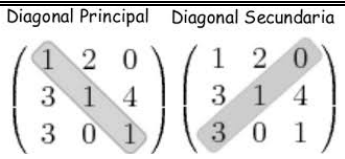
- **a)** El intercambio de los elementos de la primera Fila por los de la Tercera fila. Y antes de terminar el programa, mostrar en el monitor la matriz original y la matriz resultante.
- **b)** Modificar el programa para que solicite al usuario los números de filas debe intercambiar.

34) Hacer un programa que cargue una matriz de 5x5 (5 filas con 5 columnas cada fila) con números enteros, y al finalizar la carga, busque y muestre en el monitor, el número **mayor de cada fila**.

35) Hacer un programa que cargue una matriz de 3x3 (3 filas con 3 columnas cada fila) con números enteros, y al finalizar la carga realice:

$$\begin{bmatrix} 1 & 5 & 0 \\ 2 & -2 & -1 \\ 3 & 0 & 7 \end{bmatrix} \leftrightarrow \begin{bmatrix} 0 & 5 & 1 \\ -1 & -2 & 2 \\ 7 & 0 & 3 \end{bmatrix}$$

- **a)** El intercambio de los elementos de la primera Columna con los de la Tercera Columna. Y antes de terminar el programa, mostrar en el monitor la matriz original y la matriz resultante.
- **b)** Modificar el programa para que solicite al usuario los números de Columnas debe intercambiar.

Nro	Enunciados con Tablas Bidimensionales (Matrices)	Finaliza
36)	Hacer un programa que cargue una matriz (Según lo explicado en clase) de 4x3 (4 filas con 3 columnas cada fila) con números enteros, y al finalizar la carga, calcule y muestre en el monitor la **suma de cada columna**.	
37)	Hacer un programa que cargue una matriz (Según lo explicado en clase) de 4x3 (4 filas con 3 columnas cada fila) con números enteros, y al finalizar la carga, calcule y muestre en el monitor el **promedio de cada columna**.	
38)	Hacer un programa que cargue una matriz de 5x5 (5 filas con 5 columnas cada fila) con números enteros, y al finalizar la carga, busque y muestre en el monitor, **la posición del número menor de cada columna**.	
39)	Hacer un programa que cargue una matriz de 3x3 (3 filas con 3 columnas cada fila) con números enteros, y al finalizar la carga realice y muestre por monitor: **a)** La suma de los elementos que integran la Diagonal Principal. **b)** La suma de los elementos que integran la Diagonal Secundaria. **c)** Intercambia los elementos de la Diagonal Principal con los elementos de la Diagonal Secundaria. 	
40)	**Este ejercicio tiene dos partes** (dos programas) analiza bien lo solicitado en la segunda parte, para poder hacer la primera parte. Luego programa cada parte como un ejercicio independiente. **Parte A:** Grabar el archivo que usarás en la Parte B. Recuerda que: ➢ Son 6 censores que registra 24 grabaciones cada uno. ➢ La hora representa el número de orden, que deberás usar al grabar cada registro. ➢ Las temperaturas debes generarlas al azahar (número random) con valores mayores o iguales a -5 y menores o iguales a 45. Si no se te ocurre como hacer, te dejo una idea: Un "for" de 1 a 6 que contiene otro "for" de 1 a 24 que, adentro encontrarás la generación **ramdon** de la temperatura y graba el registro de 3 campos en el archivo. **Parte B:** Seis censores de temperaturas, ubicados en un hospital, graban en el archivo "**Mediciones.txt**" las temperaturas que se registran automáticamente cada 60 minutos. El archivo mencionado se encuentra en el servidor del edificio central. Importante: ➢ Los censores están numerados de uno en uno, comenzando por el 1 (uno). ➢ La primera grabación realizada en el archivo, corresponde a la temperatura registrada a la una de la madrugada, la segunda grabación a la temperatura registrada a las dos de la madrugada, y así sucesivamente hasta la grabación 24 que corresponde a la temperatura registrada a las 24hs (12 de la noche). ➢ Cada registro del archivo contiene tres campos: Nro de Censor, Nro de Grabación y Temperatura Registrada. Se pide que hagas un programa que lea el archivo "**Mediciones.txt**" y Cargue en memoria todos los datos contenidos en una matriz de 6 filas con 24 columnas cada una. Una vez finalizada la carga, muestre en pantalla un menú se opciones en la se pueda elegir: **a)** Calcular y muestre la temperatura promedio por censor. **b)** Calcular la temperatura promedio del edificio a una hora determinada. **c)** A que hora registro la mayor temperatura cada censor. **d)** A que hora registro la menor temperatura cada censor. **e)** Finalizar el programa.	

Nro	Enunciados con Tablas Bidimensionales (Matrices)	Finaliza

41)

Una empresa dedicada a la venta de materiales de construcción, tiene un cuerpo de 10 vendedores (codificados del 1 al 10), que realizan su actividad telefónicamente en TODO el territorio Argentino. Por otro lado, dispone de 30 sucursales (codificadas del 1 al 30) ubicadas estratégicamente en el territorio nacional, dedicadas a la entrega de los productos vendidos. Cuando un vendedor realiza una venta, el sistema notifica a la sucursal de la zona (la más cercana al cliente) para que realice la entrega en forma inmediata.

Durante un periodo determinado, todos los datos referentes a las ventas/entregas, fueron grabados en el archivo "**Distrubucion_01.txt**", el que contiene el siguiente formato de registro:

NroVend	NroSuc	NroCliente	NroFactura	Importe

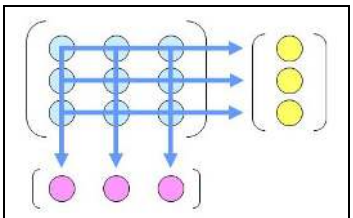
Adicionalmente debes saber que el separador de campo usado es "/", y que el archivo que debes usar lo puedes encontrar en la nube, bájalo.

Hacer un programa, en el que solo *usando una Matriz* (Una fila por sucursal y una columna para cada vendedor) calcule y muestre el total vendido por cada vendedor (sin importar que sucursal hace la entrega) y el importe total entregado por cada sucursal (sin importar que vendedor realizo la venta) en concepto de mercadería.

Usar este Archivo:	Archivos/Distribucion_01.txt

42)

Este programa ejemplo, fué **diagramado y codificado completamente en Clase**. Recuerda que debe estar incluido como un programa más en la carpeta.



Suma de Filas y Columnas

Cargar una tabla bidimensional (Matriz) de 4x5 (Cuatro filas con 5 columnas) con números enteros positivos menores que 100 (cien) y al terminar la carga de datos deberás:

- **a)** Crea/Genera el vector "**SF**" en el que cada elemento contiene la **Sumatoria de cada Fila** de la matriz.
- **b)** Busca en el vector "**SF**" e informa la posición y valor del mayor elemento.
- **c)** Crea/Genera el vector "**SC**" en el que cada elemento contiene la **Sumatoria de cada Columna** de la matriz.
- **d)** Busca en el vector "**SC**" e informa la posición y valor del mayor elemento.
- **e)** Antes de finalizar el programa, informa por el monitor, los totales calculados, indicando a que fila o columna corresponde.

Recuerda usar una función **lee** y otra que **controla** cada numero que cargas en la Matriz. Ejemplo para que analices y puedas avanzar con este tema tan importante de la programación.

A)-	Codigo/10_Listas_Matriz_Estatica_002_A_Vector_con_Suma_de_Filas.txt

B)-	Codigo/10_Listas_Matriz_Estatica_002_B_Vector_con_Suma_de_Columnas.txt

43)

Este ejercicio tiene **tres partes**: La primera, en la que debes realizar el análisis del código de un programa y 2 (dos) programas que debes hacer:

(Acá usamos **tabla = [[valor_inicial for _ in range(columnas)] for _ in range(filas)]**)

Análisis: Realiza la Prueba de escritorio y explicar línea por línea, que hace este código.

Codigo/10_Listas_Matriz_Estatica_001_C_Definicion_y_Carga.txt

Programa A: Generar (escribir) el archivo "**VentasKiosco_01.txt**" en el que cada registro contiene los siguientes 5 (cinco) campos: Mes, Dia, Desc, PUnitario, CanVendida. Los datos que grabaras en el archivo, correspondiente a cada una de las ventas realizadas a lo largo del último año. Usar como separador de campo la coma ",". Controlar que la carga de datos sea la correcto mediante funciones diseñadas para tal efecto.

Si algún dato es incorrecto, avisa del error con un mensaje y cárgalo nuevamente.

Nro	Enunciados con Tablas Bidimensionales (Matrices)	Finaliza
	Deberás dar por finalizada la carga de datos cuando se ingrese algún valor negativo para el campo "Mes". **Programa B**: Procesar los datos contenidos en el archivo "**VentasKiosco_01.txt**", usando una matriz de 12x31(filas para los 12 meses del año y 31 columnas para sumar los importes de las ventas realizadas en cada día). Una vez concluida la lectura del archivo, a continuación se solicita: Mediante un Menú de selección, según la tecla presionada: **a)** Mostrar el importe total vendido en cada mes (sumatoria de cada Fila) **b)** El día del mes que se realiza mayor recaudación (sumatoria de cada Columna) **c)** El mes que se recaudo más plata. **d)** Importe Total Recaudado en el año controlado.	
44)	Este ejercicio tiene tres partes, el análisis del código de un programa y 2 (dos) programas que deberás hacer tu solo: **Análisis:** Realiza la Prueba de escritorio y explicar línea por línea, que hace este código. Codigo/10_Listas_Matriz_Estatica_001_D_Definicion_y_Carga.txt **Programa A:** Generar (escribir) el archivo "**Persoal_01.txt**" en el que cada registro contiene los siguientes 4 (cuatro) campos: **Nombre, DNI, Antigüedad y Sueldo**. Usar como separador de campo la coma ",". Durante la carga de datos, debes usar mínimamente, las siguientes funciones de control: ➢ CtrlNombre( ) Función que verifica el nombre, donde solo se carguen Letras mayúsculas y minúsculas o espacios en blanco. ➢ CtrlDNI( ) Función que verifica el numero de documento, donde solo se carguen números enteros de hasta 7 dígitos. ➢ CtrlAntiguedad( ) Función que verifica la Antigüedad del empleado, donde solo se carguen números enteros mayores o iguales que 0 (cero) y menores que 60 años. ➢ CtrlSueldo( ) Función que verifica el sueldo del empleado, donde solo se carguen números reales mayores que cero. Si algún dato es incorrecto, avisa del error mediante un mensaje y cárgalo nuevamente. Deberás dar por finalizada la carga de datos cuando se ingrese la Palabra "**Fin**" en lugar del nombre de un empleado. **Programa B**: Cargar en memoria los datos contenidos en el archivo "**Persoal_01.txt**" Usando una matriz de 4xN (filas para los 4 campos y N columnas para contener todos los registros). Una vez concluida la lectura del archivo, a continuación se solicita: **e)** Mostrar un primer listado con el Nombre, Antigüedad y Sueldo de todos los empleados que tienen un sueldo mayor al sueldo promedio **f)** Mostrar un segundo listado con el Nombre, DNI y antigüedad de todos los empleados que tienen una antigüedad mayor o igual a 20 años. **g)** El nombre de los 5 empleados con mayor sueldo (Sugerencia: Carga un vector con nombre y sueldo de todos los empleados. Ordena el vector y muestra los 5 últimos o primeros según sea el orden que hayas ejecutado)	
	Enunciados con Matrices - Guardemos más de un Elemento en Cada Posición	
45)	**IMPORTANTE**: En cada posición de la matriz, puedes guardar más de un campo o elemento. A continuación te dejo la definición de una matriz estática de 4 (cuatro) filas x 3 (tres) columnas, en la que cada posición almacena dos elementos: una cadena de caracteres (puede ser un nombre o lo que prefieras) y un número. Analiza el Código: Definición: Filas = 4 Columnas = 3 Matriz A = [[**[' ', 0]** for i in range(Columnas)] for j in range(Filas)]	

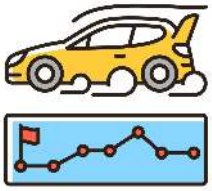
Función Carga:	def CargaMatriz(M, CF, CC):
	  for F in range(CF):
	  for C in range(CC):
	  M[F][C][0] = input("Ingresar texto: ")
	  M[F][C][1] = input("Ingresar Numero: ")
	  print("\a")

Función Muestra:	def MuestraMatriz(M, CF, CC):
	  for F in range(CF):
	  for C in range(CC):
	  print(M[F][C][0], end=" ")
	  print(M[F][C][1], end=" ")
	  print("\a")
	  print("\a")

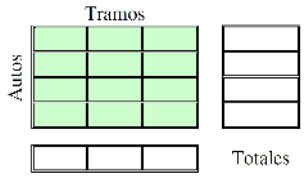
Programa Principal:	CargaMatriz(Matriz_A, Filas, Columnas) MuestraMatriz(Matriz_A, Filas, Columnas)

46) **Programa trabajado en Clase:** Durante una carrera de autos se grabó el archivo "Carrera_01.txt" en el que cada registro usa como separador de campo **"/"** y contiene los siguientes campos:

Tramo	NroAuto	Tiempo	Piloto



Sabiendo que la carrera se divide en 10 tramos, y en ella compiten 20 autos. Si algún auto abandona o no participa de algún tramo, el tiempo del, o los tramos no recorridos, figurarán en 0 (cero) y ese Auto debe ser descalificado y excluido de toda la competencia.



A la izquierda, dejo una imagen para que te des una idea, como será la matriz en donde guardas los datos del archivo, y los vectores (A la derecha y abajo) donde guardarías los datos generados, calculados o encontrados (según sea lo solicitado). Puedes usar dos matrices, una para el tiempo y la otra para el nombre del piloto o *una matriz en la que cada posición contenga el tiempo y el nombre.*

Una vez concluida la lectura del archivo, a continuación deberás calcular:

Mediante un Menú de selección, según la tecla presionada:

- **a)** Mostrar el número de auto y tiempo del ganador de la competición.
- **b)** El auto más veloz (seguro el ganador) y su tiempo de cada tramo.
- **c)** Listado de todos los autos y sus tiempos totales (Poner un Asterisco junto al Número de Auto si no participo en uno o más tramos). El listado deberá estar ordenado en forma decreciente acorde al tiempo.
- **d)** Listado de todos los competidores, tramo por tramo, indicando Coche, Tiempo, piloto. El listado deberá estar ordenado en forma decreciente en cada tramo (Cuidado, esto hay que pensarlo).
- **e)** El programa debe, cada vez que el usuario elija una opción del menú: mostrar lo solicitado, esperar a que el usuario presione una tecla, limpiar la pantalla y mostrar nuevamente el menú de selección.

Usar este Archivo:	Archivos/Carrera_01.txt

El Recolector de Basura


Una Imagen mejor que 1000 Palabras.

Garbage Collector o Recolector de Basura: El recolector de basura en Python, es un mecanismo automático que se encarga de liberar la memoria utilizada por cualquier variable (objetos) que ya no se esté utilizando en el programa. Cuando un objeto ya no tiene referencias a él, el recolector de basura lo identifica y elimina de la memoria para que pueda ser reutilizada. Esto ayuda a prevenir fugas de memoria y mantener el uso eficiente de los recursos del sistema. El recolector de basura en Python funciona de manera transparente (invisible) para el desarrollador y el usuario, lo que simplifica la gestión de memoria, que será automática y sin molestar. Es una característica importante que

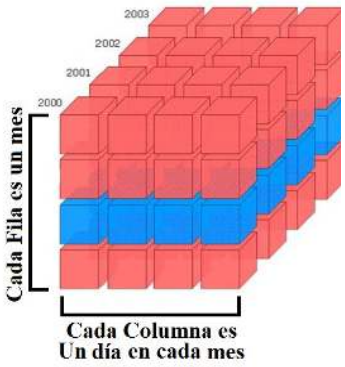
contribuye a la facilidad de uso y la eficiencia de Python como lenguaje de programación.

En Python, garbage collector utiliza un algoritmo de *conteo de referencias* y un *recolector de ciclos* para manejar la memoria:

- *Conteo de referencias*: Cada objeto en Python tiene un contador de referencias, que se incrementa cada vez que una nueva referencia al objeto es creada y disminuye cuando una referencia es eliminada.
- *Recolector de ciclos*: Este componente adicional es capaz de detectar ciclos de referencias, en los que un conjunto de objetos se referencia mutuamente, pero no son accesibles desde el código.

Listas n-Dimensionales

Introducción: *Comencemos con las Matrices Cúbicas o tridimensionales*, que son aquellas a las que podremos acceder a los datos usando tres índices, uno para cada dimensión.



A continuación dispones del código necesario para definir una matriz cúbica, cargarla y luego imprimir todos los datos, pero antes comprende el primer ejemplo.

Primer ejemplo: Si tenemos el archivo con datos históricos "**Negocio_01.txt**" y lo usamos para cargar una matriz de tres dimensiones. Los datos del archivo corresponden a los totales diarios (plata) de las ventas, realizadas en cada uno de los días correspondientes a los doce meses del año, a lo largo de 4 años 2000, 2001, 2002 y 2003.

Para comenzar, imaginemos una matriz cuadrada que tiene 12 filas (una para cada mes del año) y cada fila contienen 31 columnas (una para cada día del mes). Esta matriz cuadrada, representará un año completo, entonces pongamos 4 matrices, donde cada matriz será un año completo ☹..... **Pero....**, en vez de pensarlo como cuatro matrices cuadradas, podemos verlo como una matriz cúbica , es decir juntamos las cuatro matrices. Ver imagen.

Para acceder al dato que corresponde a un día determinado, deberemos usar tres índices, uno para identificar el año (que llamaremos rebanada), otro para el mes (Filas), y finalmente el tercer índice para identificar el día (columnas).

Bajar Código desde:	Codigo/10_Listas_Matriz_Estatica_Cubo_000_A_Carga_Recorre_y_Muestra.txt

Importante: En este tipo de listas, también disponemos de listas estáticas y dinámicas, se definen de igual forma que las cuadradas, solo que acá tendremos 3 índices, etc.

Nro	Enunciados con Tablas n-Dimensiónales (Matrices Cúbicas)	Finaliza
47)	Continuemos con el Primer Ejemplo de Listas n-Dimensionales. Usando el archivo "Negocio_01.txt" (que puedes bajar de la nube y ya contiene los datos), realiza el programa que lo lea y cargue los datos en una matriz Cúbica, y una ves terminada la carga, calcule y muestre en el monitor: **a)** Total vendido en cada año (total de cada Rebanada) **b)** Total vendido en cada uno de los 12 meses de los 4 años controlados, informando además que mes y que año estás procesando. **c)** Realiza un dibujo de la Matriz Cúbica que definas, e indicar detalladamente que contiene cada rebanada, cada fila, cada columna, etc. *Pautas Generales:* ✓ El separador de campo usado en el archivo es la coma. ✓ Utiliza la función "**Carga( )**", para leer el archivo y cargar la Matriz cúbica. Deberás pasar como parámetro, la matriz vacía. ✓ Formato de registro: Año | Mes | Dia | TotalDia Archivo Cargado: Archivos/Negocio_01.txt	
48)	El propietario de una cadenas de Kioscos, con 5 sucursales, centraliza TODAS las ventas realizadas en el archivo "**VentasSucursales.txt**" que contiene el siguiente formato de registro: NroSuc | NroVend | CodArt | Cantidad | PrecUnit **DESCRIPCIÓN DE LOS CAMPOS:** NroSuc: Numero de Sucursal, que oscila entre 1 y 5 NroVend: Numero de Vendedor, es un numero entero y puede oscilar entre 1 y 6 CodArt: Código de artículo, es un numero entero y puede oscilar entre 1 y 10 Cantidad: Cantidad de un artículo que vendió el vendedor en una misma transacción, este valor es un número entero que oscila entre 1 y 99. En caso de haber vendido más de 99 unidades de un mismo artículo en la misma transacción, el archivo contendrá más de un registro de ese vendedor con ese artículo.	

Nro	Enunciados con Tablas n-Dimensiónales (Matrices Cúbicas)	Finaliza

PrecUnit	Precio unitario del artículo. Este puede variar entre venta y venta, ya que los precios pudieron ser actualizados y siempre oscila entre $0.1 y $99.99

Desde acá podrás descargar el archivo ya creado y que deberás usar para calcular los totales Solicitados.

Archivo: Archivos/VentasSucursales.txt

Realizar los siguientes cálculos:

- **a)** Realiza un dibujo de la Matriz Cúbica que definas, e indicar detalladamente que contiene cada rebanada, cada fila, cada columna, etc.
- **b)** Que sucursal recaudo mayor importe en concepto de ventas?
- **c)** Importe total recaudado en conceptos de ventas por cada vendedor de cada sucursal.
- **d)** Cantidad vendida de cada Artículo.
- **e)** Artículo más vendido en cada una de las sucursales.
- **f)** Artículo más vendido en toda la cadena.
- **g)** Artículo más vendido por cada vendedor.
- **h)** Importe de la venta promedio realizada por cada vendedor
- **i)** En el código inserta un breve comentario indicando como realizas cada cálculo.

Una vez terminado los cálculos:

- **j)** Mostrar Totales en el monitor.
- **k)** Imprimir totales por la impresora (el diseño del formato de impresión queda a cargo del alumno).

Pautas Generales:

- ✓ El Archivo deberá ser leído una sola vez
- ✓ Cargar los datos en memoria usando una o varias matrices cúbicas (te sugiero veas el próximo ejercicio).
- ✓ Deberás usar como mínimo la función "CargaDatos( )" que lee el archivo y carga la/las matrices cúbicas con los datos leídos desde el archivo o calculados. Luego puedes usar todas las funciones que creas conveniente usar en tu programa.
- ✓ Puedes usar cualquier tipo de arreglos adicionales que consideres necesario para realizar los cálculos solicitados.
- ✓ Considera usar más de un arreglo **(matriz)**, uno para totales y otro para cantidades (te sugiero veas el próximo ejercicio).
- ✓ El archivo usa la coma como separador de campo.

Matrices Tridimensionales con más de un campo en cada Posición

Esta variante es muy simple, ya que solo necesitas inicializar con dos elementos cada posición de la matriz (arreglo o tabla). Veamos las partes de este proceso. Debes notar que el orden de inicialización es inverso al orden en que lo recorres.

Definición:

```
Rebanadas = 1
Filas = 2
Columnas = 3

Matriz_A = [[[[' ', 0] for i in range(Columnas)] for j in range(Filas)] for k in range(Rebanadas)]
```

Función Carga:

```
def CargaCubo(M, CR, CF, CC):
    for R in range(CR):
        for F in range(CF):
            for C in range(CC):
                M[R][F][C][0] = input("Ingresar texto: ")
                M[R][F][C][1] = input("Ingresar Numero: ")
            print("\a")
```

Función Muestra:

```
def MuestraCubo(M, CR, CF, CC):
    for R in range(CR):
        for F in range(CF):
            for C in range(CC):
                print(M[R][F][C][0], end=" ")
                print(M[R][F][C][1], end=" ")
            print("\a")
        print("\a")
```

Programa Principal:

```
CargaCubo(Matriz_A, Rebanadas, Filas, Columnas)
MuestraCubo(Matriz_A, Rebanadas, Filas, Columnas)
```

Debes tener presente, que puedes ampliar el tamaño del cubo acorde a tus necesidades, especialmente el numero de Rebanadas del cubo.

A continuación te dejo el ejemplo completo, en el que cada posición de la matriz contiene una cadena (guarda nombre o lo que necesites) y un número (sueldo, cantidad o lo que prefieras). Bájalo de la nube y analízalo, que pronto lo usaremos.

Código: Codigo/10_Listas_Matriz_Estatica_Cubo_000_B_Carga_Recorre_y_Muestra.txt

Nro	Enunciados con Matrices que tienen más de un elemento en cada Posición	Finaliza

49) Se dispone del archivo "**Padron_01.txt**" que debe ser cargado en memoria con el objetivo de realizar algunos informes posteriormente.
Cada registro del archivo mencionado, usa la coma como separador y tiene los siguientes campos:

NroOrden	Sexo	Nombre	Edad	EstadoCivil

DESCRIPCIÓN DE LOS CAMPOS:

NroOrden	Numero de orden que ocupa esa persona en el Padrón.
Sexo	Sexo de la persona que figura en el documento: Hombre/Mujer
Nombre	Nombre de la persona. El que figura en el Documento.
Edad	Edad que tiene la persona al momento en que fue empadronada. Se calculó según la fecha de nacimiento que figura en el documento.
EstadoCivil	Estado civil de la persona. Para el caso de este ejercicio, se considera que toda persona menor de 18 años es Menor de edad y soltera, pero una vez cumplidos los 18 años, ya es Mayor y Puede tener otro estado civil, que se clasifica según la siguiente codificación 0- Soltero 1- Casado 2- Viudo 3- Divorciado.

Desde acá podrás descargar el archivo ya creado y que deberás usar para calcular los totales Solicitados.

Archivo:	Archivos/Padron_01.txt

Pautas Generales:
- ✓ El Archivo deberá ser leído una sola vez para cargarlo en memoria.
- ✓ Una vez se completo la lectura del archivo se podrá comenzar a realizar los cálculos solicitados
- ✓ Una persona podrá votar, si tiene estrictamente 16 años o más años, sin limite de edad

Realizar los siguientes cálculos:
- **a)** Cuantos Votantes Casados son Hombres y cuantas son Mujeres.
- **b)** Cuantos Votantes Solteros son Hombres y cuantas son Mujeres.
- **c)** Cuantos NO Votantes Casados son Hombres y cuantas son Mujeres **(cuidado con esto!)**.
- **d)** Cuantos Votantes que no tienen la edad para Casarse son Hombres y cuantas son Mujeres.
- **e)** Cuantos Menores de edad son Hombres y cuantas son Mujeres.
- **f)** Cual es la cantidad actual de Votantes (no importa el sexo).
- **g)** Dividir el padrón actual en dos partes, uno de Hombres y otro de mujeres, manteniendo la misma estructura en ambos archivos.

Una vez terminado los cálculos:
- **h)** Mostrar Totales en el monitor.
- **i)** Imprimir totales por la impresora (el diseño del formato de impresión queda a cargo del alumno).

Las Tuplas en la Programación Python

Definición de Tupla: Las **tuplas** en Python, son colecciones de elementos, para que tenga una idea mental, son muy parecidas a los vectores o listas, pero:
- Son "**inmutables**", lo que significa que, una vez creadas, no se puede modificar su contenido (el valor contenido en cada posición).
- No podremos quitar o agregar elementos.
- Pueden contener diferentes tipos de datos y..
- Se definen utilizando paréntesis

```
MiTupla = (1, 'Perro', 3.14, 3)
```

Características de una Tupla

A continuación una resumida enumeración de las características, que debes tener siempre en mente al momento de utilizarlas:

a) Inmutabilidad: No puedes cambiar, agregar o eliminar elementos de una tupla existente. Si necesitas modificarla, debes crear una nueva tupla combinando la original con los nuevos elementos.

b) Creación de Nuevas Tuplas: Puedes crear nuevas tuplas al concatenar la tupla existente con otros elementos. Veamos un primer ejemplo, en el que se evidencia la creación de una nueva tupla para agregar elementos nuevos:

```
tupla_original = (1, 2)
nueva_tupla = tupla_original + (3, 4)  # Crea una nueva tupla
```

A continuación veremos otro caso, pero la creación de la nueva tupla no es tan evidente (también lo encontraras en el código ejemplo que puedes bajar de la nube) :

```
MiTupla = ('Perro', 'Gato', 'Loro')
Cantidad = len(MiTupla) # Cantidad de elementos
for i in range(Cantidad):
    elemento = input(f'Ingresa el elemento {i+1}: ')
    MiTupla += (elemento,)  # Atento a la coma
```

Dentro del bucle, cada vez que se ejecuta **"MiTupla += (elemento,)"** , estás creando una nueva tupla que incluye todos los elementos de **MiTupla** más el nuevo elemento. La operación += no modifica la tupla original; en su lugar, asigna la nueva tupla a la misma variable **MiTupla**. Y la tupla vieja quedara (por decirlo de alguna forma) flotando en el espacio de la memoria (ocupando parte de ella) sin que nadie la use. **Pero tranquilo..** Muy pronto vendrá el recolector de basura **(garbage collector)** y limpiará la memoria.

c) Uso de Memoria: Cada vez que creas una nueva tupla, se asigna un nuevo espacio en memoria para ella. Las instancias anteriores (las viejas tuplas) seguirán ocupando memoria hasta que el recolector de basura de Python las elimine, lo cual sucede cuando ya no hay referencias a esas tuplas (por que la variable que la contenía o referenciaba, fue reasignada a otra tupla).

d) Eficiencia: Si planeas construir una colección que requiera muchas modificaciones, es más eficiente usar listas al principio debido a su mutabilidad. Una vez completada la colección, puedes convertirla a una tupla si es necesario.

e) Acceso a Elementos: Puedes acceder a los elementos de una tupla usando índices, como en las listas:

```
PrimerElemento = MiTupla[0]
```

f) Para que se pueden usar las Tuplas: Las tuplas en Python son estructuras de datos que se utilizan para almacenar colecciones de elementos que no deberían cambiar una vez creadas (no se pueden agregar, eliminar ni cambiar sus elementos después de su creación).

Son especialmente útiles para almacenar datos que son constantes o que deben permanecer inalterados.

Ejemplos: Analiza algunos ejemplos, en los que se usan todos los conceptos anteriormente explicados y algo más.

Ejemplo del uso de una Tupla:	Codigo/11_Tuplas_000_De_Todo_01_Primer_Ejemplo_A.txt

L*as tuplas se pueden utilizar como claves en diccionarios*, incluso hasta contener los datos del diccionario, Siendo esto posible por su inmutabilidad (Como luego veremos, hay alternativas para actualizar los datos del diccionario), mientras que las listas no pueden ser utilizadas con ese propósito.

Puedes ver el Uso de Diccionarios (Si no los conoces y debes refrescar tu conocimiento):

Repaso - Guía de Estudio Diccionarios (Parte I):	Diccionarios_Creacion_y_Uso_Parte_01.pdf
Repaso - Guía de Estudio Diccionarios (Parte II):	Diccionarios_Creacion_y_Uso_Parte_02.pdf
1- Uso de Tupla con Diccionario:	Codigo/11_Tuplas_002_Declara_Define_01_Tupla_y_Diccionario_A_Claves.txt
2- Uso de Tupla con Diccionario:	Codigo/11_Tuplas_002_Declara_Define_02_Tupla_y_Diccionario_A_Claves.txt

Cosas Interesantes (Parte VII)

Graficar en Python con Matplotlib: Graficar datos es muy útil para visualizar información de manera clara y comprensible. Python, con la ayuda de la librería "**Matplotlib**", permite crear gráficos de alta calidad con pocos comandos, haciéndolo accesible incluso para quienes están empezando. Esta biblioteca es muy flexible y se usa en muchas áreas: desde ciencia de datos hasta presentaciones visuales en informes.
Tipos de Gráficos que Puedes Crear: Puedes generar una gran variedad de gráficos, entre los más comunes encontrarás:

• Gráficos de líneas:	Ideales para representar datos continuos, como el cambio de temperatura a lo largo del tiempo.
• Gráficos de barras:	Perfectos para comparar categorías de datos, como las ventas de diferentes productos.
• Gráficos de dispersión:	Útiles para visualizar la relación entre dos variables, común en análisis estadísticos.
• Histogramas:	Muestran la distribución de datos agrupándolos en intervalos, ideal para ver la frecuencia de valores en un rango.
• Gráficos circulares:	Para mostrar proporciones de un todo, como el desglose de un presupuesto.

Además, podrás personalizar cada aspecto de los gráficos, desde los colores hasta las etiquetas, logrando presentaciones visuales adaptadas a tus necesidades.

Integración de gráficos en Documentos PDF: Fácilmente puedes agregar o crear documentos PDF con tus gráficos. Esta es una gran ventaja, especialmente cuando necesitas presentar reportes profesionales o material educativo. Con unas pocas líneas de código, puedes generar gráficos y guardarlos como imágenes, luego insertarlos en un PDF para visualización y distribución. Puedes bajar de la Nube un rápido ejemplo.

Bajar ejemplo de la Nube:	Codigo/60_Graficos_000_Ejemplo_Guia_01_Basico_con_PDF_A.txt

Accede a la Guía de trabajo: (el documento se actualizará próximamente)

Bajar Documento de la Nube:	Graficos_01_Usando_Matplotlib.pdf

Librerías Parte 2 - Paquetes y Módulos Creados por el Usuario: Si estas interesado/a en aprender más de Paquetes y Módulos (Usualmente llamadas Librerías), acá te dejo un breve resumen con muchos ejemplos. Espero te sirva.

Bajar Documento: Librerias_Propias_Creacion_Almacenamiento_Uso.pdf

Mínimamente Distingue el orden de las Jerarquías:	img/Jerarquias_Modulos_Python_01.png

Librerías Parte 3 - Paquetes y Módulos que necesitarás Instalar en Tu PC: A lo largo de tus años en el Ciclo Orientados (4to,5 to,6 to y 7mo año) Necesitarás instalar y/o que funcionen algunas librerías (Paquetes y/o módulos) de Python. Acá te dejo el listado de las que usaremos, si hicieran falta otras librerías en el colegio, se agregarán y actualizará este documento. Ahora puedes acceder e instalar todas las librerías necesarias en forma automatizada.

Bajar Documento de la Nube:	Documentos/Librerias_Que_Necesito_en_la_PC.doc

Reconocer si el Número de Teléfono es Válido: Esto se agrega a pedido de alumnos que en su Programa/Sistema quieren reconocer Números de Teléfono. Con este algoritmo, tendrás una idea superficial, dándote el rumbo de lo que debes hacer. Espero les sirva.

Descarga ejemplo:	Codigo/90_Interesante_025_Reconoce_Numero_Telefono_01_Caracteres_Validos_A.txt

Investiga que hay otros algoritmos.

Cosas Interesantes (Parte VII)

Lee una Fecha de Nacimiento y Calcula su Edad: Esto se agrega a pedido de alumnos que en su Programa/Sistema quieren Calcular la Edad y Antigüedad de un Empleado. Con este algoritmo, tendrás una idea superficial, dándote el rumbo de lo que debes hacer. Espero les sirva.

Descarga ejemplo:	Codigo/90_Interesante_030_Fechas_01_Calculo_Edad_Con_Fecha_Nacimiento_A.txt

Investiga que hay otros algoritmos.

Verificación Condicional por Agregación Lógica o simplemente Evaluación Condicional. En términos generales, se refiere a un proceso donde se combinan muchas y variadas condiciones lógicas (verdadero/falso) para determinar un resultado final.

Imagina que tienes una multitud de sentencias "si-entonces" que se evalúan secuencialmente o en paralelo, y su resultado combinado determina la acción a tomar. "Agregación" implica cómo se combinan estas condiciones (por ejemplo, usando operadores AND, OR, XOR). "Verificación Condicional" se refiere al proceso de evaluar estas condiciones y tomar decisiones basadas en los resultados.

Para nuestro caso, y dicho de una manera mas formal, es una técnica usada en programación, que permite evaluar conjuntos de condiciones simples o compuestas, agrupándolas mediante operadores lógicos (and, or, not, etc), combinándolas en estructuras como listas y/o matrices, que nos permitirá ejecutar reglas complejas de manera flexible y escalable.

Cada elemento de nuestra lista (Vector o Matriz) será una condición, que puede ser una expresión booleana, una función, o incluso un conjunto compuesto de reglas.

Esto permite que nuestro código sea capaz de tomar decisiones complejas, que involucran grandes cantidades de parámetros, condiciones o reglas que pueden cambiar a cada instante.

Accede a la Guía de trabajo: (el documento se actualizará próximamente)

Bajar Documento de la Nube:	Verificacion_Condicional_por_Agregacion_Logica.pdf

Nro	Enunciados Interesantes	Finaliza
50)	**Cuenta Repetición de Palabras Contenidas en un Archivo**: Si tienes una archivo de texto que contiene, por ejemplo un libro, o una novela, y quisieras realizar una estadística, para saber, que palabras contiene el escrito y cuantas veces se repite cada palabra. Bueno, para este proceso debes tener conocimientos mínimos de: - Lectura de Archivos - Manejo de vectores o listas - Manejo de diccionarios - Y claro, la necesidad o ganas de hacerlo Acá te dejo el código, que te diviertas. Codigo/35_Diccionarios_200_Ejemplos_01_Cuenta_Palabras_01.py	

Cosas del Mundo REAL que nos Enseñan a Programar.

Esto te va a gustar. Investiga que es la "*Maquina Enigma*", para que se usó, y lo más importante, quien descifró la Maquina Enigma? Que desencadenó este acontecimiento?

"A veces, la imaginación es más importante que el conocimiento.
El conocimiento es limitado, mientras que la imaginación no"
Albert Einstein.

Nro	Encriptación y Desencriptación de Texto (Procesos Básicos)	Finaliza
51)	**Nivel 00 (Concepto)** - Intercambio de Caracteres. Definir una palabra (que usaremos como clave de encriptación) luego recorrerla mostrando caractér a caractér. Codigo/50_Criptografia_00_INICIAL_001_Cadena_00.txt	

Nro	Encriptación y Desencriptación de Texto (Procesos Básicos)	Finaliza
52)	Mostrar cada letra de la palabra Clave y el número de orden o posición que ocupa cada caractér dentro de la palabra clave. Codigo/50_Criptografia_00_INICIAL_001_Cadena_01.txt	
53)	Pedir al usuario la palabra o frase para encriptar. A continuación mostrar caractér a caractér (verticalmente) la letra de la palabra leída, la que será su reemplazo al ser encriptada (si corresponde) o de lo contrario, la letra que no se pudo encriptar. Codigo/50_Criptografia_00_INICIAL_001_Cadena_02.txt	
54)	Repetir el proceso del programa anterior, pero ahora armar la palabra o frase encriptada (completa) y mostrarla al finalizar el proceso. Codigo/50_Criptografia_00_INICIAL_001_Cadena_03.txt	
55)	Te queda realizar el proceso inverso. Cuando el usuario ingresa la frase encriptada, el programa muestra la frase desencriptada.	
56)	Ahora que ya sabes encriptar y desencriptar, te propongo extender o agrandar la clave a una frase o cadena de caracteres al azahar (Quien lo intenta, generalmente NO puede. Solamente proponte hacerlo y hazlo). Si no tienes idea, acá te dejo una forma (aunque tiene un posible error que no siempre se presenta, descúbrelo). Codigo/50_Criptografia_00_INICIAL_002_Cadena_00.txt	
57)	**Clave Morse**: Realiza un programa que sea capas de traducir un texto a clave Morse. Acá te dejo una imagen con la tabla de equivalencias de letras y números a Morse. Recuerda que para iniciar y comprender el código puedes representar el espacio (en texto) como una barra inclinada "/" (en Morse). img/Codigo_Morse_Internacional.png https://youtu.be/dQRkcVWnNEc?si=x-2o_EfdCmkn4NjD	
58)	**Nivel 01 (iniciando)** - Cambio del Código ASSII - Clave Interna Fija Para comenzar, debes conocer que es el Código Assii y el Código UTF-8. Refresca tu conocimiento e investígalos. Con ellos vamos a trabajar en una serie de ejemplos sucesitos. Que te diviertas.	
59)	**Cifrado César**: Recorrer el texto caractér a caractér desplazando a cada caractér 3 unidades el código Assii, Mostrar texto encriptado (cifrado) y a continuación desencriptarlo, verificando que el texto final quede exactamente igual que el texto inicial. img/Cifrado_Cesar.png Codigo/50_Criptografia_01_INICIAL_001_Cadena_01.txt El cifrado César fue nombrado así en honor a Julio César, quien usó un alfabeto con desplazamiento de tres espacios.	
60)	**Nivel 02 (Encriptación Leve)** - Cambio del Código ASSII - Clave Interna Variable. Codigo/50_Criptografia_01_INICIAL_002_Cadena_01.txt	
61)	**Nivel 03 (Fortificando la Encriptación)** - Cambio Código ASSII - Clave Interna y Externa. Codigo/50_Criptografia_01_INICIAL_003_Cadena_01.txt	
62)	Te animas a realizar algún tipo de encriptación, pero esta vez mezclando, insertando o lo que requiera tu método personalizado con los Emojis? Codigo/90_Interesante_010_Caracteres_Unicode_01_emojis_A.txt	
	Hasta acá te acompaño, ya puedes comenzar a estudiar el tema sin ayuda	

Nro	Enunciados con Manejo de Imágenes	Finaliza
63)	Hacer un programa que muestre una imagen (foto) existente en tu PC Codigo/70_Imagenes_00_INICIAL_001_Muesta_Imagen.txt	

Nro	Enunciados con Manejo de Imágenes	Finaliza
64)	Hacer un programa que muestre una imagen (foto) existente en tu PC Codigo/70_Imagenes_00_INICIAL_002_Imagen_Dimencionada_Posicionada.txt	
65)	Mostrar en la misma línea, una foto, y a continuación nombre de la imagen, luego la descripción y para finalizar el precio. :) Codigo/70_Imagenes_00_INICIAL_003_Imagen_y_Descripcion.txt	
66)	En un negocio donde sirven 4 platos de comida, tienen un listado en el que figuran la foto de cada plato, Nombre, Descripción y Precio. Hacer el programa que muestre estos datos. Codigo/70_Imagenes_00_INICIAL_004_Multiples_Imagenes_Listadas.txt	

-0-

01001101 01100001 01111001 00100000 01110100 01101000 01100101 00100000 01000110
01101111 01110010 01100011 01100101 00100000 01100010 01100101 00100000 01110111
01101001 01110100 01101000 00100000 01111001 01101111 01110101