

TAREA 2 – PROGRAMACIÓN 2025-1

# Objetivo

* La siguiente tarea tiene por objetivo evaluar las competencias adquiridas hasta el momento en el curso. Para ello, deberán trabajar en parejas para resolver los problemas planteados en este documento.
* Se busca evaluar el correcto uso de variables auxiliares, control de ﬂujo y arreglos de datos. Por lo tanto, para la resolución de estos problemas, deberán utilizar únicamente estos recursos.

# Formato de Entrega

Los entregables para esta tarea se componen de dos elementos:

* Un archivo de Python (extensión .py) completamente funcional y correctamente documentado.

# Fecha de Entrega

* Esta tarea debe entregarse a través del buzón destinado para ello en la plataforma eFinis antes del lunes 5 de mayo a las 23:00 horas.

# Evaluación

* Solo un integrante por pareja deberá subir la tarea. En caso de que ambos integrantes realicen entregas, solo se corregirá la primera de ellas.
* Cualquier sospecha de copia o uso de herramientas de IA será evaluada con nota mínima.

# Simulador de Comisiones de Ventas (30% del Total)

Una empresa de ventas necesita determinar y reportar las comisiones mensuales de sus vendedores, en función del total de ventas alcanzado durante el mes. La comisión se calculará aplicando las siguientes reglas:

|  |  |
| --- | --- |
| Venta Mensual | Cálculo de Comisión |
| Menores a $500 | No se otorga comisión. |
| Entre $500 (inclusive) y $1000 | 5% del total de ventas. |
| Entre $1000 (inclusive) y $5000 | 8% del total de ventas. |
| Igual a mayores a $5000 | 10% más un bono fijo de $200. |

Para ello, debe implementar un programa en Python que permita:

* + Solicitar al usuario la cantidad de meses a evaluar.
  + Para cada mes, solicitar el total de ventas
  + Calcular la comisión correspondiente para cada mes, de acuerdo con las reglas anteriores.

Por cada mes ingresado, debe mostrar por consola al usuario:

* + El total de ventas.
  + La comisión obtenida.

Una vez ingresada la información de ventas de la totalidad de los meses, debe mostrar un informe al usuario señalando:

* + El total acumulado de comisiones.
  + El promedio de comisión por mes.
  + El número de meses en que la comisión fue mayor a $300.
* *Considere que es necesario validar que la información ingresada por el usuario sean valores numéricos positivos.*
* *Considere que el uso de estructuras de datos como listas, tuplas, conjuntos o diccionarios, no está permitido.*

# Modelamiento de Rebotes (30% del Total)

Cuando una pelota se deja caer verticalmente, llega al suelo con cierta energía. Al chocar, parte de esa energía se pierde y la pelota rebota hasta una altura menor que la inicial. Para modelar de forma simplificada ese fenómeno se usa el coeficiente de restitución r, un número entre 0 y 1 que representa “qué porcentaje” de la altura se recupera:

nueva\_altura = altura\_anterior × r

* + Si r = 1, la pelota rebotaría para siempre a la misma altura (caso ideal).
  + Si r = 0, la pelota no rebota nada (se queda pegada al suelo).
  + En la vida real r suele estar entre 0.4 y 0.9 según el material.

Debes escribir un programa en Python que simule los rebotes de la pelota hasta que la altura sea tan pequeña que dejemos de considerarla (umbral). El programa debe:

# Entrada:

* + Recibir la altura inicial en metros (número ﬂotante positivo).
  + Recibir el coeficiente de restitución (0 < r < 1).

# Procesamiento:

* + Repetir el cálculo de rebotes mientras la nueva altura sea ≥ 0.10 [m].

# Salida:

Al finalizar la simulación, una vez que la altura de rebotes sea menor a 0.10[m] debe mostrar un informe al usuario indicando:

* + **Rebotes.** Número total de impactos contra el suelo (el primer choque cuenta como rebote 1).
  + **Distancia Total.** Suma de todas las distancias recorridas (bajadas + subidas), sin incluir la última subida que no llega al umbral.
  + **Altura máxima.** La mayor altura que la pelota alcanzó rebotando y en cual rebote ocurrió.

Considere formatear los números en la salida:

* + Rebotes es un número entero.
  + Distancia Total y Altura máxima con dos decimales.

# Historia de un algoritmo viajero: Run‑Length Encoding (40% del Total)

Año 1984. Bill Atkinson, ingeniero de Apple, acaba de terminar MacPaint: un programa que permite a los usuarios dibujar con un ratón por primera vez. El disco de 400 KB donde debe caber la aplicación está repleto y Bill necesita un truco para que los iconos y brochas — grandes franjas de píxeles blancos con algunos trazos negros— ocupen mucho menos espacio.

La solución llega de los laboratorios de telecomunicaciones: Run‑Length Encoding (RLE), un método tan simple que cabe en unas pocas líneas de código y tan ingenioso que reduce drásticamente los archivos donde predominan los valores repetidos. Bastaba con escribir “blanco 50 veces” en lugar de “blanco, blanco, blanco, blanco , blanco …” cincuenta veces.

Poco después, las máquinas de fax Group 3 adoptan la misma idea para enviar documentos impresos casi enteramente blancos; la transmisión que antes tardaba minutos ahora se completa en segundos. A principios de los 90, Windows 3.x arranca mostrando un bitmap comprimido con RLE. Y hoy, cuarenta años más tarde, impresoras térmicas, carteles electrónicos y bases de datos columnares siguen confiando en aquel algoritmo minimalista para ahorrar memoria y ancho de banda.

Ahora es tu turno de unirte a esa tradición.

Una startup de señalética digital te contrata para crear un **mini-compresor RLE en Python** que quepa en un microcontrolador con muy poca RAM. Te piden:

## Entrada:

* Solicitar al usuario un texto por consola, que tenga como máximo 100 caracteres.

## Procesamiento:

* Transformar la cadena de texto ingresada por el usuario a su versión comprimida mediante RLE. Esto implica ree

## Salida:

* Mostrar al usuario la cadena original.
* Mostrar al usuario la cadena comprimida.
* Mostrar al usuario la tasa de compresión (cantidad de caracteres finales vs originales).

**Ejemplos de Compresión:**

|  |  |  |
| --- | --- | --- |
| Texto Original | Texto Comprimido | Observación |
| ABCDE | A1B1C1D1E1 | Por cada uno de los caracteres se agrega el carácter y las instancias consecutivas de este. |
| ABBBBCDDDD | A1B4C1D4 |
| AABBBBBBBBBB | A2B10 |

|  |  |  |
| --- | --- | --- |
| A | A1 | Si hay un único carácter, igualmente se informa el  carácter y un “1”. |
| BBBBB | B5 | Si hay cinco caracteres consecutivos, entonces, se  informa el carácter repetido y un “5”. |
| 122333444455555 | 1122334455 | Números son tratados como texto. |
| AAAaaa | A3a3 | Diferencia entre minúsculas y mayúsculas. |
| A BB CCC | A1 5B2 | Los espacios también son comprimidos. |