Retos de programación diarios.

Andrés, un estudiante, necesita almacenar y analizar las puntuaciones diarias que obtiene en un reto de programación durante una semana. Desea una forma simple de registrar cada puntuación diaria y poder calcular fácilmente la puntuación total.

Una estructura de tamaño fijo, donde los elementos se almacenan en orden definido y pueden accederse directamente por su posición, es adecuada para esta tarea.

Andrés le pide ayuda a su hermana que es ingeniera de software y entre ambos logran definir los requerimientos del programa utilizando las siguientes historias de usuario.

Historia de usuario Nro.	1	**Título:**	Registrar y acceder a las puntuaciones diarias
Descripción	COMO:	Estudiante	
	QUIERO:	Registrar mis puntuaciones diarias durante una semana y poder acceder a ellas por día	
	PARA:	Calcular mi progreso y puntuación total	
Criterios de aceptación	▪ Se deben registrar exactamente 7 puntuaciones, una por cada día de la semana. ▪ Las puntuaciones deben ser enteros no negativos. ▪ Debe poder obtenerse la puntuación de un día específico (Día 0 a Día 6). ▪ Debe calcularse y mostrarse la puntuación total de la semana. ▪ Si se ingresan más o menos de 7 puntuaciones, debe imprimirse un mensaje de error.		

Usted es contratado por Andrés para construir un programa en Python que cumpla las funcionalidades requeridas por Andrés teniendo como referencia las historias de usuario presentadas previamente.

Entrada	Una línea con 7 números enteros separados por espacio, representando las puntuaciones del Día 0 al Día 6.
Salida	La suma total de las 7 puntuaciones. Si la entrada no contiene exactamente 7 números, se imprime "ERROR".

Instrucciones para la calificación automática

Antes de enviar la solución del reto, por favor tenga en cuenta los siguientes aspectos:

- Cada caso de prueba debe estar en una línea.
- Cada línea debe contener exactamente 7 valores enteros separados por espacios.
- Debe existir una clase principal llamada RetoArray.
- En esa clase debe existir un método llamado run.

- El código debe leer de la entrada estándar y escribir a la salida estándar. No debe imprimir mensajes adicionales.

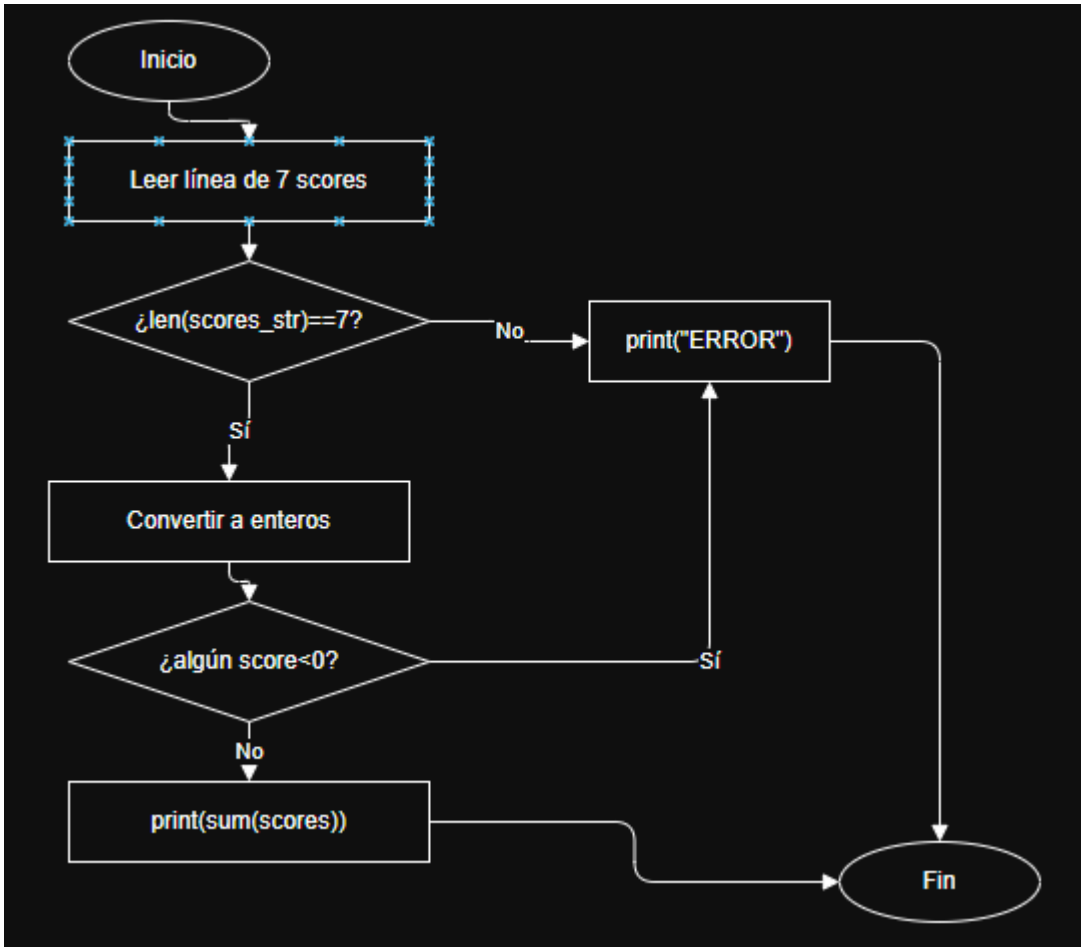
Casos de prueba (visibles)

Entradas de ejemplo	10 15 12 18 20 14 16
Salida de ejemplo	105
Entradas de ejemplo	5 5 5 5 5 5 5
Salida de ejemplo	35

Casos de prueba (ocultos)

Caso prueba	Entrada	Salida esperada
1	0 0 0 0 0 0 0	0
2	100 90 80 70 60 50 40	490
3	1 2 3 4 5 6	ERROR
4	1 2 3 4 5 6 7 8	ERROR

Diseño: diagrama de flujo



Código Python :

```
class RetoArray:

    def run(self):

        try:

            line = input()

            scores_str = line.split()

            if len(scores_str) != 7:

                print("ERROR")

                return



            scores = [int(s) for s in scores_str]
```

```
            if any(score < 0 for score in scores):
                print("ERROR")
                return

            print(sum(scores))

        except ValueError:
            print("ERROR")
        except Exception:
            print("ERROR")

if __name__ == "__main__":
    RetoArray().run()
```

```
if __name__ == "__main__":
    RetoArray().run()
```

```
10 15 12 18 20 14 16
105
```

```
if __name__ == "__main__":
    RetoArray().run()
```

```
0 0 0 0 0 0 0
0
```

```
if __name__ == "__main__":
    RetoArray().run()
```

```
100 90 80 70 60 50 40
490
```

```
if __name__ == "__main__":
    RetoArray().run()
```

```
1 2 3 4 5 6
ERROR
```

```
if __name__ == "__main__":
    RetoArray().run()
```

```
1 2 3 4 5 6 7 8
ERROR
```