# Lista 02: Estructuras de Datos

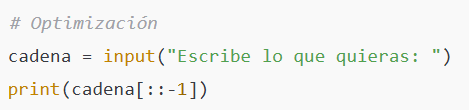
Este alumno demuestra una buena comprensión de las estructuras de datos fundamentales, con un enfoque funcional para resolver problemas diversos. Sin embargo, el código podría beneficiarse de una mayor modularidad, nombres más descriptivos para las variables, y el uso de funciones integradas de Python para simplificar ciertas operaciones.

* **Cosas positivas:**
  + Uso correcto de listas, diccionarios, conjuntos y tuplas.
  + Buen manejo de la entrada y salida de datos con input() y print().
  + Creatividad en los ejercicios, especialmente en la creación de matrices y el manejo de cadenas.
* **Posibles mejoras:**
  + Simplificar el código utilizando funciones integradas de Python para mejorar la legibilidad y la eficiencia.
  + Evitar el uso de nombres de variables poco descriptivos como str, que puede causar conflictos con funciones predefinidas.
  + Añadir más comentarios explicativos para facilitar la comprensión del código.

# Por ejercicios:

### **Ejercicio 1**

**Estado del ejercicio:** Correcto.  
**Errores encontrados:** Ninguno.  
**Código optimizado:** Utiliza slicing para invertir la cadena en lugar de convertirla a lista y viceversa.



**Comentario:** Has resuelto el ejercicio correctamente, pero la solución puede simplificarse considerablemente con slicing. Una cadena de caracteres o string por defecto es una lista. Por lo que no hace falta reconvertirla.

### **Ejercicio 2**

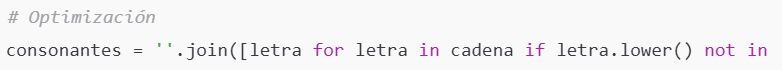
**Estado del ejercicio:** Correcto.  
**Errores encontrados:** Ninguno.  
**Código optimizado:** Usa una comprensión de listas para mezclar las cadenas.



**Comentario:** El uso de zip y comprensiones hace que el código sea más compacto y fácil de entender.

### **Ejercicio 3**

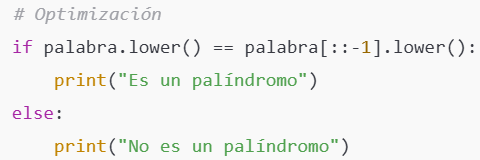
**Estado del ejercicio:** Correcto.  
**Errores encontrados:** Ninguno.  
**Código optimizado:** Usa una comprensión de listas para filtrar consonantes.



**Comentario:** Tu solución es funcional, pero esta optimización reduce la complejidad del código.

### **Ejercicio 4**

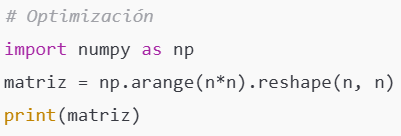
**Estado del ejercicio:** Correcto.  
**Errores encontrados:** Ninguno.  
**Código optimizado:** Usa slicing para comprobar si una palabra es un palíndromo.



**Comentario:** Aunque funcional, esta optimización mejora la legibilidad.

### **Ejercicio 5**

**Estado del ejercicio:** Correcto.  
**Errores encontrados:** Ninguno.  
**Código optimizado:** Usa numpy para generar matrices si es posible.



**Comentario:** Has implementado bien la lógica; sin embargo, el uso de librerías como numpy puede simplificar la creación de matrices.

### **Ejercicio 6**

**Estado del ejercicio:** Correcto.  
**Errores encontrados:** Ninguno.  
**Código optimizado:** Usa sum para calcular la suma de columnas.



**Comentario:** Excelente manejo de matrices. Usar zip mejora la legibilidad y eficiencia.

### **Ejercicio 7**

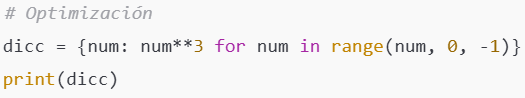
**Estado del ejercicio:** Correcto.  
**Errores encontrados:** Ninguno.  
**Código optimizado:** Usa una comprensión de listas para la transposición.



**Comentario:** La lógica es funcional, pero esta versión reduce la complejidad del código.

### **Ejercicio 8**

**Estado del ejercicio:** Correcto.  
**Errores encontrados:** Ninguno.  
**Código optimizado:** Usa un diccionario directamente para almacenar claves y valores.



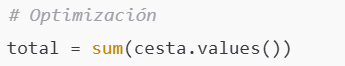
**Comentario:** Muy buen trabajo creando diccionarios dinámicos.

### **Ejercicio 9**

**Estado del ejercicio:** Correcto.  
**Errores encontrados:** Ninguno.  
**Código optimizado:** Usa input() para simplificar la captura de datos.  
**Comentario:** Buen manejo de diccionarios para almacenar información personal.

### **Ejercicio 10**

**Estado del ejercicio:** Correcto.  
**Errores encontrados:** Ninguno.  
**Código optimizado:** Usa sum para calcular el total.



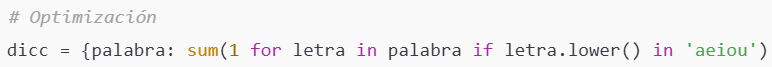
**Comentario:** La lógica es funcional y adecuada.

### **Ejercicio 11**

**Estado del ejercicio:** Correcto.  
**Errores encontrados:** Ninguno.  
**Código optimizado:** Usa mean de numpy para calcular la media.  
**Comentario:** Excelente implementación de diccionarios con listas anidadas.

### **Ejercicio 12**

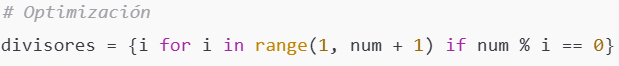
**Estado del ejercicio:** Correcto.  
**Errores encontrados:** Ninguno.  
**Código optimizado:** Usa un generador en lugar de un bucle para contar vocales.

****

**Comentario:** Muy bien manejado el conteo de vocales por palabra. Esta versión reduce la cantidad de líneas de código.

### **Ejercicio 13**

**Estado del ejercicio:** Correcto.  
**Errores encontrados:** Ninguno.  
**Código optimizado:** Usa una comprensión de listas para generar divisores.



**Comentario:** Excelente lógica. El uso de conjuntos mejora la eficiencia al evitar duplicados.

### **Ejercicio 14**

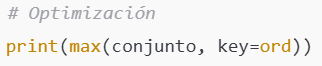
**Estado del ejercicio:** Correcto.  
**Errores encontrados:** Ninguno.  
**Código optimizado:** Usa min para encontrar el valor más bajo.



**Comentario:** Tu enfoque funciona, pero las funciones integradas como min son más rápidas y legibles.

### **Ejercicio 15**

**Estado del ejercicio:** Correcto.  
**Errores encontrados:** Ninguno.  
**Código optimizado:** Usa max directamente para encontrar el carácter con mayor valor Unicode.



**Comentario:** Bien resuelto, aunque esta optimización mejora la claridad.

### **Ejercicio 16**

**Estado del ejercicio:** Correcto.  
**Errores encontrados:** Ninguno.  
**Código optimizado:** No es necesario; el uso de conjuntos y control de errores está bien implementado.  
**Comentario:** Excelente lógica para trabajar con los primeros caracteres de las palabras.

### **Ejercicio 17**

**Estado del ejercicio:** Correcto.  
**Errores encontrados:** Ninguno.  
**Código optimizado:** No es necesario, pero podrías usar una función lambda para determinar si es par o impar.  
**Comentario:** Buen manejo de bucles y condiciones.

### **Ejercicio 18**

**Estado del ejercicio:** Correcto.  
**Errores encontrados:** Ninguno.  
**Código optimizado:** Usa una comprensión de listas para construir las tuplas.

****

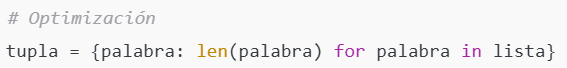
**Comentario:** Buen trabajo. Esta optimización reduce líneas y mantiene la funcionalidad.

### **Ejercicio 19**

**Estado del ejercicio:** Correcto.  
**Errores encontrados:** Ninguno.  
**Código optimizado:** Usa el desempaquetado de listas de forma directa.  
**Comentario:** Excelente implementación de tuplas y desempaquetado.

### **Ejercicio 20**

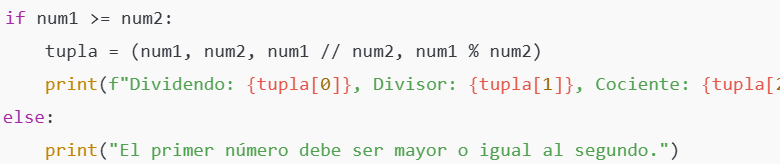
**Estado del ejercicio:** Correcto.  
**Errores encontrados:** Ninguno.  
**Código optimizado:** Usa una comprensión de diccionarios para construir el resultado.



**Comentario:** Buena lógica y uso de diccionarios.

### **Ejercicio 21**

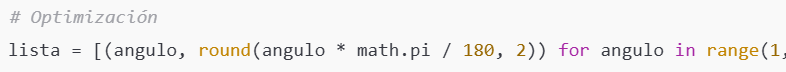
**Estado del ejercicio:** Incorrecto.  
**Errores encontrados:** No valida correctamente la relación entre los números antes de realizar las operaciones.  
**Código optimizado:**

****

**Comentario:** Recuerda validar correctamente los datos antes de realizar operaciones matemáticas.

### **Ejercicio 22**

**Estado del ejercicio:** Correcto.  
**Errores encontrados:** Ninguno.  
**Código optimizado:** Usa una comprensión de listas para construir las tuplas con grados y radianes.

****

**Comentario:** La lógica es funcional, pero esta optimización simplifica el código.