Guía de Ejercicios

Recomendaciones al realizar las guías

- Prestá atención al leer el enunciado. En particular:
  - Si se pide una función que *devuelva* o *calcule* un valor, la función debe hacer uso de `return`.
  - Si se pide una función que *imprima* un valor, la función debe tener un `print`.
  - Si se pide una función que *pida* o *pregunte* algo al usuario, la función debe tener un `input`.
  - A menos que se diga específicamente "pedirle al usuario", no es necesario que el programa contenga `input`. En todo caso, hacer que la función reciba el o los datos por parámetro.
- Cada ejercicio puede tener muchas soluciones posibles. Una vez que encuentres una solución, en lugar de pasar al siguiente ejercicio, pensá si se te ocurre una solución cuyo código sea más simple o incluso mejor en términos de calidad.
- Es muy importante que el código sea lo más claro y legible posible.
  - En particular, nombres de funciones y variables deben ser descriptivos.
  - También prestá atención a los espacios en blanco y a la indentación.
- No documentes en exceso, pero tampoco ahorres documentación necesaria.
- Probá siempre que el código cumpla con lo solicitado.

> **Discord**
>
> La materia usa Discord como plataforma adicional para la resolución de los ejercicios de las guías.
> Tengan a bien leer con atención el mensaje de bienvenida y las reglas de convivencia. Pueden ingresar al servidor a través del siguiente link.

Guía 1: Introducción a la Algoritmia y la Programación

> **Recomendación**
>
> En esta guía nos dedicaremos a introducirnos en los conceptos de programación y algoritmo. Para los primeros ejercicios, te recomendamos ver este video para recordar cómo entiende la computadora nuestras instrucciones.

1.1. Ejercicios de descomposición

1. Se tiene que explicar a una máquina exactamente cómo servir un vaso de jugo (de los que vienen en cartón) de la heladera. Recordando la definición de algoritmo, hacer una descripción paso a paso de lo que se tiene que hacer y usar para lograr el objetivo. Pista: No vas a necesitar nada de código en este ejercicio, sólo nombrar los pasos.

2. Se tiene que explicar a una máquina exactamente cómo hacer una tostada con queso, pensá qué ingredientes se necesitan con sus cantidades, cómo tiene que ser el espacio de trabajo y los elementos que va a necesitar usar. Recordando la definición de algoritmo, hacer una descripción paso a paso de lo que se tiene que hacer y usar para hacer una tostada con queso. Pista: No vas a necesitar nada de código en este ejercicio, sólo nombrar los pasos.

3. Se te pide que organices una colecta de alimentos no perecederos por la Ciudad de Buenos Aires. Contamos con algunos automóviles y camionetas de voluntarios, un listado de donaciones, listado de los alimentos a donar, la disponibilidad horaria y la dirección en la cual se dejan los alimentos. La colecta se realiza en un solo día. ¿Cómo la organizarías? Pista: No vas a necesitar nada de código en este ejercicio, sólo nombrar los pasos.

4. Tenés que enviar invitaciones personalizadas para tu cumpleaños. Cada invitación tiene que mencionar el nombre de la persona y la cantidad de invitados adicionales con los que puede venir. Contamos con una impresora a la que le das el texto a enviar, un listado con los nombres de los invitados y los invitados adicionales que pueden ellos traer. ¿Cómo redactarías el texto de la invitación? Pista: No vas a necesitar nada de código en este ejercicio, sólo nombrar los pasos.

5. Se te encargó definir qué datos son necesarios para el registro de estudiantes en un curso de inglés. ¿Qué datos crees que deberían ser obligatorios y cuáles opcionales? ¿Y si el curso es de cocina? Pista: No vas a necesitar nada de código en este ejercicio, sólo nombrar los pasos.

6. Contás con un listado de cosas a comprar y tenés que ir a un supermercado que cuenta con distintas góndolas o pasillos. Cada góndola o pasillo puede contar con varios, uno o ninguno de los productos de tu lista. ¿Cuál sería el listado de instrucciones para poder terminar lo más rápido posible? Pista: No vas a necesitar nada de código en este ejercicio, sólo nombrar los pasos.

7. Querés explicarle a una persona que nunca usó un cajero automático cómo extraer dinero. Describí paso a paso todas las instrucciones necesarias, considerando que la persona no sabe nada sobre cajeros. ¿Qué información necesita tener antes de empezar? ¿Qué decisiones debe tomar durante el proceso? Pista: No vas a necesitar nada de código en este ejercicio, sólo nombrar los pasos.

8. Una biblioteca quiere organizar sus libros en estantes. Cada libro tiene: título, autor, género (novela, ciencia, historia, etc.) y año de publicación. Los estantes tienen capacidad

limitada. Describí un algoritmo para decidir en qué estante va cada libro, considerando que queremos que sea fácil encontrarlos después. Pista: No vas a necesitar nada de código en este ejercicio, sólo nombrar los pasos.

9. Tenés que explicarle a un robot cómo ordenar una pila de cartas numeradas del 1 al 10 que están desordenadas. El robot solo puede hacer estas acciones: mirar la carta de arriba de la pila, tomar una carta, dejar una carta en una de tres pilas auxiliares, y comparar dos cartas para ver cuál es mayor. Describí los pasos para que el robot ordene las cartas de menor a mayor. Pista: No vas a necesitar nada de código en este ejercicio, sólo nombrar los pasos.

10. Una aplicación de delivery necesita asignar pedidos a repartidores. Cada pedido tiene: dirección de entrega, hora límite de entrega y tipo de producto (comida caliente, bebidas, productos frágiles). Cada repartidor tiene: ubicación actual, tipo de vehículo (bicicleta, moto, auto) y pedidos ya asignados. Describí qué criterios usarías para decidir qué repartidor debe tomar cada pedido nuevo. Pista: No vas a necesitar nada de código en este ejercicio, sólo nombrar los pasos.

1.2. Primer programa

11. Con el anexo de Google Colab de la Unidad 1, realizá tu primer programa: hacé que se imprima por pantalla un `"¡Hola mundo!"`.

Guía 2: Tipos de Datos, Expresiones y Funciones

2.1. Ejercicios de Programación

1. Guardar el texto “Hola, Mundo!” en una variable e imprimirla por pantalla.
2. Guardar los números 1, 2 y 3 en tres variables distintas e imprimirlos por pantalla.
3. a. Guardar los números 1, 2 y 3 en tres variables distintas y luego sumarlos e imprimir el resultado por pantalla.

   b. Repetir con las distintas operaciones disponibles que se vieron en la unidad 2: resta, multiplicación, división, división entera, resto, potencia; combinando los números entre sí.
4. Crear un programa que le solicite al usuario:

   a. Su nombre y lo imprima por pantalla.
   b. Su edad y la imprima por pantalla.
   c. Su edad, le sume 1, y la imprima por pantalla.

5. Crear un programa que le solicite al usuario un número, y que imprima el resto obtenido de dividirlo por 2.
   ¿Qué operador vimos para obtener el resto?

6. Escribir un programa que le pida al usuario su año de nacimiento, y que le diga qué edad tiene en el año actual.

7. Crear un programa que le solicite al usuario 5 enteros y que muestre por pantalla el promedio de ellos. Hacerlo de dos formas:

   a. Primero, usando 5 variables para cada entero.
   b. Después, usando una sola variable para almacenar la suma de los 5 enteros. ¿Cómo se te ocurre que podrías hacer?

A partir de ahora y en adelante, todo lo que se nos pida (incluso si dice "programa") se debe realizar dentro de una o más funciones.

8. Crear una **función** que reciba un número y que devuelva el valor absoluto.

9. Crear una **función** que reciba un número y que devuelva `True` si es par, y `False` si es impar.

10. Crear una **función** que reciba un número y un string, y que devuelva ambos concatenados dentro de un nuevo string.

11. Crear una **función** que reciba dos enteros y que devuelva el resto y el cociente entre ellos.

12. Crear una **función** que le pida al usuario su nombre y apellido, e los imprima con el siguiente formato: "Apellido, Nombre".

13. Hacer una **función** que reciba una palabra y devuelva la cantidad de letras que tiene.

14. a. Hacer una **función** que reciba una palabra y que imprima los primeros 5 caracteres únicamente. Ejemplo: Si se recibe "pensamiento" se debe imprimir "pensa".
    b. Hacer una **función** que reciba una palabra y que imprima sólo los caracteres ubicados en posiciones pares. Ejemplo: Si se recibe "pensamiento" se debe imprimir "pnaino".
    c. Hacer una **función** que reciba una palabra y que imprima la palabra dada vuelta. Ejemplo: Si se recibe "materia" se debe imprimir "airetam".

15. Hacer una **funcion** que reciba una palabra, le borre todas las letras "a" e imprima el resultado por pantalla. Pista: usar una función predefinida de Python. Ejemplo: Si se recibe "casa" se debe imprimir "cs".

16. Analizar qué tipo de dato (o error) se obtiene al hacer las siguientes operaciones:

    a. `5 / 2`
    b. `5 // 2`
    c. `5 % 2`
    d. `5 ** 2`

e. `5.0 / 2`
f. `5.0 // 2`
g. `5.0% 2`
h. `5.0 ** 2`
i. `5 / 2.0`
j. `5 // 2.0`
k. `5% 2.0`
l. `5 ** 2.0`
m. `5.0 / 2.0`
n. `5.0 // 2.0`
ñ. `5.0% 2.0`
o. `5.0 ** 2.0`
p. `"Hola" * 2`
q. `"Hola" + 2`
r. `"Hola" + "2"`
s.
```
x = "Hola"
x += " mundo"
```

17. a. Escribir una función que convierta un valor dado en grado Celsius, a Fahrenheit. Recordar que la fórmula para la conversión es: `F = 9/5 * C + 32`.
    b. Escribir una función que convierta un valor dado en grados Fahrenheit, a Celsius. Usar la misma fórmula anterior.

18. Escribir una función que calcule el área de un triángulo recibiendo como parámetros su base y su altura.

19. Siendo el cálculo de la norma de un vector v en R^3:

$$||v|| = \sqrt{v_1^2 + v_2^2 + v_3^2}$$

    Escribir una función que calcule la norma de un vector en R3 recibiendo como parámetros las 3 componentes v_1, v_2 y v_3 del mismo.

20. **Desafío** (no obligatorio): Calcular el área de un rectángulo (alineado con los ejes x e y), dadas sus coordenadas x_1, x_2, y_1 e y_2. Nota: considerar que tiene que funcionar para cualquier cuadrado, independientemente de los cuadrantes en los que tenga sus vértices.

2.2. Evaluación de Código

> **Nota**
>
> En estos ejercicios se presenta código con errores, malas prácticas de programación o problemas de eficiencia. El objetivo es identificar los errores, explicar por qué fallan y proponer una corrección. Podés probar el código en tu entorno y, opcionalmente, consultar con herramientas de IA para verificar tu análisis.

1. El siguiente código intenta calcular el promedio de tres números, pero tiene varios problemas. Identificá cuál es y corregilo.

```
def prom(a, b, c):
    suma = a + b + c
    promedio = suma / 3
```

2. El siguiente código intenta convertir una temperatura de Celsius a Kelvin, pero tiene varios problemas. Identificá el error y corregilo.

```
def celsius_a_kelvin(celsius):
    celsius = input("Ingrese la temperatura en Celsius: ")
    kelvin = celsius + 273.15
    print kelvin
```

3. El siguiente código intenta concatenar un nombre con un número entero para crear un usuario, pero tiene varios problemas. Identificá por qué falla y proponé una solución.

```
funcion CrearUsuario(nombre, numero):
    usuario = nombre + numero
    return usuario
```

Guía 3: Estructuras de Control

3.1. Decisiones

1. Escribir una función que, dado un número entero n, calcule si es impar o no.
2. Escribir una implementación propia de la función *abs*, que devuelva el valor absoluto de cualquier valor que reciba. Ejemplo: `mi_abs(5)` devuelve `5` y `mi_abs(-5)` devuelve `5`. Pista: No se puede usar la función predefinida `abs`.
3. Escribir una función que reciba un número y devuelva `True` si es entero y `False` si no lo es. Pista: no se puede usar la función `isinstance`.

4. Escribir una función para determinar si una letra recibida es vocal o no. La misma debe devolver un valor booleano. Luego, escribir una función para determinar si una letra es consonante o no.

    a. Resolver *sin* el uso de `in` ni `not in`.
    b. Resolver *usando* `in` y `not in`.
    c. Resolver para que la función identifique tanto mayúsculas como minúsculas. Pista: investigar los métodos `lower` y `upper` de string.

> Tip: `in` y `not in`
>
> ¿Conocés el uso de `in`?
> Para saber si un elemento está en una lista o en un string, podemos usar `in` y `not in`. Por ejemplo:

```
'a' in 'hola'
```

```
True
```

```
'w' in 'hola'
```

```
False
```

```
'w' not in 'hola'
```

```
True
```

```
'casa' in ['cama', 'mesa', 'silla']
```

```
False
```

5. Escribir funciones que resuelvan los siguientes problemas:

    a. Dado un año, que devuelva si es bisiesto. Nota: un año es bisiesto si es un número divisible por 4, pero no si es divisible por 100, excepto que también sea divisible por 400.
    b. Dado un mes y un año, que devuelva la cantidad de días correspondientes.
    c. Pedirle al usuario su día y mes de cumpleaños. El programa debe imprimir un mensaje indicando a qué signo corresponde el usuario.

```
Aries: 21 de marzo al 20 de abril.
Tauro: 21 de abril al 20 de mayo.
Géminis: 21 de mayo al 21 de junio.
```

```
Cáncer: 22 de junio al 23 de julio.
Leo: 24 de julio al 23 de agosto.
Virgo: 24 de agosto al 23 de septiembre.
Libra: 24 de septiembre al 22 de octubre.
Escorpio: 23 de octubre al 22 de noviembre.
Sagitario: 23 de noviembre al 21 de diciembre.
Capricornio: 22 de diciembre al 20 de enero.
Acuario: 21 de enero al 19 de febrero.
Piscis: 20 de febrero al 20 de marzo.
```

6. Piedra, papel o tijera: escribir un programa de "Piedra, papel o tijera" tal que sea imposible que el usuario gane. El usuario debe ingresar **R** (piedra), **P** (papel), o **T** (tijera) y la computadora debe siempre ganarle. Las ejecuciones son individuales: el usuario sólo ingresa una sola vez su jugada, el programa le gana, y la ejecución termina.

   Ejemplo 1 de Ejecución:

   ```
   ¡Piedra (R), papel (P) o tijera (T)!
   Ingrese jugada: R
   ¡Papel! ¡Gané!
   ```

   Ejemplo 2 de Ejecución:

   ```
   ¡Piedra (R), papel (P) o tijera (T)!
   Ingrese jugada: P
   ¡Tijera! ¡Gané!
   ```

   Ejemplo 3 de Ejecución:

   ```
   ¡Piedra (R), papel (P) o tijera (T)!
   Ingrese jugada: T
   ¡Piedra! ¡Gané!
   ```

   Ejemplo 4 de Ejecución:

   ```
   ¡Piedra (R), papel (P) o tijera (T)!
   Ingrese jugada: M
   Esa jugada no está disponible.
   ```

7. Suponiendo que el primer día del año fue lunes, escribir una función que reciba un número con el día del año (de 1 a 366) y devuelva el día de la semana que le toca. Por ejemplo: si se recibe '3', debe devolver "miércoles", y si se recibe '9', debe devolver "martes".

3.2. Ciclos

1. Escribir función que:

   a. Imprima por pantalla todos los números entre 10 y 20.
   b. Salude a todas las personas de esta lista `[Flaminia, Serena, Agustina, Priscila, Sol, Agostina, Iara, Lu]` con el mensaje `"Hola <nombre>! Vamos a aprender a programar"`.
   c. Le pida al usuario que ingrese 5 números y le muestre la suma total de todos ellos.
   d. Imprima por pantalla todos los números entre 100 y 199 que sean divisibles por 7.
   e. Reciba dos números, y recorra todos los números entre ellos, imprimiendo en pantalla si es par o impar. Por ejemplo, recibiendo 1 y 3, debe imprimir:

```
1 es impar
2 es par
3 es impar
```

2. Se quiere hacer un programa para enseñar a los niños las tablas de multiplicar del 1 al 10. Crear una función que reciba un número e imprima por pantalla la tabla de multiplicar de ese número. Ejemplo:

   ```
   mostrar_tablas_para(1)
   ```

   debe imprimir:

   ```
   1 x 1 = 1
   1 x 2 = 2
   1 x 3 = 3
   1 x 4 = 4
   1 x 5 = 5
   1 x 6 = 6
   1 x 7 = 7
   1 x 8 = 8
   1 x 9 = 9
   1 x 10 = 10
   ```

   ```
   mostrar_tablas_para(-2)
   ```

   debe imprimir:

   ```
   Error: El número debe ser positivo y estar entre 1 y 10
   ```

3. Crear una función que cante el feliz cumpleaños. Dado un entero, debe imprimir 'Que los cumplas feliz' en distintas líneas por esa cantidad de veces.

4. a. Necesitamos escribir un programa de cobro en el supermercado. La función debe recibir un número entero que representa el monto a pagar y debe permitir al usuario que ingrese valores, hasta que el pago se haya realizado en su totalidad. Además, le debe ir indicando cuánto le queda por pagar. El programa no da vuelto.

   Ejemplo:

   ```
   Su total a pagar es: 500
   Ingrese el monto a pagar: 100
   Pendientes: 400. Ingrese el monto a pagar: 200
   Pendientes: 200. Ingrese el monto a pagar: 200
   Pendientes: 0. Gracias por su compra.
   ```

   b. Hacer que el programa anterior dé vuelto:

   Ejemplo:

   ```
   Su total a pagar es: 500
   Ingrese el monto a pagar: 100
   Pendientes: 400. Ingrese el monto a pagar: 200
   Pendientes: 200. Ingrese el monto a pagar: 300
   Pendientes: 0. Su vuelto es: 100. Gracias por su compra.
   ```

5. Escribir un programa que le pida al usuario que ingrese un número. Para ese número, se imprime la tabla de multiplicar del 1 al 10. Luego, se le vuelve a pedir otro número. Si el usuario ingresa “X”, el programa debe terminar. El usuario debe poder ingresar números indefinidamente hasta que ingrese “X”. Se puede reutilizar la función del ejercicio 9 de esta guía.

   Ejemplo:

   ```
   Hola! Esto es Tablas de Multiplicar
   Ingrese un número o "X" para salir: 1
   1 x 1 = 1
   1 x 2 = 2
   1 x 3 = 3
   1 x 4 = 4
   1 x 5 = 5
   1 x 6 = 6
   1 x 7 = 7
   1 x 8 = 8
   1 x 9 = 9
   1 x 10 = 10
   Ingrese un número o "X" para salir: -2
   Error: El número debe ser positivo y estar entre 1 y 10
   ```

```
Ingrese un número o "X" para salir: X
¡Adios!
```

6. **Manejo de contraseñas**

   a. Escribir un programa que contenga una contraseña inventada, que le pregunte al usuario la contraseña, y no le permita continuar hasta que la haya ingresado correctamente.
   b. Modificar el programa anterior para que solamente permita una cantidad fija de intentos.
   c. Modificar el programa anterior para que sea una función que devuelva si el usuario ingresó o no la contraseña correctamente, mediante un valor booleano (`True` o `False`).

7. a. Hacer una función que reciba un número del 1 al 10, y luego permita al usuario poder adivinar ese número, ingresando valores repetidamente. Para cada ingreso del usuario, el programa debe indicarle si su número es menor o mayor al número a adivinar. Una vez que el usuario ingresa el número correcto, lo felicita y termina.

   b. Repetir permitiendo únicamente 3 intentos.

   c. Repetir generando el número aleatoriamente de la siguiente forma dentro de la función, sin recibirlo por parámetro:

```python
import random
numero_a_adivinar = random.randint(1, 10)
print(numero_a_adivinar)
```

```
1
```

> **Tip: Bibliotecas**
>
> ¿Sabías que Python tiene muchas bibliotecas que podés usar para hacer cosas más complejas? Por ejemplo, la biblioteca `random` tiene funciones para generar números aleatorios. También hay otras bibliotecas como `Pandas` para trabajar con datos, `Matplotlib` para hacer gráficos, `Numpy` para trabajar con matrices, y muchas más. Vamos a estar viendo estas tres en la última unidad de la materia.
> Una biblioteca es un conjunto de funciones que alguien más escribió y que podemos usar en nuestros programas. Para usar una biblioteca, primero tenemos que importarla. Por ejemplo, para usar la biblioteca `random`, tenemos que poner `import random` al principio de nuestro programa (arriba de todo en nuestro archivo). Luego, podemos usar las funciones de la biblioteca, como `random.randint(1, 10)`.

8. a. Queremos modelar una máquina de sacar juguetes. Debemos hacer una función que reciba un número que representa la cantidad de fichas x que necesita la máquina para funcionar. Se debe imprimir un mensaje en pantalla que indique "Ingresá x fichas para comenzar". El usuario deberá ingresar entonces letras "F", que representan a las fichas. Notar que si se ingresa algo distinto a "F", se ignora.

   Se debe seguir solicitando fichas siempre que no se haya alcanzado la cantidad necesaria para funcionar. Cuando se haya alcanzado la cantidad necesaria, se debe imprimir un mensaje que indique "¡A jugar!". Ejemplo:

```
Ingresá 2 fichas para comenzar: F
Ingresá 2 fichas para comenzar: B
Ingresá 2 fichas para comenzar: Hola
Ingresá 2 fichas para comenzar: F
¡A jugar!
```

   b. Modificar el programa anterior para que vaya mostrando la cantidad de fichas que faltan para comenzar a jugar. Ejemplo:

```
Ingresá 2 fichas para comenzar: F
Ingresá 1 fichas para comenzar: B
Ingresá 1 fichas para comenzar: ficha
Ingresá 1 fichas para comenzar: F
¡A jugar!
```

9. Crear una función que calcule si un número es primo o no. Un número es primo cuando solamente es divisible por sí mismo y por 1. Pista: usar el operador módulo `%`.

10. **Desafío** (obligatorio): Crear una función que reciba un número entero e imprima los números primos entre 1 y el número ingresado.

11. **Desafío** (obligatorio):

    a. Crear una función que reciba dos números, y devuelva la suma de todos los números múltiplos de 7 entre esos dos números. Por ejemplo, si recibe 3 y 25, debe devolver 7 + 14 + 21 = 42. Si recibe 3 y 4, debe devolver 0, ya que no hay múltiplos de 7 entre esos dos números.

    b. Repetir calculando el promedio en vez de la suma.

    c. Repetir calculando únicamente el promedio entre los primeros 3 múltiplos de 7 encontrados. Pista: usar `break`.

    d. Repetir calculando únicamente el promedio entre los múltiplos de 7 encontrados que no sean múltiplos de 2. Pista: usar `continue`.

12. **Desafío** (obligatorio):

a. Escribir una función que dada la cantidad de ejercicios de un examen, y el porcentaje de ejercicios bien resueltos necesario para aprobar dicho examen, revise un grupo de exámenes.

Para ello, en cada paso debe preguntarle al usuario la cantidad de ejercicios resueltos por el alumno, o pedirle que ingrese "*" para salir. Debe mostrar por pantalla el porcentaje correspondiente a la cantidad de ejercicios resueltos respecto a la cantidad de ejercicios del examen y una leyenda que indique si aprobó o no.

b. Adicional al punto anterior: imprimir un mensaje informándole al usuario la cantidad de ejercicios y el % de aprobación.
Validar que el usuario siempre ingrese números positivos y menor o iguales a la cantidad de ejercicios del examen, o "*". De lo contrario, mostrar un mensaje de error y volver a pedirle el dato al usuario.

3.3. Evaluación de Código

> **Nota**
>
> En estos ejercicios se presenta código con errores, malas prácticas de programación o problemas de eficiencia. El objetivo es identificar los errores, explicar por qué fallan y proponer una corrección. Podés probar el código en tu entorno y, opcionalmente, consultar con herramientas de IA para verificar tu análisis.

1. El siguiente código intenta verificar si un número está entre 1 y 10 (inclusive), pero no funciona correctamente para todos los casos y tiene varios problemas. Identificá lo que está mal y corregilo.

```python
def esta_en_rango_entre_numero_1_y_10_con_extremos_inclusive(numero):
    if numero > 1:
        if numero < 10:
            return True
        else:
            return False
    else:
        return False

# Pruebas:
print(esta_en_rango(1))   # Debería dar True
print(esta_en_rango(5))   # Debería dar True
print(esta_en_rango(10))  # Debería dar True
```

2. El siguiente código intenta sumar los números del 1 al N usando un ciclo while, pero ocurre un error. Identificá los posibles problemas y corregilo.

```
def suma_hasta(n):
    suma = 0
    i = 1
    while i <= n:
        suma = suma + i
    return suma
```

3. El siguiente código intenta contar cuántos números pares hay en un rango, pero el resultado siempre es incorrecto. Identificá los posibles problemas y corregilo.

```
def contar(i, f):
    contador = 0
    for i in range(i, f):
        if i % 2 == 0:
            contador = 1
        return contador
```

Guía 4: Tipos de Estructuras de Datos

4.1. Cadenas de caracteres

1. Escribir funciones que dada una cadena y un caracter:

   a. Inserte el caracter entre cada letra de la cadena. Ejemplo: `'separar'` y `'-'` debería devolver `'s-e-p-a-r-a-r'`.

   b. Reemplace todos los espacios por el caracter. Ejemplo: `'mi archivo de texto.txt'` y `'_'` debería devolver `'mi_archivo_de_texto.txt'`.

   c. Reemplace todos los dígitos de la cadena por el caracter. Ejemplo: `'su clave es: 1540'` y `'*'` debería devolver `'su clave es: ****'`.

   d. Inserte el caracter cada 3 dígitos en la cadena. Ejemplo: `'2552552550'` y `'.'` debería devolver `'255.255.255.0'`

   e. Modificar todas las anteriores para que, adicionalmente, reciba un parámetro que indique la cantidad máxima de reemplazos o inserciones a realizar. Ejemplo: `'su clave es: 1540'`, `'*'` y 3 debería devolver `'su clave es: ***0'`.

2. Escribir una función que reciba una cadena que contiene un largo número entero y devuelva una cadena con el número y las separaciones de miles. Por ejemplo, si recibe `1234567890`, debe devolver `1.234.567.890`. Cuidado: no es lo mismo `123.456.789.0` que `1.234.567.890`. Tienen que ser separaciones de miles y quedar un número válido.

3. Escribir funciones que dada una cadena de caracteres:

   a. Devuelva la primera letra de cada palabra. Ejemplo: si se recibe `Ciclo Básico Común` se debe devolver `CBC`.
   b. Indique si se trata de un palíndromo. Por ejemplo, `anita lava la tina` es un palíndromo (se lee igual de izquierda a derecha que de derecha a izquierda).

4. Escribir funciones que dadas dos cadenas de caracteres:

   a. Indique si la segunda cadena es subcadena de la primera. Por ejemplo, `'compu'` es subcadena de `'computacional'`.
   b. Devuelva la que sea anterior en orden alfábetico. Por ejemplo, si recibe `'kde'` y `'gnome'` debe devolver `'gnome'`.

5. Escribir una función que, dada una cadena de caracteres, devuelva una lista con cada uno de los caracteres que la componen en mayúscula. Ejemplo: `'Hola'` debe devolver `['H', 'O', 'L', 'A']`. Restricción: no se permite el uso de ciclos for/while. Pista: Buscá en el apunte cómo usar `map`.

6. Escribir una función que, dada una cadena de caracteres, devuelva una tupla con cada uno de los caracteres que no es una vocal. Ejemplo: `'Algoritmos'` debe devolver `('l', 'g', 'r', 't', 'm', 's')`. Restricción: no se permite el uso de ciclos for/while.

7. Escribir una función que, dada una cadena de caracteres, devuelva el número de índice de posición del último caracter. Por ejemplo, para la cadena `'Hola'` debe devolver `3`. Restricción: no se permite el uso de ciclos for/while.

8. **Desafío** (obligatorio):

   a. Se quiere implementar un buscador dentro de un editor de texto, que permita encontrar todas las ocurrencias de una palabra en un texto. Para ello, se debe implementar una función que reciba como parámetro una palabra y un texto, y que devuelva la primer aparición de la palabra en el texto. Pista: `index` arrojará un error si la subcadena no se encuentra. ¿Qué otro método tenemos disponible para buscar subcadenas?

   b. Modificar la función anterior para que devuelva una lista con las posiciones de inicio de cada ocurrencia de la palabra dentro del texto. Ejemplo: si se busca `'al'` en `'calcule el precio al valor actual'`, debe devolver `[1, 18, 22, 31]`. Pista: del método usado en el punto anterior, ¿conocemos algún parámetro adicional que le podamos pasar?

c. Modificar la función anterior para que devuelva la cantidad de ocurrencias encontradas. Ejemplo: si se busca `'al'` en `'calcule el precio al valor actual'`, debe devolver `4`. Restricción: No se puede usar el método `len`.

9. **Desafío** (no obligatorio): Escribir una función que reciba dos cadenas de caracteres y devuelva una lista con todos los caracteres que no tienen en común. Ejemplo: `'Python'` y `'Hola'` debería devolver el conjunto de letras `['P', 'y', 't', 'l', 'a', 'n']`, indiferentemente del orden y de si está en mayúscula o minúscula. Nota: para que un caracter esté en la lista, no es necesario que esté en la misma posición.

4.2. Rangos, Tuplas y Listas

1. Usar un rango para:

   a. Imprimir los números del 10 al 50 inclusive, saltando de 5 en 5.
   b. Imprimir los números del 40 al 20 en orden decreciente, saltando de 2 en 2.
   c. Crear una lista con los números del 4 al 10. Luego, acceder con el *índice* a los elementos que contienen a los números 4, 6 y 9 e impimirlos por pantalla. Pista: recordar que los índices comienzan en 0.

2. Escribir una función que reciba:

   a. Una lista y devuelva `True` si su longitud es par y `False` si su longitud es impar.

   b. Una lista de números cualesquiera y devuelva el elemento máximo y el mínimo.
   c. Una lista de números y devuelva otra lista con los mismos números ordenados de menor a mayor. Por ejemplo, si recibe `[5, 10, 7, 3]` debe devolver `[3, 5, 7, 10]`.

3. a. Escribir una función que reciba una lista de nombres y un número, que representa el cupo. La función debe devolver en una lista a los nombres que no pudieron entrar al curso por falta de cupo. Ejemplo: `chequear_cupo(['Agustina', 'Iara', 'Priscila', 'Sol', 'Lucía'], 3)` debe devolver `['Sol', 'Lucía']`.

   b. Modificar la función anterior para que devuelva únicamente a la última persona de la lista de la gente que pudo entrar. Ejemplo: `chequear_cupo(['Agustina', 'Iara', 'Priscila', 'Sol', 'Lucía'], 3)` debe devolver `'Priscila'`, porque es la última que tuvo cupo.

4. Dada la lista de tuplas `[("Argentina", 3), ("España",1), ("Uruguay", 2), ("Francia",2)]`, donde cada tupla contiene un país y la cantidad de mundiales que ganaron:

   a. Hacer una función que reciba la lista por parámetro e imprima la información de cada país con el siguiente formato:

`País: <nombre> - Copas: <cantidad>`

Si y sólo si el país es "Argentina", se debe imprimir el nombre con 3 estrellas: `"Argentina "`. Usar el operador abreviado `+=`.

b. Hacer una función que reciba la lista por parámetro y devuelva la cantidad de mundiales que ganaron entre todos los países. Ejemplo: `contar_mundiales([("Argentina", 3), ("España",1), ("Uruguay", 2), ("Francia",2)])` debe devolver `8`.

c. Hacer una función que reciba la lista por parámetro y la devuelva, ordenada por cantidad de copas ganadas.

d. Hacer una función que reciba la lista por parámetro y devuelva en una tupla: una lista con los países que tienen más de una copa ganada, y otra lista con valores booleanos que nos diga si la cantidad de copas es par o impar. Pista: ¿Cómo podemos usar `filter`? ¿Y `map`?
Ejemplo: `[("Argentina", 3), ("España",1), ("Uruguay", 2), ("Francia",2)]` devuelve:
`([("Argentina", 3), ("Uruguay", 2), ("Francia",2)] , [False, True, True])`

5. Escribir una función que reciba dos fichas de dominó y determine si *encajan* o no entre sí.

a. Resolver teniendo en cuenta que las fichas se reciben con formato de tuplas. Ejemplo: `(3,4)` y `(5,4)`.
b. Resolver teniendo en cuenta que las fichas se reciben con formato de string. Ejemplo: `'3-4'` y `'5-4'`.

6. Escribir una función que reciba dos vectores y devuelva su prod_escalar. El prod_escalar se calcula como: Siendo $v1 = (v1_1, v1_2, ..., v1_n)$ y $v2 = (v2_1, v2_2, ..., v2_n)$, entonces

$$v1 \cdot v2 = (v1_1 \cdot v2_1) + (v1_2 \cdot v2_2) + ... + (v1_n \cdot v2_n)$$

Si los vectores no tienen las mismas dimensiones, la función debe devolver `None`.

7. a. Escribir una función que reciba una tupla, un índice, y un nuevo valor. La función debe modificar la tupla, cambiando el valor en la posición dada por el índice, por el nuevo valor pasado como parámetro. Devolver la tupla modificada.

b. Repetir el ejercicio anterior, pero con una lista.

c. Repetir ambos si ahora, en vez de recibir un índice, se recibe el valor a eliminar. Si no se contiene al valor, se devuelve la estructura tal cual se recibió.

8. Escribir una función que reciba una lista y un número n. Para dicho número n, debe imprimir los últimos n elementos de la lista en orden inverso, y luego devolver la lista sin ellos. Ejemplo: Si se recibe `[1, 2, 3, 4, 5]` y `n = 2`, debe imprimir `5`, `4` y devolver `[1, 2, 3]`.

9. Escribir una función que reciba una lista de números y devuelva la misma lista en orden inverso.

10. Escribir una función que dado un valor n, devuelva una lista con los números del 1 a n.

11. Escribir una función que reciba una matriz y una tupla (fila, columna), y devuelva el valor ubicado en esa posición de la matriz. Ejemplo: si se recibe la matriz `[[1, 2], [3, 4]]` y la tupla `(0, 1)`, debe devolver `2`.

12. Se tiene una lista de supermercado escrita como string con productos separados por coma: `"pan, arroz, pescado, jugo, fideos,..."`.

    a. Escribir una función que reciba la cadena de caracteres de los productos de supermercado y devuelva una lista con cada uno de los productos por separado: `['pan', 'arroz', 'pescado', 'jugo', 'fideos', ...]`.

    b. Se tiene además otra cadena de caracteres con los precios de cada producto: `"100, 50, 200, 80, 30,..."`. Escribir una función que reciba ambas cadenas y devuelva una lista con tuplas de (producto, precio): `[('pan', 100), ('arroz', 50), ('pescado', 200), ('jugo', 80), ('fideos', 30), ...]`.

    c. Para la función del punto anterior, escribir otra función que reciba la lista de tuplas y devuelva el precio total de la lista de compras.

13. Se quiere crear una lista de supermercado, solicitándole al usuario productos hasta que ingrese el valor 'X'. La función debe devolver los productos en un string, separados por comas. Ejemplo: si se ingresa 'pan', 'arroz', 'pescado', 'X', debe devolver `"pan, arroz, pescado"`.

14. Hacer una función que reciba una lista de palabras, las ordene en orden alfabético y luego las una en un string separadas por espacios. Ejemplo: si recibe `['hola', 'como', 'estas']`, debe devolver `"como estas hola"`.

15. **Desafio** (obligatorio): Escribir una función que reciba un tamaño y devuelva una matriz con 1 en la diagonal principal y 0 en el resto. Ejemplo: si recibe 4, debe devolver la matriz identidad de tamaño 4x4.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

16. **Desafio** (obligatorio): Escribir una función que reciba una matriz y devuelva su transpuesta. Ejemplo: si recibe la matriz `[[1, 2, 3], [4, 5, 6]]`, debe devolver `[[1, 4], [2, 5], [3, 6]]`.

    Si se recibe:

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

Se debe devolver:

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

17. **Desafio** (no obligatorio): Agenda Simplificada
Escribir una función que reciba una cadena a buscar y una lista de tuplas `(nombre_completo, telefono)`, y busque dentro de la lista todas las entradas que contengan en el nombre completo la cadena recibida (puede ser el nombre, el apellido o sólo una parte de cualquiera de ellos). Debe devolver una lista con todas las tuplas encontradas.

18. **Desafio** (no obligatorio): Sistema de facturación simplificado.
Se cuenta con una lista ordenada de productos con tuplas de (identificador, descripción, precio), y una lista de los productos a facturar, con tuplas de (identificador, cantidad).

    Se desea generar una factura que incluya la cantidad, la descripción, el precio unitario y el precio total de cada prod_comprado, y al final imprima el total general.

    Escribir una función que reciba ambas listas e imprima por pantalla la factura solicitada.

19. **Super Desafio** (no obligatorio): Batalla Naval

    Se tiene una matriz de 10x10 que representa un tablero. Cada celda contiene un 0 si está vacía, o un 1 si hay un barco (consideramos que en este caso, sólo hay barcos unitarios que ocupan un espacio).

    La posición de los barcos se representa con tuplas de (fila, columna). Por ejemplo, si se tiene un barco en la fila 1, columna 3, se representa con la tupla (1, 3).

    Escribir una función que cree un tablero con 10 barcos ubicados aleatoriamente (usar la biblioteca `random`), y que permita al usuario intentar adivinar dónde están.

    El usuario luego ingresa una posición, y la máquina indica si había un barco en esa posición (mostrando un mensaje por pantalla "¡Hundido!") o no ("¡Agua!").

    El usuario gana cuando hunde todos los barcos del tablero. Si se equivoca más de 5 veces, pierde.

    > **!** Batalla Naval: Modo Supervivencia
    >
    > ¿Te animás a que el juego sea un ida y vuelta? Es decir, que el usuario también pueda poner barcos y la máquina intente adivinar dónde están. Una posibilidad es que el usuario tenga su propio tablero en un papel, y una vez cada uno, la máquina

y el usuario elijan una posición para atacar.
Te dejamos unos tips:

- Las posiciones son limitadas por el tablero 10x10
- Las posiciones no deberían repetirse

¿Se te ocurre una forma fácil de generar y guardar todas las posiciones posibles del tablero, e ir sacando de a una para que no se repitan? ¿Quién pensás que ganaría, la máquina o el usuario? En este caso, el usuario y la máquina tienen intentos ilimitados intercalados hasta que alguno de los dos gane.

20. Se tiene una base de datos con nombres de libros de la siguiente forma `["La Noche de la Usina", "La Pregunta de sus Ojos", "Ser Feliz era Esto",...]`, y se quiere saber cuántos libros repetidos tienen. Escribir una función que reciba la base de datos y devuelva, para cada uno de los títulos, cuántos ejemplares hay. La lista no tiene un tamaño fijo, y puede contener muchos títulos repetidos.
Pista: tenés que usar un diccionario.

4.3. Diccionarios

1. Escribir una función que reciba una lista de tuplas, y que devuelva un diccionario en donde las claves sean los primeros elementos de las tuplas, y los valores una lista con los segundos. Por ejemplo:

```
l = [('Hola', 'don Pepito'), ('Hola', 'don Jose'), ('Buenos', 'días')]
print(tuplas_a_diccionario(l))
```

```
{'Hola': ['don Pepito', 'don Jose'], 'Buenos': ['días']}
```

2. Escriba una función que reciba una cadena y devuelva:

   a. Un diccionario con la cantidad de apariciones de cada palabra en la cadena. Por ejemplo, si recibe `"Que lindo dia que hace hoy"` debe devolver: `{'que': 2, 'lindo': 1, 'dia': 1, 'hace': 1, 'hoy': 1}`.
   b. Un diccionario con la cantidad de apariciones de cada caracter en la cadena.

3. Escribir una función que reciba una cantidad de iteraciones N.

   a. Se deberá simular una persona que tira un dado N veces, y se deberá devolver un diccionario con la cantidad de apariciones de cada valor en el dado. Nota: para simular una tirada, usar `import random`y `random.randint(1, 6)`.

b. Repetir el punto anterior, si ahora en vez de tirar 1 dado, tira 2. Se debe devolver un diccionario con la cantidad de apariciones de cada valor de la suma de ambos dados.

4. Se tiene una agenda implementada como diccionario, que guarda nombres de personas y sus números de teléfono. Escribir un programa que le pida al usuario que ingrese nombres.

   a. Si el nombre se encuentra en la agenda, debe mostrar el teléfono.
   b. Si el nombre no se encuentra, debe permitir ingresar el teléfono correspondiente.

   En ambos casos, El usuario puede utilizar la palabra "EXIT" para dejar de ingresar nombres.

5. Escribir una función que reciba un texto y para cada caracter presente en el texto, devuelva la palabra más larga en la que se encuentra ese caracter.

6. Nos contratan para hacer un nuevo sistema de FIUBA para almacenar información de sus estudiantes:

nombre	apellido	dni	carrera
Violeta	Perez	42000000	Informática
Carla	Guanca	42001001	Mecánica
Manuela	Gomez	42002002	Química

   a. Crear un diccionario que sirva para representar a cada persona. Debe contener las claves `nombre`, `apellido`, `dni` y `carrera`. Los diccionarios se deben guardan en una lista llamada `estudiantes`.

   b. Agregar al diccionario creado un nuevo elemento, que debe ser otro diccionario y represente las notas obtenidas en la carrera. La clave debe ser el `codigo` y el valor la `nota` (del 1 al 10) obtenida.

   c. Crear código que agregue para la estudiante Violeta Perez la nota 7 en la materia Algoritmos y Programación III (7507), y la nota 4 en la materia Análisis Matemático II (6103).

   d. Teniendo la lista de estudiantes, buscar en la lista la persona con mayor cantidad de notas e imprimirla por pantalla.

7. En un vivero se guardan las plantas en una lista de diccionarios con la siguiente información: especie, luz directa (si/no), precio. Se necesita un sistema que guarde las plantas a medida que van llegando. Hacer una función que reciba la lista de diccionarios de plantas, y los datos de la planta nueva, y agregue esa planta a la lista de diccionarios.

8. Escribir una función que reciba una lista de diccionarios y una clave, y devuelva una lista con los valores correspondientes a esa clave.

9. Se tiene un ticket de supermercado en forma de diccionario con los siguientes datos:

- Nombre del Producto
- Precio por Unidad
- Cantidad

Se pide hacer una función que reciba el ticket y devuelva el monto a pagar total.

10. Rosita tiene una lista de diccionarios donde guarda todas las películas que vió. La información para cada una es: el nombre de la película, año en que salió, y la puntuación que le puso del 1 al 10. Hacer una función que reciba el diccionario y devuelva una nueva lista de diccionarios donde sólo estén las películas que tienen puntaje mayor a 7.

    a. Resolver sin usar `filter`
    b. Resolver usando `filter`.

11. La profesora Llamell guarda las notas del parcial de Pensamiento Computacional en una lista de diccionarios. Cada diccionario tiene la siguiente información: nombre, apellido, intento, nota.

    Los intentos pueden ser 1 (si es la primera vez que rinde el parcial) o 2 (si está en el recuperatorio).

    a. Se pide hacer una función que dada esta lista de diccionarios, se devuelva el promedio de las notas en la primera oportunidad de los alumnos.

    b. Generalizar la función anterior, para que también reciba el número de intento y se pueda devolver el promedio de cualquiera de los dos intentos.

12. En una fábrica se tiene una base de datos donde se guardan todos los códigos de los productos que se fabrican como claves de un diccionario. Los valores de cada clave son nuevos diccionarios, con la siguiente información: fecha de vencimiento (mes,año), si pasó el chequeo de calidad o no.

    Se pude hacer una función que reciba esta lista de diccionarios, y elimine a todos los productos que no pasaron el chequeo de calidad. Devolver en una tuple todos los productos eliminados en formato {codigo: diccionario del producto}.

13. Se quiere guardar información de un grupo de maratonistas. Se necesita guardar su nombre, DNI y todas las maratones que corrió. Para esto último, se guardan: nombre de cada una, año, puesto y el tiempo que tardaron en correrlas (en minutos).

    a. Crear un diccionario de ejemplo que represente esta situación.
    b. Teniendo esta lista de diccionarios, ordenarlos alfabéticamente por el nombre de los maratonistas.
    c. Teniendo esta lista de diccionarios, ordenar las maratones en tiempo ascendente según el tiempo que tardaron en correrlas.

14. **Desafío** (obligatorio): Laura tiene una lista de diccionarios donde guarda el valor de todas las reviews laborales anuales que le hicieron. La información de cada una es año, seniority en ese momento (trainee, junior, semisenior, senior), el sueldo en ese momento y el valor del bono de performance que le dieron. La semana pasada le avisaron que por políticas de la empresa, los bonos ahora deben calcularse como un porcentaje de su sueldo.

    Laura quiere entonces actualizar sus diccionarios, para que en vez de guardar el monto exacto del bono, guarde el porcentaje que le corresponde. Ejemplo: si en el 2019 su sueldo era de $1.000.000 y el bono que le dieron era de $40.000, el bono fue del 4 % del sueldo.

    a. Hacer una función que reciba la lista de diccionarios, y para cada una de las reviews, modifique el valor del bono por el porcentaje correspondiente.

    b. Hacer una función que reciba la lista de diccionarios ya modificada y devuelva los años en los que Laura tuvo un bono mayor al 50 % de su sueldo. Restricción: usar `filter` y `map`.

15. **Desafío** (obligatorio): Los estudiantes de la materia Pensamiento Computacional quieren crear un programa que les facilite la cursada. Uno de los problemas es que no se tiene fácil acceso a los enunciados de los ejercicios, porque la guía es larga y hay que scrollear mucho. En el programa, la guía de ejercicios se guarda en un diccionario, donde cada clave es el número de guía y cada valor una lista con los enunciados de los ejercicios, cada uno en su posición correspondiente (la posición 0 de la lista sólo guarda None). Se quiere que el usuario que está usando el programa pueda acceder por pantalla a un enunciado puntual de una guía con sólo pedirlo (si existe). Para el problema mencionado, hacer una función o más que lo resuelva. Usar los temas visto en la materia hasta el momento, en la forma que se considere mejor y siguiendo las buenas prácticas y convenciones enseñadas.

16. **Desafío** (no obligatorio): **Donarg** (https://www.donarg.com.ar/) es un proyecto que nació con estudiantes de FIUBA con el fin de optimizar procesos tanto para donantes de sangre como para hospitales y servicios de hemoterapia. Formado por estudiantes y graduados universitarios comprometidos, fue galardonado con el primer puesto en la FIUBATON 2020 “Desafío Cuarentena” del FIUBA Consulting Club, destacándose entre más de 100 proyectos.

    Donarg necesita un sistema que permita filtrar una base de datos de posibles donantes de sangre, quedándose con los que cumplen los requisitos.

    La base contiene los siguientes datos de cada posible donante:

    - Nombre
    - Apellido
    - Edad
    - Peso
    - Fecha de la última donación. Puede ser ‘None’ si nunca donó. Formato: (dia,mes,año)

- Fecha del último tatuaje. Puede ser 'None' si no tiene tatuajes. Formato: (dia,mes,año)
- Tipo de sangre. Puede ser '0+', '0-', 'A+', 'A-', 'B+', 'B-', 'AB+', 'AB-'

Los requisitos son:

- Tener entre 16 y 65 años
- Pesar más de 50 kilos
- Que hayan pasado 2 meses desde la última donación
- Que hayan pasado 6 meses desde el último tatuaje

a. Se pide hacer una función que reciba una lista de diccionarios con la información de cada posible donante, y devuelva una lista con los que cumplen los requisitos.

b. Se pide hacer una función que priorice a los donantes que tienen sangre tipo 0 (positivo y negativo) por sobre todos los A, B y AB (positivos y negativos); ya que son los que más se necesitan. La función debe recibir la lista de diccionarios con la información de cada posible donante ya filtrada por requisitos, y devolver una nueva lista ordenados de mayor a menor prioridad.

c. Se pide hacer una función que reciba la lista de diccionarios con la información de cada posible donante ya filtrada por requisitos y ordenada por prioridad, que se quede con los que son 0+ y 0-, y los ordene por órden alfabético de apellido.

   Si querés saber más sobre el proyecto, podés visitar su página web:
   https://www.donarg.com.ar/
   o sacar turno para donar sangre en
   https://www.donarg.com.ar/dondedono.

4.4. Evaluación de Código

> **Nota**
>
> En estos ejercicios se presenta código con errores, malas prácticas de programación o problemas de eficiencia. El objetivo es identificar los errores, explicar por qué fallan y proponer una corrección. Podés probar el código en tu entorno y, opcionalmente, consultar con herramientas de IA para verificar tu análisis.

1. El siguiente código intenta obtener el último elemento de una lista, pero contiene errores. Identificá por qué falla y corregilo.

```
def obtener_ultimo():
    ultimo = lista[len(lista)]
    return ultimo
```

2. El siguiente código intenta modificar una tupla, pero contiene errores. Explicá qué ocurre y proponé una solución alternativa.

```
def cambiar_segundo_elemento(tupla, nuevo_valor):
    tupla[1] = nuevo_valor
    return tupla
```

3. El siguiente código intenta contar la cantidad de veces que aparece cada letra en una palabra usando un diccionario, pero presenta varios problemas. Identificá cuál es y corregilo.

```
contador = {
    'a': 0, 'b': 0, 'c': 0, 'd': 0, 'e': 0, 'f': 0, 'g': 0, 'h': 0, 'i': 0, 'j': 0, 'k':
}

def contar_letras(palabra):
    for letra in palabra:
        contador[letra] = contador[letra] + 1
    return contador
```

4. En una biblioteca se tiene un catálogo de libros donde cada código de libro es una clave de un diccionario. Los valores son diccionarios con la información del libro: título, autor y si está disponible para préstamo o no.

   El siguiente código intenta eliminar del catálogo todos los libros que no están disponibles y devolver en una tupla los libros eliminados. Identificá los errores y corregilo.

```
# Ejemplo de catálogo de libros
catalogo = {
    "LIB001": {"titulo": "Cien años de soledad", "autor": "García Márquez", "disponible"
    "LIB002": {"titulo": "El túnel", "autor": "Ernesto Sabato", "disponible": False},
    "LIB003": {"titulo": "Rayuela", "autor": "Julio Cortázar", "disponible": True},
    "LIB004": {"titulo": "Ficciones", "autor": "Jorge Luis Borges", "disponible": False}
}
```

```
def eliminar_no_disponibles_devolver_eliminados_de_diccionario(libros):
    eliminados = {}
    for codigo in libros:
        if libros[codigo]["disponible"] == False and libros[codigo]["disponible"] != Tru
            eliminados.append(libros[codigo])
            del libros[codigo]
    return eliminados.filter( { libros[codigo]["disponible"] != True })
```

Guía 5: Entrada y Salida

5.1. Archivos

1. Escribir una función llamada `contar` (o `count`) que dado un archivo retorne la cantidad de filas que tiene.
2. Escribir una funcion llamada `imprimir`(o `cat`) que dado un archivo imprima por pantalla todo el contenido.
    a. Agregar un parametro a la funcion llamado `ignorar_vacias` (o `blank`) que en caso de ser `True`, no se impriman las lineas en blanco.
3. Escribir una función llamada `ver_encabezado` (o `head`) que dado un archivo y un número N retorne en una lista las primeras N lineas del archivo.
4. Escribir una función llamada `ver_final` (o `tail`) que dado un archivo y un número N retorne en una lista las últimas N lineas del archivo.
5. Escribir una función llamada `crear` (o `touch`) que dado el nombre de un archivo lo cree. Si el archivo exixte borra todo el contenido.
6. Escribir una función, llamada `wc`, que dado un archivo de texto, lo procese e imprima por pantalla cuántas líneas, cuantas palabras y cuántos caracteres contiene el archivo.
7. Escribir una función, llamada grep, que reciba una cadena y un archivo de texto, e imprima las líneas del archivo que contienen la cadena recibida.
8. Se tiene un archivo con una pregunta, llamado `pregunta.txt`. Se pide leerlo y mostrar el contenido en consola. Luego, pedirle al usuario una respuesta, y guardarla en un archivo llamado `respuesta.txt`.
9. Sea un diccionario cuyas claves y valores son cadenas.
    a. Escribir una función `guardar_diccionario` que reciba el diccionario y un nombre de archivo, y guarde el contenido del diccionario en el archivo, en formato CSV, con un par clave-valor por línea.
    b. Escribir una función `cargar_diccionario` que reciba el nombre de un archivo con el formato mencionado en el punto anterior, y devuelva el diccionario original.
10. **Desafío** (obligatorio): Distribución de Carga
    Se tiene un archivo de texto muy grande con un libro completo. Como el archivo es muy pesado para procesarlo de una vez, se quiere dividir en archivos más pequeños.

    Procesar el archivo: Se desea que el archivo original se separe en archivos resultantes de no más de 50 líneas cada uno, además no tienen que quedar líneas en blanco. Por ejemplo si el archivo original tiene 200 líneas de las cuales 20 están en blanco, el resultado del

procesamiento tienen que ser 4 archivos de no más de 50 líneas cada uno (sin líneas en blanco). El tamaño del archivo original puede variar en cada ejecución. De necesitarlo, se pueden usar las funciones pedidas en los ejercicios anteriores.

11. La señora Rowling sospecha que su vecino le robó algunos manuscritos y los publicó como propios con algunas modificaciones. Se desea procesar una lista de archivos de texto y guardar en un archivo llamado `reporte_plagio.txt` la siguiente información por cada archivo procesado:

```
archivo;palabras;apariciones
archivo1.txt;765030;547
```

Un ejemplo de uso podría ser:

```
archivos = ["archivo1.txt", "archivo2.txt"]
procesar_archivos(archivos, "harry")
```

En `procesar_archivos` se tiene que analizar los archivos y dejar el resultado en reporte_plagio.txt

12. Hacer una función que reciba un archivo y dos palabras: una que se quiere reemplazar, y otra por la que se quiere reemplazar. La función debe modificar el archivo reemplazando todas las apariciones de la primera palabra por la segunda.

13. Se tiene un archivo de texto que contiene un registro de tareas pendientes. Cada línea tiene el formato: `[ ] Descripción de la tarea` para tareas pendientes o `[x] Descripción de la tarea` para tareas completadas.

Ejemplo de archivo `tareas.txt`:

```
[ ] Estudiar para el parcial
[x] Entregar el TP
[ ] Ir al supermercado
[x] Llamar al médico
```

a. Hacer una función que lea el archivo y devuelva dos listas: una con las tareas pendientes y otra con las tareas completadas.
b. Hacer una función que le pida al usuario la descripción de una tarea y la agregue al final del archivo como pendiente. Debe seguir pidiéndole al usuario descripciones de tareas hasta que este ingrese como tarea "X".
c. Hacer una función que reciba un número de línea y marque esa tarea como completada.

14. En un cine tienen dos archivos .txt, uno con salas y otro con nombres de películas. Se sabe que en la sala de una fila del archivo se va a transmitir la película de la misma fila del archivo de películas. Se pide leer los dos archivos, y crear un nuevo archivo csv que tenga el nuevo formato sala;pelicula

Por ejemplo si se tienen los siguientes archivos:

(salas.txt)

```
D2
F1
E4
```

(peliculas.txt)

```
Megamente
The Menu
Shrek
```

El nuevo archivo deberá quedar:
(funciones.csv)

```
D2;Megamente
F1;The Menu
E4;Shrek
```

15. Se tiene un archivo con información de ventas de productos en un supermercado. El archivo tiene el siguiente formato:

```
producto;precio;cantidad;fecha
arroz;50;100;2021-01-01
fideos;40;200;2021-01-01
arroz;50;100;2021-01-02
fideos;40;200;2021-01-02
arroz;50;100;2021-01-03
fideos;40;200;2021-01-03
```

    Se pide hacer una función que reciba el nombre del archivo y un producto, y devuelva el precio promedio de ese prod_en el archivo.

16. Se tiene una lista de archivos y se quiere imprimir las primeras 5 líneas de cada uno. Si el archivo no existe, se lo debe crear e imprimir el mensaje: 'el archivo {nombre} no existe. Se crea y deja vacío.'.

17. **Desafío** (obligatorio): Los docentes de Pensamiento Computacional saben que los ejercicios de las primeras guías son cortitos de resolver, entonces les dicen a los estudiantes que "si el ejercicio tiene más de 5 líneas, probablemente haya que revisarlo". Los estudiantes se lo toman muy literal, y quieren buscar la forma de chequear si un ejercicio que resolvieron tiene más de 5 líneas, pero sin contar a la firma de la función.
    El ejercicio puede estar guardado en un archivo o en un string todo junto, lo que resulte más cómodo para su resolución. Además, quieren la posibilidad de poder pasar varios ejercicios juntos para ahorrar tiempo, y saber en cuáles se pasaron de las 5 líneas o si

algun archivo (en caso de elegir esa forma de almacenamiento) no existe. Para el problema mencionado, hacer una función o más que lo resuelva. Usar los temas visto en la materia hasta el momento, en la forma que se considere mejor y siguiendo las buenas prácticas y convenciones enseñadas.

18. **Desafío** (obligatorio): Los estudiantes del CBC pensaron que podía ser buena idea guardar los apuntes de todas las materias en un mismo archivo, para no marearse con tantos documentos diferentes. Pero terminó siendo un dolor de cabeza y es muy dificil encontrar las notas de cada materia. Para poder acceder facil a esa informacion, decidieron extraer del archivo original la informacion de las notas. Sabemos que a cada materia se le puso de título un hashtag, por ejemplo "#analisisMatematico", y que la fila que contiene un título no contiene nada más. Lo que se quiere hacer es obtener un archivo de texto con el nombre de cada sección/materia, donde el contenido del mismo es el número de línea donde comienza la sección y el número de línea donde termina, separados por `;`. Para el problema mencionado, hacer una función o más que lo resuelva. Usar los temas visto en la materia hasta el momento, en la forma que se considere mejor y siguiendo las buenas prácticas y convenciones enseñadas.

    Por ejemplo, el archivo `analisisMatematico.txt` podría ser:

    ```
    3;97
    ```

5.2. Manejo de Errores

1. Crear una función que asegure la apertura de un archivo. Debe recibir un nombre y si el archivo existe, retorna el contenido archivo. Si no existe, imprime un mensaje descriptivo.
2. Escribir una función que le pida al usuario que ingrese 5 números y le muestre la suma total de todos ellos. Si el valor ingresado no es un número, debe mostrar un mensaje de error y seguir pidiendo números.
3. Escribir un programa que le pida números al usuario e imprima si son pares o impares. Si el usuario ingresa un valor que no es un número, el programa debe imprimir un mensaje de error y seguir pidiendo números. El programa termina cuando el usuario ingresa `"*"`.
4. Un climatólogo tiene archivos donde guarda las temperaturas promedio de la Ciudad de Buenos Aires de cada mes. Cada línea es un número que representa la temperatura promedio del día. Por ejemplo:

    ```
    25.5
    26.0
    24.5
    # etc
    ```

    Como el climatólogo es humano, a veces se equivoca y tipea cosas que no son números. Por ejemplo: `25p3` o `25.5.5`. Se pide hacer una función que reciba el nombre del archivo

y devuelva la temperatura promedio del archivo. Si en algún momento se encuentra con un valor que no es un número, debe ignorarlo y seguir con el resto de los valores.

Al final de la ejecución, debe devolverse el valor promedio y el porcentaje de valores que no pudieron ser procesados respecto del total.

5. Se tiene un archivo de texto donde cada línea contiene dos números separados por un espacio. Se debe leer el archivo y, por cada línea, realizar la división del primer número por el segundo. Los resultados se guardan en un archivo llamado "resultados.txt".

   Se debe tener en cuenta que en el archivo se pueden encontrar filas con errores. En esos casos, en lugar de escribir el resultado, escribir "Error en la línea X: {descripción específica del error}" donde X es el número de línea.

   Ejemplo de archivo de entrada:

   ```
   100 5
   50 0
   abc 10
   30 6
   ```

6. **Desafio**: Para la venta de entradas de un teatro se cuenta con un diccionario que contiene las filas: "A", "B", "C"… y en cada fila contiene una lista con las ubicaciones disponibles en forma de lista `["L","O","O","O","L","L","L"]` donde "L" es libre y "O" es ocupada.

   Escribir una funcion que reciba el diccionario, la fila y la ubicación en ella, y reserve el asiento en caso de que este libre. Considerar que el tamaño de la lista de ubicaciones puede variar por fila. Si la fila o la ubicación no son correctas, mostrar un mensaje descriptivo. Si el asiento ya se encuentra ocupado, mostrar el mensaje: `El asiento ya se encuentra ocupado`.

   Ejemplo

   ```
   sala = {"A":["L","O","O","O","L","L","L"], "B": ["L","O","O","O","L"]}
   reservar_butaca(sala, "A", 0)
   # En este ejemplo la fila "A" deberia quedar ["O","O","O","O","L","L","L"]
   ```

7. **Desafio** (obligatorio): Se tiene esta función, que recibe dos números y devuelve la división entre ellos:

   ```
   def dividir(n1, n2):
       return n1/n2
   ```

   La función puede lanzar una excepción `ZeroDivisionError` si el segundo número es 0 (es decir, la ejecución va a terminar con un error).

a. Probar usar la función de arriba, pasándole 5 y 0 como argumentos. ¿Qué pasa? ¿Cómo podrías modificar la función para evitar que el programa termine con un error? Considerar qué valor podría llegar a devolver la función si ocurre un error.

b. Se pide hacer un programa que le pida al usuario dos números, y muestre el resultado de la división usando/invocando a la función `dividir` definida arriba. Si el usuario ingresa un 0 como segundo número, se debería mostrar un mensaje de error y retornar `None`.

c. Considerar ahora también que el usuario podría ingresar algo que no es un número. En ese caso, debe mostrar otro mensaje de error diferente, y seguir pidiendo números.

8. **Desafío** (no obligatorio): El kiosko de la facultad quiere automatizar un letrero que tome datos de un programa y le cobre al estudiante.

   Se tienen dos diccionarios, uno con un código y el producto, y otro con el código y el precio de cada producto.

```
opciones = {
    1: "hamburguesas",
    2: "milanesas",
    3: "gaseosa",
    4: "alfajor",
    5: "papas fritas",
    6: "agua"
}

valores = {
    1:1000,
    2:1500,
    3:500,
    4:300,
    5:600,
    6:350
}
```

   Se quiere hacer un programa que muestre la información de la siguiente forma en la pantalla:

```
1. hamburguesas - $1000
2. milanesas - $1500
3. gaseosa - $500
4. alfajor - $300
5. papas fritas - $600
6. agua - $350
```

Y le pida al usuario una opción y una cantidad. Luego, debe imprimir el total a pagar.

Se debe considerar que el usuario podría ingresar una opción que no está en el diccionario, o ingresar una opción que no sea un número. El programa debe en esos casos imprimir un mensaje de error que sea descriptivo y terminar su ejecución.

5.3. Evaluación de Código

> **Nota**
>
> En estos ejercicios se presenta código con errores, malas prácticas de programación o problemas de eficiencia. El objetivo es identificar los errores, explicar por qué fallan y proponer una corrección. Podés probar el código en tu entorno y, opcionalmente, consultar con herramientas de IA para verificar tu análisis.

1. El siguiente código intenta leer un archivo y contar las líneas, pero presenta varios problemas. Identificalos y corregilos.

```
def contar_lineas(nombre_archivo):
    archivo = open(nombre_archivo, 'w')
    contenido = archivo.read()
    lineas = contenido.split("\n")
    cantidad = len(lineas)
    return cantidad
```

2. El siguiente código intenta escribir una lista de nombres en un archivo, pero presenta varios problemas. Identificalos y corregilos.

```
def guardar(lista_nombres, nombre_archivo):
    for nombre in lista_nombres:
        archivo = open(nombre_archivo, 'w')
        archivo.write(nombre + '\n')
        archivo.close()
```

3. El siguiente código intenta leer un archivo y devolver su primera línea. Si el archivo no existe, debe devolver None. Identificá los problemas y corregilos.

```
def leer_primera_linea(nombre_archivo):
    try:
        archivo = open(nombre_archivo, 'r')
        contenido = archivo.read()
        lineas = contenido.split("\n")
        return lineas[0]
    except:
```

```
        print("Ocurrió un error")
        return None
```

4. El siguiente codigo es un buscador de palabras en un archivo que sabemos que tiene más de un millón de líneas y es muy pesado. La función recibe una ruta de archivo y una palabra, y devuelve una lista con los números de las líneas en donde se encontró la palabra. Si el archivo no existe, se debe devolver una lista vacía. Identificar los problemas y corregirlo.

```
def encontrar_apariciones(nombre_archivo, palabra):
    try:
        archivo = open(nombre_archivo, 'r')
        contenido = archivo.read()
        lineas = contenido.split("\n")

        indice_linea = 1
        for linea in lineas:
            apariciones = []

            if linea in palabra:
                apariciones.append(palabra)
            else:
                palabra = None

        return lineas
    except:
        print("Ocurrió un error")
        return None
```

Guía 6: Bibliotecas de Python

1. Pandas

> **Nota**
>
> Para estos ejercicios, recomendamos usar Google Colab (ver apunte) y tener una celda por ejercicio.

1.1. Operaciones Básicas con DataFrames

Nos contratan para hacer un nuevo sistema de FIUBA para almacenar información de sus estudiantes.

Usando el siguiente DataFrame:

```
import pandas as pd
data = {
    'nombre': ['Violeta', 'Carla', 'Manuela', 'Lucia', 'Emilia', 'Mariana', 'Aldo'],
    'apellido': ['Perez', 'Guanca', 'Gomez', 'Capon', 'Duzac', 'Szischik', 'Rastrelli'],
    'dni': [42000000, 42001001, 42002002, 37010020, 40001002, 38090080, 38111222],
    'año_nac': [1997, 1998, 1998, 1993, 2003, 1993, 1994],
    'mail': ['vp@fi.uba.ar', 'cg@fi.uba.ar', None, None, None, 'ms@fi.uba.ar', 'ar@fi.uba.ar
    'carrera': ['Informática', 'Mecánica', 'Química', 'Informática', 'Informática', 'Electró
}

df = pd.DataFrame(data)
df
```

	nombre	apellido	dni	año_nac	mail	carrera
0	Violeta	Perez	42000000	1997	vp@fi.uba.ar	Informática
1	Carla	Guanca	42001001	1998	cg@fi.uba.ar	Mecánica
2	Manuela	Gomez	42002002	1998	NaN	Química
3	Lucia	Capon	37010020	1993	NaN	Informática

	nombre	apellido	dni	año_nac	mail	carrera
4	Emilia	Duzac	40001002	2003	NaN	Informática
5	Mariana	Szischik	38090080	1993	ms@fi.uba.ar	Electrónica
6	Aldo	Rastrelli	38111222	1994	ar@fi.uba.ar	Informática

1. Mostrar el resumen de la información del DataFrame
2. Mostrar su forma
3. Mostrar la lista de los nombres de las columnas
4. Mostrar las primeras 3 filas
5. Mostrar las últimas 3 filas
6. Agregar la nueva columna al DataFrame con la información de la edad de los estudiantes para el año actual y mostrar el DataFrame resultante
7. Mostrar a los estudiantes que tienen edad mayor a 25.
8. Para los estudiantes que tienen edad mayor a 25, mostrar el promedio de edad.
9. Mostrar a la persona que es más joven.
10. Mostrar a la persona que se encuentra en la mitad de la tabla.
11. Mostrar sólo las columnas: “carrera” y “edad”.
12. Mostrar sólo los estudiantes que estudian Informática.
13. Mostrar sólo los estudiantes que estudian informática y tienen menos de 25 años.
14. Reemplazar `Informática` por `Informática o Sistemas` en la columna “carrera”.
15. Mostrar la cantidad de estudiantes por carrera.
16. Renombrar la columna “carrera” por “ingeniería”.
17. Para la primer estudiante, cambiar su carrera a “Electrónica”.
18. Agregar una nueva fila al DataFrame con tus datos.
19. Agregar una nueva columna ‘tiene_mail’, con un valor booleano que indique si la estudiante tiene mail en el sistema.
20. Agregar una nueva columna, con un valor booleano que indique si a la estudiante se le necesita pedir un mail (sólo se le tiene que pedir un mail si no tiene mail asignado). Asumir que todavía no se tiene la columna ‘tiene_mail’.
21. Ordenar el dataFrame por carrera y apellido, en ese orden.

22. Ordenar el dataFrame por carrera y edad. Carrera por forma ascendente, edad por forma descendente.

23. La profesora Llamell guarda las notas del parcial de Pensamiento Computacional en un dataFrame. Cada fila tiene la siguiente información: nombre, apellido, intento, nota. Los intentos pueden ser 1 (si es la primera vez que rinde el parcial) o 2 (si está en el recuperatorio).

    a. Obtener el promedio de las notas en la primera oportunidad de los alumnos.
    Nota: Este ejercicio ya fue resuelto en la Guía 4 de Diccionarios, es el 4.3.11.

```
data = {
    'nombre': ['Violeta', 'Carla', 'Manuela'],
    'apellido': ['Perez', 'Guanca', 'Gomez'],
    'dni': [42000000, 42001001, 42002002],
    'carrera': ['Informática', 'Mecánica', 'Química'],
    'nota': [7, 4, 6],
    'intento': [1, 2, 1]
}

df_notas = pd.DataFrame(data)
df_notas
```

	nombre	apellido	dni	carrera	nota	intento
0	Violeta	Perez	42000000	Informática	7	1
1	Carla	Guanca	42001001	Mecánica	4	2
2	Manuela	Gomez	42002002	Química	6	1

24. Rosita tiene un dataFrame donde guarda todas las películas que vió. La información para cada una es: el nombre de la película, año en que salió, y la puntuación que le puso del 1 al 10. Obtener sólo las películas que tienen puntaje mayor a 7. Nota: este ejercicio ya fue resuelto en la Guía 4 de Diccionarios, es el 4.3.10.

```
data = {
    'nombre': ['Harry Potter', 'El Señor de los Anillos', 'Barbie', 'Rapido y Furioso 18'],
    'año': [2001, 2003, 1972, 2040],
    'puntuacion': [8, 9, 8, 3]
}

df_pelis = pd.DataFrame(data)
df_pelis
```

1.2. Lectura y Escritura de Archivos CSV

1. Se tiene un archivo CSV llamado `futbol.csv` con información de jugadores de fútbol. El archivo tiene las columnas: nombre, equipo, posicion, goles, asistencias.

   a. Leer el archivo usando Pandas y mostrar las primeras 5 filas.
   b. Mostrar la información del DataFrame (tipos de datos, cantidad de filas, etc.).
   c. Guardar sólo los jugadores que tienen más de 10 goles en un nuevo archivo llamado `goleadores.csv`.

2. Se tiene un archivo CSV llamado `compras_supermercado.csv`, donde se describen las compras realizadas por una empresa para el desayuno de sus empleados. El archivo tiene las columnas: fecha, producto, cantidad, precio_unitario.

   a. Leer el archivo y agregar una nueva columna `total` que indique el precio total que se tiene del producto, según la cantidad y el precio unitario.
   b. Guardar el resultado en un nuevo archivo llamado `compras_con_total.csv`.
   c. Calcular el gasto total de la compra e imprimirlo por pantalla.

3. Se tiene un archivo CSV que contiene información sobre el stock de una librería. Un posible ejemplo de este archivo es el siguiente:

   ```
   lapiceras;34512;50;120
   cuadernos;41611;500;130
   sacapuntas;62812;30;210
   ```

   Donde cada línea representa un prod y contiene el nombre del producto, el código de barras, la cantidad en stock y el precio unitario. Hacer una función que reciba la ruta del archivo y le solicite al usuario datos de un nuevo producto y lo agregue al final del archivo. Debe seguir pidiéndole al usuario datos de productos hasta que este ingrese como producto "X". Hacer una función que reciba la ruta del archivo y realice lo pedido. Considerar que el archivo podría no existir. En ese caso, retornar.

4. Se tiene un archivo CSV llamado `peliculas.csv` con las columnas: titulo, año, genero, calificacion.

   a. Leer el archivo y filtrar sólo las películas con calificación mayor a 7.
   b. Calcular el promedio de calificación de todas las peliculas.
   c. Guardar las películas de género "Acción" en un archivo llamado `peliculas_accion.csv`.

5. Se tiene un archivo CSV con datos de temperaturas diarias de una ciudad. Las columnas son: fecha, temperatura_min, temperatura_max.

   a. Leer el archivo y agregar una columna con la temperatura promedio del día.
   b. Encontrar el día con la temperatura máxima más alta y el día con la temperatura mínima más baja. Mostrar todos sus datos.

c. Guardar los días donde la temperatura promedio superó los 25 grados en un archivo llamado `dias_calurosos.csv`.

6. **Desafío** (obligatorio): Una empresa de cine quiere analizar sus datos de venta de entradas por película de un sábado por la noche. Se tiene un archivo `cine.csv` con las columnas: pelicula, sala, fecha, horario, entradas_vendidas, precio_entrada. Las películas no se repiten.

   a. Leer el archivo y calcular los ingresos totales. Mostrarlos en pantalla.
   b. Obtener la película más taquillera (la que más ingresos generó).
   c. Encontrar las 5 peliculas que tuvieron más entradas vendidas, y mostrar su nombre y su sala.
   d. Guardar los datos "pelicula, fecha, horario" por cada película, en un archivo llamado `historial_proyecciones_peliculas.csv`.

7. La empresa yerbatera "La Hoja Verde" realiza controles de calidad diarios en sus lotes de producción. Cada lote es analizado para medir el **porcentaje de palo** (ramas) que contiene, ya que un exceso afecta la calidad del producto. La información se guarda en un archivo CSV con las columnas: **lote**, **fecha**, **porcentaje_palo**, **humedad**.

   Los tres tipos de calidad son:

   - **Premium**: menor a 15 % palo
   - **Estándar**: entre 15 % y 30 % palo
   - **Baja**: mayor a 30 % palo

   Se pide:

   a. Filtrar los lotes por cada tipo de calidad y agregar a cada grupo una columna `calidad` con el valor correspondiente.
   b. Mostrar los 3 lotes con mayor porcentaje de palo.
   c. Mostrar solo los lotes de calidad Premium.

   Ejemplo de archivo `control_calidad_yerba.csv`:

```
lote,fecha,porcentaje_palo,humedad
Lote A,2026-02-19,12.5,9.8
Lote B,2026-02-19,28.3,11.2
Lote C,2026-02-20,8.2,10.5
Lote D,2026-02-20,32.7,8.9
Lote E,2026-02-21,18.4,9.3
Lote F,2026-02-21,35.8,10.1
```

8. La empresa de logística "Rapiflash" necesita analizar el cumplimiento de sus rutas de entrega. Cada vehículo tiene un archivo CSV con su registro de entregas, donde cada línea contiene: **direccion**, **minutos_planificados**, **minutos_reales**. Se considera un **incumplimiento** cuando un vehículo llega a una dirección con más de 5 minutos de diferencia respecto a lo planificado (ya sea adelantado o atrasado).

   Se pide implementar un programa que:

   a. Reciba una lista con las rutas de archivos de varios vehículos.
   b. Para cada archivo, calcular la diferencia entre minutos reales y planificados, y contar los incumplimientos.
   c. Si un archivo no existe, mostrar un mensaje de error y continuar con el siguiente.
   d. Guardar en un archivo `resumen_entregas.csv` el nombre del archivo y la cantidad de incumplimientos.

   Considerar que los archivos podrían no existir. En ese caso, imprimir un mensaje descriptivo y avanzar con el siguiente.

   Ejemplo de archivo `vehiculo_34.csv`:

   ```
   direccion,minutos_planificados,minutos_reales
   Av. San Martín 123,156,158
   Lavalle 456,206,198
   Sarmiento 789,209,209
   Belgrano 321,244,260
   ```

   Ejemplo de uso:

   ```
   archivos = ["vehiculo_34.csv", "vehiculo_12.csv", "vehiculo_78.csv"]
   ```

   Ejemplo de salida `resumen_entregas.csv`:

   ```
   archivo,incumplimientos
   vehiculo_34.csv,2
   vehiculo_12.csv,1
   vehiculo_78.csv,0
   ```

   (En el ejemplo, vehículo 34 tiene 2 incumplimientos: Lavalle con 8 min de diferencia y Belgrano con 16 min)

9. En una fábrica de alimentos detectaron que las planillas diarias de producción tienen datos faltantes, causando problemas con los envíos. La Ingeniera de planta decide analizar qué tan confiables son estos registros.

   Cada día se genera una planilla CSV con las columnas: **operacion**, **codigo**, **unidades**, **tiempo**, **observaciones**. Algunas celdas pueden estar vacías (datos faltantes).

   Se pide implementar un programa que:

a. Calcule el total de unidades producidas (solo operaciones de tipo “produccion”).
b. Cuente los registros válidos (aquellos donde `unidades` y `tiempo` no están vacíos).
c. Cuente las operaciones “ordenadas” (donde `observaciones` es “-”).
d. Calcule el porcentaje de registros inválidos.
e. Determine si los datos son “sucios” (si el porcentaje de inválidos es mayor o igual a 50 %).
f. Guarde un resumen en `reporte_produccion.csv`.

Ejemplo de archivo `05-09-2024.csv`:

```
operacion,codigo,unidades,tiempo,observaciones
produccion,JG30,50,20,-
envio,HP45,,15.05,-
produccion,FRE54,100,,revisión
envio,FRE54,100,105.70,-
produccion,Hy78,40,8,-
```

Ejemplo de salida `reporte_produccion.csv`:

```
concepto,valor
Total unidades producidas,190
Operaciones totales,5
Operaciones válidas,3
Operaciones ordenadas,4
Porcentaje inválidos,40%
Datos sucios,False
```

10. La organización **“Bibliotecas Abiertas”** lleva un registro mensual de voluntarios. Cada archivo CSV mensual contiene: **nombre**, **apellido**, **dni**, **horas_donadas**. El campo `horas_donadas` puede estar vacío si hubo errores de carga.

    Se pide implementar una función que reciba una lista de archivos mensuales y:

    a. Calcule la suma total de horas donadas (solo registros válidos).
    b. Cuente la cantidad total de registros y la cantidad de registros válidos.
    c. Calcule el porcentaje de registros sin errores.
    d. Guarde en `errores.csv` los registros con datos faltantes para revisión manual.
    e. Guarde las estadísticas en `estadisticas_voluntarios.csv`.

    Considerar que algunos archivos podrían no existir. En ese caso, mostrar un mensaje de error y continuar con el siguiente.

    Ejemplo de archivo `08_2025.csv`:

```
nombre,apellido,dni,horas_donadas
Lucía,López,22334455,8.0
Marcos,Pérez,39800222,12.5
Ana,Gómez,18900222,
Laura,Sosa,41111222,10.0
Roberto,Medina,35000888,
Sofía,Luna,29999999,15.8
```

Ejemplo de salida `estadisticas_voluntarios.csv`:

```
concepto,valor
suma_horas,46.3
registros_totales,6
registros_validos,4
porcentaje_sin_errores,66.7%
```

Ejemplo de salida `errores.csv`:

```
nombre,apellido,dni,horas_donadas
Ana,Gómez,18900222,
Roberto,Medina,35000888,
```

2. Numpy

1. Crear un array de 20 números espaciados uniformemente entre 0 y 10. Luego, imprimir:
   a. La dimensión
   b. La forma
   c. El tamaño
2. Crear un array llamado "arr" con números enteros del 0 al 9. Luego:
   a. Cambiar su forma de manera tal que tenga 2 filas y 5 columnas
   b. Insertar el valor 100 en la posición 3.
3. Para el array `x = np.arange(10)`, calcular:
   a. `y = x^2`
   b. `y = 3x + 2`
   c. `y = 1/(x+1)`
   d. `y = log10(x)`
4. Usando el array `y = 1/(x+1)` del ejercicio anterior:

a. Calcular la media de `y`.
b. Calcular la diferencia entre `y` y su media

5. Repetir el ejercicio 4.2.2 de la Guía 4:

   Escribir una función que reciba:

   a. Una lista y devuelva True si su longitud es par y False si su longitud es impar.

   b. Una lista de números cualesquiera y devuelva el elemento máximo y el mínimo.

   c. Una lista de números y devuelva otra lista con los mismos números ordenados de menor a mayor. Por ejemplo, si recibe [5, 10, 7, 3] debe devolver [3, 5, 7, 10].

6. Repetir el ejercicio 4.2.6. de la Guía 4:

   Escribir una función que reciba dos vectores y devuelva su prod_escalar. El prod_escalar se calcula como: Siendo $v1 = (v1_1, v1_2, ..., v1_n)$ y $v2 = (v2_1, v2_2, ..., v2_n)$, entonces

   $$v1 \cdot v2 = (v1_1 \cdot v2_1) + (v1_2 \cdot v2_2) + ... + (v1_n \cdot v2_n)$$

   Si los vectores no tienen las mismas dimensiones, la función debe devolver `None`.

7. Repetir el ejercicio 4.2.11 de la Guía 4:

   Escribir una función que reciba una matriz y una tupla (fila, columna), y devuelva el valor ubicado en esa posición de la matriz. Ejemplo: si se recibe la matriz [[1, 2], [3, 4]] y la tupla (0, 1), debe devolver 2.

8. Repetir el ejercicio 4.2.15 de la Guía 4:

   Desafio (obligatorio): Escribir una función que reciba un tamaño y devuelva una matriz con 1 en la diagonal principal y 0 en el resto. Ejemplo: si recibe 4, debe devolver la matriz identidad de tamaño 4x4.

   $$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

9. Repetir el ejercicio 4.2.16 de la Guía 4:

Desafio (obligatorio): Escribir una función que reciba una matriz y devuelva su transpuesta. Ejemplo: si recibe la matriz `[[1, 2, 3], [4, 5, 6]]`, debe devolver `[[1, 4], [2, 5], [3, 6]]`. Si se recibe:

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

Se debe devolver:

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

10. Una persona quiere cercar el frente de su casa con alambre. Los postes se deben poner a 1 metro de distancia como mínimo entre sí. Hacer una función que reciba un número entero que sea la cantidad de metros a cubrir, y devuelva un array con la posición de los postes necesarios. Por ejemplo, si se reciben `5` metros, debe devolver: `[0. 1.25 2.5 3.75 5.]`

11. Se va a realizar un concierto gratis en el Jardín Japonés, para lo cual se habilitó la reserva de asientos por internet. Por un error, las reservas se hicieron en una lista de 50 asientos, guardando el número de reserva si está ocupado o 0 sino: `[1,2,3,0,7,6,4,0,0,8,9, ...]`.
    Los asientos en el Jardín Japonés se van a organizar en forma de matriz, teniendo 5 filas de 10 asientos cada uno. Se quiere transformar la lista de asientos en una matriz. Hacer una función que reciba la lista de asientos y devuelva la matriz de asientos organizada por filas.

    Ejemplo: Si se recibe `[1,2,3,0,7,6,4,0,0,8,9,10,0,13,11,12]`, se debe devolver: `[[1,2,3,0,7,6,4,0,0,8],[9,10,0,13,11,12]]`.

12. **Desafio** (no obligatorio): Se tiene una matriz de 10x10 que representa un tablero de Batalla Naval. Cada celda contiene un 0 si está vacía, o un 1 si no. Se pide hacer una función que cree un tablero con 10 barcos ubicados aleatoriamente (usar la biblioteca `random`).

    b. Luego, se debe hacer una función que reciba la matriz y permita al usuario intentar adivinar dónde están.
    Pista: se puede extender/modificar el **Super Desafio** (no obligatorio) de Batalla Naval de la Guía 4.

3. Matplotlib

1. Graficar las funciones obtenidas en el ejercicio 3 de Numpy, usando los números del 1 al 100 del eje x. Hacerlo primero en distintos gráficos y luego las 4 en el mismo.

    a. y = x^2
    b. y = 3x + 2
    c. y = 1/(x+1)
    d. y = log10(x)
    e. ¿Hay alguna función que no se esté viendo correctamente en el gráfico? Tratar de entender por qué y graficarla en otro gráfico. Pista: mirar los valores que toma la función en el eje `y`.

2. Usando Numpy: Siendo `x` un conjunto de 1000 valores de 0 a 2 * pi separados de forma uniforme:

    a. Graficar, con grilla, título y referencias apropiadas, la función `seno de x` (`sin(x)`).
    b. Graficar, con grilla, título y referencias apropiadas, la función `coseno de x` (`cos(x)`).
    c. Graficar, con grilla, título y referencias apropiadas, la función `(seno de x) + x/2`.

3. Se tienen los siguientes datos de ventas de un prod_en los últimos 10 años:

```
import matplotlib.pyplot as plt
import numpy as np

años = np.arange(2012, 2022)

ventas = [140, 60, 120, 250, 200, 180, 100, 300, 120, 160]
ganancias = [14000, 45000, 20400, 100000, 50000, 25000, 100000, 30000, 15000, 35000]
productos = ["prod_1", "prod_2", "prod_3", "prod_4", "prod_5", "prod_6", "prod_7", "proc
cant_ventas_productos_2024 = [180, 160, 190, 140, 100, 200, 120, 130, 150, 170]
```

    En gráficos separados:

    a. Graficar las ventas en función de los años usando gráfico de línea y gráfico de barras horizontal. ¿Cuál te parece que es mejor para este caso?
    b. Graficar las ganancias en función de los años usando gráfico de línea y gráfico de barras. ¿Cuál te parece que es el mejor gráfico para este caso?
    c. Graficar la cantidad de ventas en el año 2024 en función de los productos usando gráfico de barras y gráfico de torta. ¿Cuál te parece que es el mejor gráfico para este caso?

4. Usando el siguiente dataframe:

```
data = {
    'Jurisdicción': [
        'Ciudad Autónoma de Buenos Aires', 'Buenos Aires', 'Catamarca', 'Chaco', 'Chubut
        'Córdoba', 'Corrientes', 'Entre Ríos', 'Formosa', 'Jujuy', 'La Pampa', 'La Rioja
        'Mendoza', 'Misiones', 'Neuquén', 'Río Negro', 'Salta', 'San Juan', 'San Luis',
        'Santa Cruz', 'Santa Fe', 'Santiago del Estero', 'Tierra del Fuego, Antártida e
        'Tucumán'
    ],
    'Capital': [
        '', 'La Plata', 'San Fernando del Valle de Catamarca', 'Resistencia', 'Rawson',
        'Córdoba', 'Corrientes', 'Paraná', 'Formosa', 'San Salvador de Jujuy', 'Santa Ro
        'Mendoza', 'Posadas', 'Neuquén', 'Viedma', 'Salta', 'San Juan', 'San Luis',
        'Río Gallegos', 'Santa Fe', 'Santiago del Estero', 'Ushuaia', 'San Miguel de Tuc
    ],
    'Población (hab)': [
        3075646, 17541141, 415438, 1204541, 618994, 3760450, 1120801, 1385961,
        605193, 770881, 358428, 393531, 1990338, 1261294, 664057, 747610,
        1424397, 781217, 508328, 365698, 3536418, 978313, 173715, 1694656
    ],
    'Superficie (km2)': [
        205.9, 305907.4, 101486.1, 99763.3, 224302.3, 164707.8, 89123.3, 78383.7,
        75488.3, 53244.2, 143492.5, 91493.7, 149069.2, 29911.4, 94422, 202168.6,
        155340.5, 88296.2, 75347.1, 244457.5, 133249.1, 136934.3, 910324.4, 22592.1
    ],
    'PBI': [
        154863803.5, 292689868, 6150949.159, 9832642.672, 17747854.21, 69363739.19,
        7968012.982, 20743409.1, 3807057.419, 6484938.334, 6990262.458, 5590515.628,
        33431369.11, 9646825.835, 22564106.16, 10264584.42, 13438834.91, 8262308.568,
        11780849.36, 11663738.04, 81588690.27, 8387858.731, 7049276.383, 13856198.9
    ],
}

df = pd.DataFrame(data)
df.head()
```

a. Realizar un gráfico de barras horizontales que muestre la cantidad de habitantes por jurisdicción. Las provincias con mayor población, deben ubicarse en la parte superior.
b. Realizar un gráfico de torta que muestre el porcentaje de superficie de cada región del país.
c. Realizar un gráfico de barras verticales que muestre el PBI de cada jurisdicción. Las provincias con mayor PBI, deben ubicarse a la derecha.

d. Realizar un gráfico de puntos que muestre la relación entre la población y la superficie de cada jurisdicción. ¿Qué conclusión se puede tomar respecto al gráfico obtenido? ¿Hay alguna excepción? De ser así, analice cuál, incluyendo también un análisis de los gráficos anteriores.

> **i** datos.gob.ar
>
> ¿Sabías que en Argentina hay un portal de datos abiertos?
> Podés ingresar a datos.gob.ar y obtener información de diferentes áreas, como salud, educación, justicia, entre otras. ¡Es una excelente fuente de datos para hacer análisis!
> Para usarlos, podés guardar el archivo csv en google drive e importarlo de la siguiente forma como un dataframe:
>
> ```
> import pandas as pd
>
> # Acá va el link público de tu archivo de google drive + '/export?format=csv'
> url = "https://drive.google.com/drive/folders/ABC123XYZ456/export?format=csv"
> df = pd.read_csv(url)
> ```

5. Para calcular la energía potencial elástica (Ep) almacenada en un material elástico, se utiliza la siguiente fórmula:

$$Ep = \frac{1}{2} * k * x^2$$

donde k es la constante elástica (con unidad N/m) y x es la distancia que el resorte se estira o comprime (metros). La energía Ep resultante tiene como unidad el Joule (J).

Para un resorte con valor $k = 0.15 N/m$, graficar en línea continua la energía potencial elástica Ep en un rango de distancia x de 0 a 10m con pasos de 0.5. Incluir título, grilla, nombres de ejes y label (leyenda) con el valor de k.

6. Para calcular la posición final de un móvil dada una posición inicial, velocidad inicial, tiempo y aceleración, se tiene la siguiente fórmula:

$$x = x_0 + v_0.t + \frac{1}{2}.a.t^2$$

Donde x_0 es la posición inicial, v_0 la velocidad inicial, t el tiempo y a la aceleración.

Se tiene la siguiente información de dos móviles:

movil 1:

- Comienza a 0m

- Arranca desde el reposo con velocidad 0 m/s
- Su aceleración es 1 m/s^2

movil 2:

- Comienza a 500m
- Arranca con velocidad 4 m/s
- Su aceleración es 0.5 m/s^2

Para 5000 valores del tiempo del 0 al 100, mostrar un único gráfico con la posición respecto del tiempo de ambos móviles. Incluir grilla, títulos y label (leyenda) indicando el valor de la posicion inicial, velocidad y aceleración.

7. **Desafio** (obligatorio): Teniendo la planilla de notas del primer parcial del 1C2024 de los alumnos de la materia "Pensamiento Computacional", importar los datos como dataframe usando el siguiente código:

```
import pandas as pd

url_sheet = "https://docs.google.com/spreadsheets/d/17ei_NER5i_R-9QLnkeIjNprzTyoSqGowJ8y
df = pd.read_csv(url_sheet)
df.head()
```

Si la nota es "NaN", significa que el o la estudiante no rindió el examen.
En gráficos diferentes:

a. Realizar un gráfico de torta que muestre el porcentaje entre la cantidad de alumnos que asistieron al parcial y los que no.
b. Realizar un gráfico de barras que muestre la cantidad de alumnos que aprobaron y desaprobaron el parcial, sin contar a los ausentes.
c. Realizar un gráfico de puntos que muestre la relación entre el curso y la nota obtenida en el parcial. ¿Diría que existe una relación?

Evaluación de Código

> **Nota**
>
> En estos ejercicios se presenta código con errores, malas prácticas de programación o problemas de eficiencia. El objetivo es identificar los errores, explicar por qué fallan y proponer una corrección. Podés probar el código en tu entorno y, opcionalmente, consultar con herramientas de IA para verificar tu análisis.

1. La siguiente función intenta leer un archivo CSV y mostrar los estudiantes mayores de 20 años. El archivo tiene columnas: nombre, edad, carrera. Identificá el problema y corregilo.

```
import pandas as pd

def devolver_mayores_20(archivo):
    df = pd.read_csv(archivo)

    mayores = df[df[1] > '20']
    print(mayores)
```

2. La siguiente función intenta crear un array de NumPy y mostrar en pantalla su promedio, pero produce un resultado incorrecto. Identificá el error y corregilo.

```
import numpy as np

datos = [10, 20, 30, 40, 50]
array = np.array(datos)
for d in array:
    suma += d
    cant += d

return suma / cant
```

3. El siguiente código intenta agregar una columna calculada a un DataFrame, pero produce un error. Identificá por qué falla y corregilo.

```
import pandas as pd

data = {
    'producto': ['Manzana', 'Banana', 'Naranja'],
    'precio': [100, 80, 120],
    'cantidad': [5, 10, 3]
}
df = pd.DataFrame(data)

df = df[ df['precio'] * df['cantidad'] ]
print(df)
```

4. Leer el siguiente código. Identificar y explicar qué es lo que hace. El código contiene tres errores; identificarlos y solucionarlos.

```
import pandas as pd

def analizar_control_calidad(archivo):
    df = pd.read_csv(archivo)
```

```
    yogures = df[df['producto'] == 'yogur']
    yogures_validos = yogures[yogures['volumen_ml'].notnull()]
    total_volumen = yogures_validos[ yogures_validos['volumen_ml'].sum() ]

    completos = df['volumen_ml'].notnull() & df['temperatura_c'].notnull()
    cant_completos = len(completos)

    aptos = df[df['estado'] == 'aprobado']
    cant_aptos = len(aptos)

    total = len(df)
    cant_incompletos = total - cant_completos
    porcentaje_incompletos_del_total = (cant_incompletos / cant_completos) * 100

    lote_observado = porcentaje_incompletos_del_total >= 30

    resumen = {
        'concepto': [
            'Volumen total yogur',
            'Registros totales',
            'Registros completos',
            'Lotes aptos',
            'Porcentaje incompletos',
            'Lote observado'
        ],
        'valor': [
            total_volumen,
            total,
            cant_completos,
            cant_aptos,
            f"{porcentaje_incompletos_del_total:.0f}%",
            lote_observado
        ]
    }

    df_resumen = pd.DataFrame(resumen)
    df_resumen.to_csv('control_calidad.csv', index=False)
    print("Reporte generado exitosamente: control_calidad.csv")
```