**📝 Ejercicios propuestos**

1. **Conversor de monedas**  
   Escribir un programa que pida al usuario un monto en dólares y lo convierta a pesos argentinos y a euros. Usar funciones para cada conversión y mostrar el resultado con dos decimales.
2. **Calculadora geométrica**  
   Crear un programa que permita al usuario elegir una figura geométrica (cuadrado, rectángulo, triángulo, círculo) mediante un menú. Luego, calcular su área y perímetro usando funciones específicas para cada caso.
3. **Gestor de notas de alumnos**  
   El programa debe pedir el nombre de un alumno y sus 3 notas. Usando una función, calcular el promedio y mostrar si aprobó o desaprobó (mínimo 6). Guardar los resultados en un diccionario {nombre: promedio}.
4. **Juego del número secreto**  
   Generar un número aleatorio entre 1 y 100. El usuario debe adivinarlo. El programa debe dar pistas ("más alto", "más bajo") hasta que acierte. Llevar un contador de intentos y mostrarlo al final.
5. **Calculadora de factorial con recursividad**  
   Implementar una función recursiva que calcule el factorial de un número entero positivo ingresado por el usuario. Comparar el resultado con la versión iterativa.
6. **Gestión de inventario simple**  
   Simular un inventario con un diccionario de productos y precios. Permitir:
   * Agregar productos
   * Eliminar productos
   * Listar productos
   * Calcular el total del inventario.  
     Usar un menú con match-case o if-elif.
7. **Validador de contraseñas**  
   Escribir una función que valide contraseñas según estas reglas:
   * Mínimo 8 caracteres
   * Al menos una mayúscula
   * Al menos un número
   * Al menos un carácter especial (!@#$%&\*).  
     Probar la función con varias entradas.
8. **Conversor de temperaturas modularizado**  
   Crear un módulo conversores.py que tenga funciones para convertir:
   * Celsius ↔ Fahrenheit
   * Celsius ↔ Kelvin  
     Importar el módulo en otro archivo principal y probarlo.
9. **Simulación de cajero automático**  
   Hacer un programa que pida un PIN para acceder. Luego permitir:
   * Consultar saldo
   * Depositar dinero
   * Retirar dinero (si hay saldo suficiente).  
     Usar funciones para cada acción y estructuras de control.
10. **Control de versiones simulado**  
    Crear un programa que simule algunos comandos básicos de Git:

* init → iniciar repositorio
* add <archivo> → agregar archivo
* commit <mensaje> → guardar cambios
* status → mostrar estado actual.  
  Usar un diccionario para guardar el estado de los archivos y funciones para cada comando.