Juan Pedro Colomo González 1653625

EJERCICIO 1:

```
catalogo = []


def agregar_productos(*productos):
    for p in productos:
        if isinstance(p, dict):
            catalogo.append(p)


def mostrar_catalogo():
    print("\nCatálogo de productos:")
    for i, producto in enumerate(catalogo, 1):
        print(f"{i}. {producto['nombre']} - Q{producto.get('precio', 0)} - {producto.get('categoria', 'Sin categoría')}")


def buscar_productos(**filtros):
    print("\nProductos filtrados:")
    for producto in catalogo:
        cumple = True
        if "categoria" in filtros and producto.get("categoria") != filtros["categoria"]:
            cumple = False
        if "precio_max" in filtros and producto.get("precio", 0) > filtros["precio_max"]:
            cumple = False
        if cumple:
            print(f"- {producto['nombre']} | Q{producto.get('precio', 0)} | {producto.get('categoria', 'Sin categoría')}")
```

Juan Pedro Colomo González 1653625

EJERCICIO 2:

```
def validar_argumentos(*args):
    return all(isinstance(arg, (int, float)) for arg in args)


def operacion_matematica(tipo, *numeros):
    if not validar_argumentos(*numeros):
        return "Error: todos los valores deben ser numéricos."

    if tipo == "suma":
        resultado = sum(numeros)
    elif tipo == "resta":
        resultado = numeros[0]
        for n in numeros[1:]:
            resultado -= n
    elif tipo == "multiplicacion":
        resultado = 1
        for n in numeros:
            resultado *= n
    elif tipo == "division":
        try:
            resultado = numeros[0]
            for n in numeros[1:]:
                resultado /= n
        except ZeroDivisionError:
            return "Error: división entre cero."
    else:
```

```
        return "Operación no válida."

    return resultado

def registro_operacion(**detalles):
    print("\nRegistro de operación:")
    for clave, valor in detalles.items():
        print(f"{clave}: {valor}")
```

EJERCICIO 3:

```
historial_respuestas = []

def crear_encuesta(titulo, *preguntas):
    return {
        "titulo": titulo,
        "preguntas": list(preguntas)
    }

def mostrar_encuesta(encuesta):
    print(f"\nEncuesta: {encuesta['titulo']}")
    for i, pregunta in enumerate(encuesta["preguntas"], 1):
        print(f"{i}. {pregunta}")

def responder_encuesta(encuesta, **respuestas):
    if len(respuestas) != len(encuesta["preguntas"]):
        print("Error: deben responderse todas las preguntas.")
```

```
        return
    historial_respuestas.append(respuestas)
    print("Respuestas guardadas correctamente.")
```

EJERCICIO 4:

```
cv = {}

def crear_curriculum(nombre, *experiencias):
    cv["nombre"] = nombre
    cv["experiencias"] = experiencias

def agregar_datos_personales(**datos):
    cv.update(datos)

def mostrar_curriculum(completo=True, **opciones):
    print(f"\nCurrículum de: {cv.get('nombre')}")

    if completo or opciones.get("mostrar_telefono", True):
        print("Teléfono:", cv.get("telefono", "No proporcionado"))
    if completo or opciones.get("mostrar_correo", True):
        print("Correo:", cv.get("correo", "No proporcionado"))
    if completo or opciones.get("mostrar_linkedin", False):
        print("LinkedIn:", cv.get("linkedin", "No proporcionado"))

    print("\nExperiencia laboral:")
    for empresa, puesto, años in cv.get("experiencias", []):
```

```
    print(f"- {puesto} en {empresa} ({años} años)")
```