```
import os

import shutil

import random


# ----- Configuración del entorno -----

# La ruta base de las imágenes originales se define en esta variable. Cada categoría (por ejemplo, 'boton', 'corte', 'abierta') debe estar en su subcarpeta dentro de esta ruta.

origen = "imagenes_originales"


# La ruta destino del conjunto de datos se definirá aquí, donde se crearán las carpetas de train, val y test.

destino = "dataset"


# Proporciones de división: estas variables definen el porcentaje de imágenes que se asignarán a los conjuntos de entrenamiento, validación y prueba.

train_ratio = 0.7  # 70% de las imágenes serán para entrenamiento

val_ratio = 0.15  # 15% para validación

test_ratio = 0.15  # 15% para prueba


# Definimos las categorías de las flores. Cada categoría debe tener su carpeta en la ruta origen, y será usada para distribuir las imágenes.

categorias = ["boton", "corte", "abierta"]


# ----- Creación de las carpetas de salida -----

# Antes de distribuir, se crean las carpetas de train, val y test, y dentro de ellas las subcarpetas para cada categoría. Esto asegura que la estructura del dataset sea coherente y lista para el modelo.

for carpeta in ["train", "val", "test"]:

  for cat in categorias:

    path = os.path.join(destino, carpeta, cat)

    # Creamos las carpetas, sin error si ya existen
```

```
        os.makedirs(path, exist_ok=True)


# ----- Función para distribuir las imágenes por categoría -----


def distribuir(categoria):
    # Construimos la ruta de la categoría dentro del origen
    ruta_cat = os.path.join(origen, categoria)


    # Verificamos que la ruta de la categoría exista. Si no, se reporta y se salta a la siguiente categoría.
    if not os.path.exists(ruta_cat):
        print(f" ⚠️ Carpeta de origen no encontrada: {ruta_cat}")
        return


    # Listamos todas las imágenes en la carpeta, filtrando solo archivos con extensiones .jpg o .png
    imagenes = [f for f in os.listdir(
        ruta_cat) if f.lower().endswith((".jpg", ".png"))]


    # Si no hay imágenes en esa categoría, se reporta y se pasa a la siguiente.
    if len(imagenes) == 0:
        print(f" ⚠️ No hay imágenes en la carpeta: {ruta_cat}")
        return


    # Mezclamos las imágenes de forma aleatoria para evitar sesgos en la distribución
    random.shuffle(imagenes)
    total = len(imagenes)  # Número total de imágenes en la categoría


    # Calculamos cuántas imágenes van a cada conjunto basándonos en las proporciones definidas
```

```
    n_train = int(total * train_ratio)  # Cantidad para entrenamiento
    n_val = int(total * val_ratio)   # Cantidad para validación
    n_test = total - n_train - n_val  # Cantidad restante para prueba

    # Se imprime un resumen del total de imágenes y su división en cada conjunto
    print(f"{categoria}: total={total}, train={n_train}, val={n_val}, test={n_test}")

    # Recorremos las imágenes y las asignamos a su conjunto correspondiente
    for i, img in enumerate(imagenes):
        if i < n_train:
            destino_final = os.path.join(destino, "train", categoria, img)
        elif i < n_train + n_val:
            destino_final = os.path.join(destino, "val", categoria, img)
        else:
            destino_final = os.path.join(destino, "test", categoria, img)

        # Copiamos la imagen desde la ruta original a su carpeta final en el conjunto respectivo
        shutil.copy2(os.path.join(ruta_cat, img), destino_final)

# Se recorre la lista de categorías definidas previamente (boton, corte, abierta).
# Para cada categoría, se invoca la función 'distribuir()', la cual:
# 1. Accede a la carpeta correspondiente en el directorio de origen.
# 2. Mezcla aleatoriamente las imágenes.
# 3. Calcula la proporción de división (train, val, test).
# 4. Copia cada imagen a su respectiva carpeta destino.
for cat in categorias:
    distribuir(cat)

# Una vez finalizado el procesamiento de todas las categorías,
```

```
# se imprime un mensaje en consola indicando que el proceso
# de distribución del dataset se completó exitosamente.
# Este mensaje sirve como verificación de que el script terminó sin errores.
print("\n ✅ Distribución completada correctamente.")
```