'''

PIL能够实现许多图片数据的底层表示和处理，

比如在相素层面进行相关分析等，

并支撑常用的图像检索任务。

请围绕其相关功能，

结合异常捕获和自定义异常，

完成如下题目。

'''

import PIL

from PIL import Image

import numpy as np

from scipy.stats import pearsonr, spearmanr, kendalltau

import dashscope

import base64

import json

from http import HTTPStatus

from sklearn.metrics.pairwise import cosine\_similarity

import os

import sys

from ali\_key import KEY

class ImageQueryError(Exception):

    def \_\_init\_\_(self):

        pass

class ImageQueryShapeNotMatchError(ImageQueryError):

    def \_\_init\_\_(self,image\_path1,image\_path2):

        # 打开第一张图片

        img1 = Image.open(image\_path1)

        # 获取第一张图片的尺寸

        size1 = img1.size

        # 打开第二张图片

        img2 = Image.open(image\_path2)

        # 获取第二张图片的尺寸

        size2 = img2.size

        self.message= f"大小不对，img1的大小是{size1}, img2的大小是{size2}"

class ImageQuery:

    '''

    1. 异常捕获。

    实现ImageQuery类的\_create\_and\_image方法，

    其利用PIL.Image类的open方法打开并返回一个Image实例，

    但考虑到open方法可能产生FileNotFoundError或PIL.UnidentifiedImageError，

    请在该方法中对这两个异常进行捕获和处理（打印或记入日志，相关信息包括打开的文件路径和详细的异常描述）。

    '''

    def \_\_init\_\_(self, image\_path=None):

        self.image = self.\_create\_and\_image(image\_path) if image\_path else None

        self.histogram = []

    def \_create\_and\_image(self, fp):

        try:

            return Image.open(fp)

        except FileNotFoundError as e:

            print(f"文件未找到错误: {e}")

            return None

        except PIL.UnidentifiedImageError as e:

            print(f"无法识别的图片格式: {e}")

            return None

很好笑，第一问一开始没发现问题，因为第二问用的是路径导入，导致了第三问死活做不出来。问题出在一开始使用递归调用\_create\_and\_image上，self.image初始值是none，且递归没有做返回，导致虽然在\_create\_and\_image里成功调用了，但是实际上并没有成功传进去。

    '''

    2. 图片的相似性计算。

    在ImageQuery类中实现一种简单图片相似性的计算方法pixel\_difference，

    即直接对两个图片逐相素相减，并累积求和差异的绝对值，

    继而除以相素总数。

    注意该方法可能会抛出一个叫ImageQueryShapeNotMatchError的自定义异常，

    其继承了ImageQueryError（本次作业自定义的顶层异常类），

    即当比较相似性的两张图片形状（长宽）不一致性时。请在该方法中抛出该异常，包含两个图片的形状信息。

    '''

    def pixel\_difference(image\_path1, image\_path2):

        # 打开两张图片

        img1 = Image.open(image\_path1)

        img2 = Image.open(image\_path2)

        # 确保两张图片的尺寸相同

        if img1.size != img2.size:

            raise ImageQueryShapeNotMatchError(image\_path1, image\_path2)

        else:

            # 将图片转换为NumPy数组

            img1\_array = np.array(img1, dtype=np.float32)

            img2\_array = np.array(img2, dtype=np.float32)

            # 计算像素差异绝对值

            difference = np.abs(img1\_array - img2\_array)

            # 计算差异的总和

            total\_difference = np.sum(difference)

            # 计算像素总数

            total\_pixels = img1\_array.size  # 直接使用展平后的元素数量

            # 计算平均差异

            average\_difference = total\_difference / total\_pixels

            return average\_difference

第二问直接使用了路径传参进函数，对应会出现两种结果：

1. 大小不匹配

0d5ab483d80807c6ecfd0ef4b007372

二．大小匹配，可以进行比较



    '''

    3. 图片的直方图相似性计算。

    在ImageQuery类中实现更多的相似性计算方法。

    具体地，利用PIL.Image类的histogram方法，

    获取图片相素的直方图，

    进而用scipy.states中的相关性计算方法来得到不同的相似性，

    如pearson，spearman，kendall等。

    这些方法并不要求图片形状一致。注意，这些相似性方法还能够返回显著性。

    '''

    def get\_histogram(self):

        histogram = self.image.histogram()

        self.histogram = histogram

        return histogram

    def compute\_similarity(self, other\_image\_query, method='pearson'):

        """

        计算与另一个ImageQuery对象的相似性。

        :param other\_image\_query: 另一个ImageQuery对象

        :param method: 相似性计算方法，可选 'pearson', 'spearman', 'kendall'

        :return: 相似性系数和显著性水平（p值）

        """

        if method == 'pearson':

            correlation, p\_value = pearsonr(ImageQuery.get\_histogram(self), ImageQuery.get\_histogram(other\_image\_query))

        elif method == 'spearman':

            correlation, p\_value = spearmanr(ImageQuery.get\_histogram(self), ImageQuery.get\_histogram(other\_image\_query))

        elif method == 'kendall':

            correlation, p\_value = kendalltau(ImageQuery.get\_histogram(self), ImageQuery.get\_histogram(other\_image\_query))

        else:

            raise ValueError("Unsupported method. Choose from 'pearson', 'spearman', 'kendall'.")

        return correlation, p\_value

第三问，这问真的吃了很多很多亏。首先是第一问埋下的伏笔，导致这里一直报错，首先是变量无法被当做实例变量传入，然后是histogram无法调用，再然后是x y必须是多于2项的列表（如果真的跑出来一定是多于2项的， 这里少于2项原因经过debug后发现是空值）

最后改掉了第一问的问题后，第三问简单修修就过了。

以pearson为例展示



    '''

    4. 图片的大模型嵌入。

    在ImageQuery类中实现基于大模型的相似性计算方法，

    即利用相关API(具体见Demo ali\_image\_embed.py或者ark\_image\_embed.py)

    首先将图片嵌入为向量，继而通过向量的余弦相似度等给出相似性大小(cos\_simi.py)。

    注意，选一个大模型实现即可，ali和字节均提供一定的免费token额度。

    '''

    def read\_image\_as\_base64(self):

        if self.image is None:

            return None

        from io import BytesIO

        buffered = BytesIO()

        self.image.save(buffered, format=self.image.format)

        return base64.b64encode(buffered.getvalue()).decode('utf-8')

第四问，这一次用了阿里的大模型。分别对两个实例（图片）求取向量后，参考cos\_simi.demo，对两个向量进行余弦角计算，获取相似值。（有一部分代码我写在底下的函数调用里了）

image\_path1 = '第7周/w7/test.png'

image\_path2 = '第7周/w7/test2.jpg'

image\_path3 = '第7周/w7/test3.jpeg'

image\_path4 = '第7周/wx1.png'

image\_path5 = '第7周/wx2.png'

try:

    image\_query1 = ImageQuery(image\_path1)

    image\_query2 = ImageQuery(image\_path2)

    # 计算像素差异

    diff = ImageQuery.pixel\_difference(image\_path4, image\_path5)

    print(f"Pixel difference: {diff}")

    # 计算直方图相似性

    pearson\_similarity, pearson\_p\_value = image\_query1.compute\_similarity(image\_query2)

    print(f"Pearson similarity: {pearson\_similarity}, p-value: {pearson\_p\_value}")

    # 读取图片并转换为Base64

    base64\_image\_1 = image\_query1.read\_image\_as\_base64()

    image\_data1 = f"data:image/{image\_path1};base64,{base64\_image\_1}"

    base64\_image\_2 = image\_query2.read\_image\_as\_base64()

    image\_data2 = f"data:image/{image\_path2};base64,{base64\_image\_2}"

    # 输入数据

    inputs1 = [{'image': image\_data1}]

    inputs2 = [{'image': image\_data2}]

    # 调用模型接口

    resp1 = dashscope.MultiModalEmbedding.call(

        api\_key = KEY,

        model="multimodal-embedding-v1",

        input = inputs1

    )

    if resp1.status\_code != HTTPStatus.OK:

        raise Exception(f"API call 1 failed with status code {resp1.status\_code}")

    a = resp1.output['embeddings'][0]['embedding']

    resp2 = dashscope.MultiModalEmbedding.call(

        api\_key = KEY,

        model="multimodal-embedding-v1",

        input = inputs2

    )

    if resp2.status\_code != HTTPStatus.OK:

        raise Exception(f"API call 2 failed with status code {resp2.status\_code}")

    b = resp2.output['embeddings'][0]['embedding']

    # 计算余弦相似度

    similarity = cosine\_similarity([a], [b])  # 注意这里需要将向量包装为2D数组

    # 输出是一个矩阵,如果只比较两个向量，结果是 1x1 矩阵，所以取 [0][0]。

    print("余弦相似度(test.jpg, test2.jpg):", similarity[0][0])

except ImageQueryShapeNotMatchError as iqsnme:

    print(iqsnme.message)

except Exception as e:

    print(f"An error occurred: {e}")