```python
"""
Author: Harsh Bhate
GTID: 903424029
Homework #: 2
Question #: 8
Date: 6 Sept 2018
"""

import argparse
import dippykit as dip
import numpy as np
import os
import sys
import pdb

class bcolors:
    HEADER = '\033[95m'
    OKBLUE = '\033[94m'
    OKGREEN = '\033[92m'
    WARNING = '\033[93m'
    FAIL = '\033[91m'
    ENDC = '\033[0m'
    BOLD = '\033[1m'
    UNDERLINE = '\033[4m'

IMAGE_PATH = "/home/harshbhate/Desktop/cameraman.png"
SAVE_PATH = "/home/harshbhate/Desktop/q8"
DOWNSAMPLE_DIR = "downsample"
UPSAMPLE_DIR = "upsample"
DFT_DIR = "dft"
PSNR_LOG = "psnr_log_1.txt"

def basic_image_ip(im_path, args, convert_to_float = True, normalize = True):
    '''Function to read image, convert to float and normalize'''
    if (os.path.exists(im_path)):
        X = dip.image_io.im_read(im_path)
    else:
        print (bcolors.FAIL+"File Path not found, aborting!"+bcolors.ENDC)
        sys.exit()
    if args.verbose:
        print (bcolors.OKGREEN+"Converted Image to Float"+bcolors.ENDC)
    if (convert_to_float):
        X = dip.im_to_float(X)
    if (convert_to_float and normalize):
        if args.verbose:
            print ("Normalizing")
        X *= 255
    return X

def float_image_op(IMG, save_path, args, downsample = True):
    '''Function to display and save an float image'''
    IMG = dip.float_to_im(IMG/255)
    if downsample:
        save_path = os.path.join (SAVE_PATH, DOWNSAMPLE_DIR,save_path)
    else:
        save_path = os.path.join (SAVE_PATH, UPSAMPLE_DIR,save_path)
    dip.image_io.im_write(IMG, save_path)
    if args.verbose:
        if downsample:
            print ("Downsampled Image is saved")
        else:
            print ("Upsampled Image is saved")
    dip.imshow(IMG)
    dip.show()

def L_matrix(M_matrix, args):
    '''Returns the inverse of M matrix'''
    if args.verbose:
        print ("Finding the inverse of M")
    return np.linalg.inv(M_matrix)

def upsample(Xd, L, args, interpolation = None, im_path = "m_1.jpg"):
    '''Return the upsample of the image'''
    if args.verbose:
        if interpolation is not None:
            print ("Doing the linear upsampling")
        else:
            print ("Doing upsampling")
    X = dip.sampling.resample(Xd, L, interp = interpolation)
    #float_image_op(X, im_path, args, downsample=False)
    return X

def downsample (M, args, im_path = "m_1.jpg"):
    '''Shows the downsampled Image'''
    X = basic_image_ip(IMAGE_PATH, args)
    if args.verbose:
        print ("Doing the downsampling")
    X = dip.sampling.resample(X, M)
    #float_image_op(X, im_path, args)
    return X

def dft(args):
    '''Compute and show DFT of image'''
    X = basic_image_ip(IMAGE_PATH, args)
    if args.verbose:
        print (X.shape)
    fX = dip.fft2(X)
    fX = dip.fftshift(fX)
    fX = np.log(np.abs(fX))
    if args.verbose:
        print ("Done with FFT")
    dip.imshow(fX)
    dip.show()

def PSNR(Xt ,args):
    '''Returns PSNR and saving in a text file'''
    if args.verbose:
        print ("Computing the PSNR and maintaining log")
    X = basic_image_ip(IMAGE_PATH, args)
    l0,b0 = X.shape
    l1,b1 = Xt.shape
    if l0 >= l1:
        l = l1
    else:
        l = l0
    if b0 >= b1:
        b = b1
    else:
        b = b0
    X = X[0:l,0:b]
    Xt = Xt[0:l,0:b]
    P = dip.metrics.PSNR(X, Xt, 255)
    log_path = os.path.join(SAVE_PATH,PSNR_LOG)
    f = open(log_path, 'a')
```

```python
    save_str = str(P)+"\n"
    f.write(save_str)
    f.close()
    if args.verbose:
        print ("The PSNR is: "+str(P))

def parser():
    '''Parsing the input Argument'''
    parser = argparse.ArgumentParser()
    parser.add_argument("-v","--verbose", help="increase output verbosity",
                        action="store_true")
    args = parser.parse_args()
    return args

if __name__ == "__main__":
    '''Main Function'''
    pdb.set_trace()
    args = parser()
    dft(args)
    M = np.array([[3, 1],
                  [1, 2]])
    if args.verbose:
        print ("The M matrix:\n"+str(M))
    L = L_matrix(M, args)
    if args.verbose:
        print ("The L matrix:\n"+str(L))
    Xd = downsample(M, args, 'm_5.jpg')
    Xt = upsample(Xd, L, args, None, 'm_5.jpg')
    Xt = upsample(Xd, L, args, 'lin', 'm_5_lin.jpg')
    p = PSNR (Xt, args)q
```