```
#!/usr/bin/env python3

import cv2
from matplotlib import pyplot as plt
import argparse
import csv

# fancy progress bar
import tqdm

# first and obligatory arguent is the picture
parser = argparse.ArgumentParser()
parser.add_argument("jpg", help="Input picture for Histogram")
parser.add_argument("-l", "--createCSV", help="create csv logfile", type=str, nargs='?
')
parser.add_argument("-o", "--outFile", help="save to file instead of Display Result",
type=str, nargs='?')
parser.add_argument("-s", "--single", help="create 3 single histograms instead of the
combined", action="store_false")
parser.add_argument("-g", "--greyscale", help="create histogramm over greyscale conver
ted image", action="store_true")

args = parser.parse_args()


def calcHist(bgr_pic):
    r = []
    g = []
    b = []

    # split up the image
    for line in bgr_pic:               # each row (or line....i actually dont care)
        for pixel in line:             # each pixel in line (or row)
            b.append(pixel[0])
            g.append(pixel[1])
            r.append(pixel[2])

    hist_b = []
    hist_g = []
    hist_r = []

    # Smaller images speed things up ... but also change the result
    for i in tqdm.tqdm(range(255), ascii=True):
        hist_b.append(b.count(i))
        hist_g.append(g.count(i))
        hist_r.append(r.count(i))

    return hist_b, hist_g, hist_r


def calcHist_Grey(bgr_pic):
    # convert to grey
    grey = cv2.cvtColor(bgr_pic, cv2.COLOR_BGR2GRAY)

    # split up image
    gr = []
    for line in grey:               # each row (or line....i actually dont care)
        for pixel in line:             # each pixel in line (or row)
            gr.append(pixel)

    # this should be way faster (factor 3)
    hist_grey = []
    for i in tqdm.tqdm(range(255), ascii=True):
        hist_grey.append(gr.count(i))

    return hist_grey


def __histShort(blue, green=None, red=None):
    scale = range(0, 255, 1)
    if not green and not red:
        # actually blue is grey here but...t fuck it
        plt.plot(scale, blue, 'k.')
    else:
        plt.plot(scale, blue, 'b.', scale, green, 'g.', scale, red, 'r.')

    if args.outFile:
        plt.savefig(args.outFile, bbox_inches='tight')
        print("outFile created:\t %s" % args.outFile)
    else:
        plt.show()


def __createCSV(name, b, g=None, r=None):
    # This will write a file with the calculated history data
    # will be necessary for the "OVERALL" - histogramm

    with open(name, mode='w') as csv_f:
        csvWrite = csv.writer(csv_f, delimiter=',', quotechar='"', quoting=csv.QUOTE_M
INIMAL)

        csvWrite.writerow(b)
        if g and r:
            csvWrite.writerow(g)
            csvWrite.writerow(r)


def __histLong(b, g, r):
    plt.title("Histograms of color channels")
    # yaxis= max(len(b), len(g), len(r))
    yaxis = max(max(b), max(g), max(r))

    blue = plt.subplot(311)
    blue.set_xlabel("Intensity of Blue Pixels")
    blue.set_ylabel("Count of Pixles")
    # blue.set_title("Blue Channel")
    blue.axis([0, 255, 0, yaxis])
#    blue.hist(b, color='b', histtype='stepfilled')
    blue.plot(b, color='b')
    blue.fill_between(0, b)

    green = plt.subplot(312)
    green.set_xlabel("Intensity of Green Pixels")
    green.set_ylabel("Count of Pixles")
    # green.set_title("Green Channel")
    green.axis([0, 255, 0, yaxis])
#    green.hist(g, color='g', histtype='stepfilled')
    green.plot(g, color='g')
    green.fill_between(0, g)

    red = plt.subplot(313)
    red.set_xlabel("Intensity of Red Pixels")
    red.set_ylabel("Count of Pixles")
    # red.set_title("Red Channel")
    red.axis([0, 255, 0, yaxis])
#    red.hist(r, color='r',histtype='stepfilled')
    red.plot(r, color='r')
    red.fill_between(0, r)

    if args.outFile:
        plt.savefig(args.outFile, bbox_inches='tight')
        print("outFile created:\t %s" % args.outFile)
    else:
        plt.show()


def main():
    img = cv2.imread(args.jpg)
    if args.greyscale:
        # calc and use greyscale / long doesnt make sense here
        grey = calcHist_Grey(img)
        __histShort(grey)
    else:
        blue, green, red = calcHist(img)
        if args.single:
            __histShort(blue, green, red)
        else:
            __histLong(blue, green, red)

    if args.createCSV:
        if not args.greyscale:
            __createCSV(args.createCSV, blue, green, red)
        else:
            __createCSV(args.createCSV, grey)

        print("logFile Created:\t %s" % args.createCSV)

    cv2.destroyAllWindows()


def test():
    img = cv2.imread(args.jpg)
    gr = calcHist_Grey(img)
    __histShort(gr)

    cv2.destroyAllWindows()


if __name__ == '__main__':
    main()
    # test()


# TODO: is threre a nice way to FLATEN images?
```