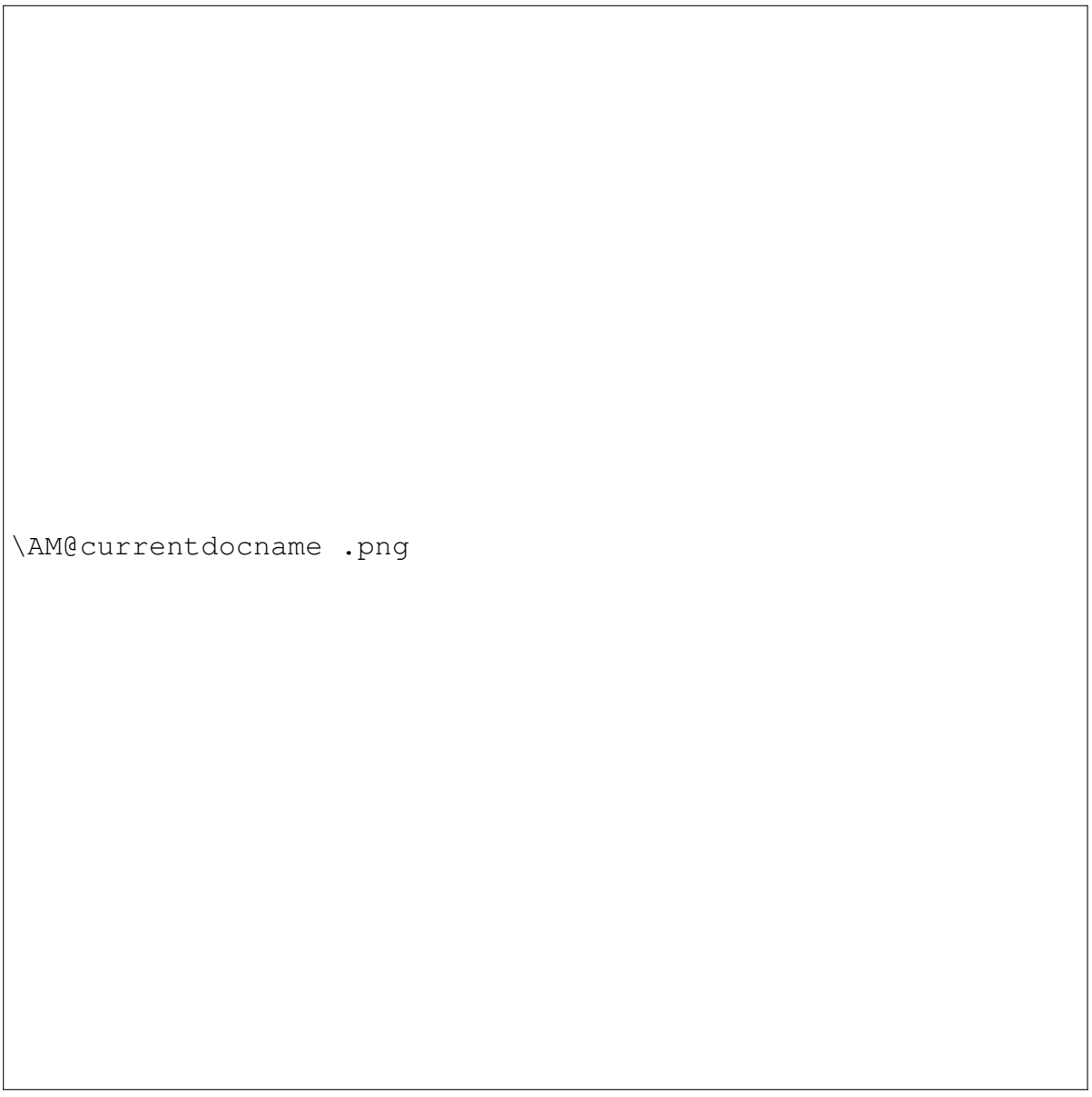
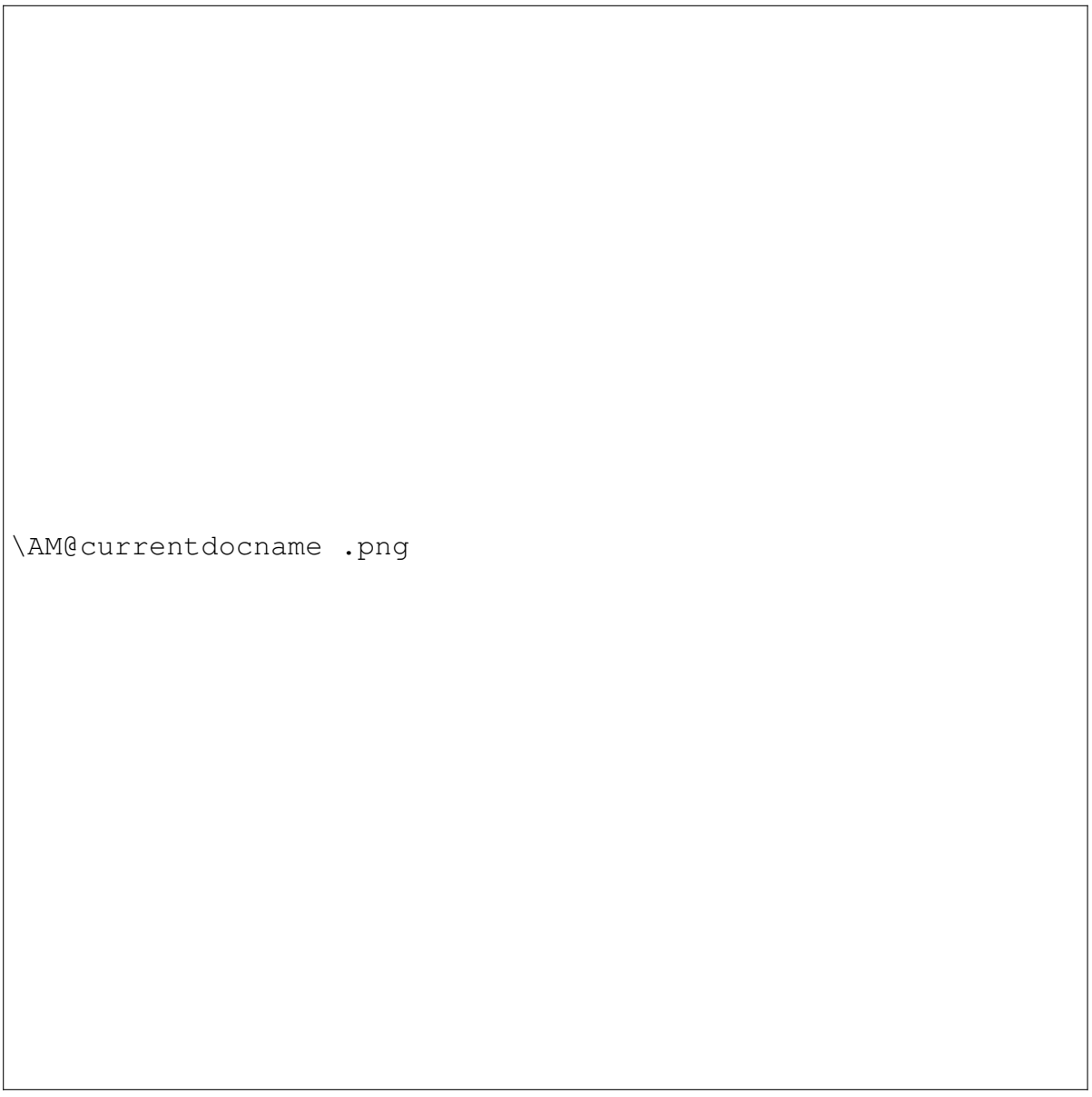


.png



.png

Aufgabe 2

```
72
73    # compute histograms
74    img_histo = compute_cumHisto(img, 1)
```

```
8  def compute_cumHisto(img, binSize=1):
9
10    return np.cumsum(bin_Histo(img, binSize))
11
12
13 def bin_Histo(img, bin=1):
14    intervalls = np.ceil(256 / bin)
15    histo = np.zeros(shape=intervalls)
16
17    for x in range(0, img.width):
18       for y in range(0, img.height):
```

.png

.png

Aufgabe 4

```
20          index = (brightness * intervalls) / 256
21          histo[index] += 1
22
23      return histo
24
25
26  def match_Histo(img_histo, ref_histo):
27
28      #img_histo . . . original histogram
29      #ref_histo . . . reference histogram
30      #returns the mapping function LUT to be applied to the image
```

Aufgabe 5.a) Wenn in einem Foto nur dunkle Bereich aufgehellt und helle Bereiche abgedunkelt werden können Details erhalten bleiben.

```
33
34      for i in range(0,255):
35          P_i = img_histo[i] / img_histo[255]
36          for j in range(0, 255):
37              P_j = ref_histo[j] / ref_histo[255]
38              if P_i == P_j :
39                  LUT[i] = j
40                  break;
41
42      return LUT
43
44
45  def apply_LUT(img, lut):
46
47      for x in range(0, img.width):
48          for y in range(0, img.height):
49              edit = img.getpixel((x, y))
50              lut_edit = lut[edit].item()
51              img.putpixel((x, y), lut_edit)
52
53      return img
54
55
56
57  def rgb2gray(rgb):
58
59      # convert to grayscale image (only one channel)
60      return rgb.convert('L')
61
62
63  if __name__ == "__main__":
64
65      # read img
66      img = Image.open("bild01.jpg")
67      ref = Image.open("bild02.jpg")
68
69      # convert to grayscale
```