In [ ]:
```
import binascii
import numpy as np
import os
import torch
import cv2
```

In [ ]:
```
filename = 'test.png'
with open(filename, 'rb') as f:
    content = f.read()
content=binascii.hexlify(content)
print(content) #not an arry like
```

In [ ]:
```
crux = str(content)[2:len(content)]
print(crux)
data = bytes.fromhex(crux)
with open('image.png','wb') as f:
    f.write(data)
```

In [ ]:
```
def strip(content):
    curx = str(content)[2:len(content)]
    return curx
def direct_compare(img1, img2):
    with open(img1, 'rb') as f:
        content1 = f.read()
        f.close()
    with open(img2, 'rb') as f2:
        content2 = f2.read()
        f2.close()
    difference = np.empty
    content1 = strip(content1)
    content2 = strip(content2)
    for i in range(len(content1)):
        os.system('clear')
        give_out = str(i*100/len(content1))+" %"+"done"
        print(give_out)
        if (content1[i]!=content2[i]) :
            type_out = "at :"+str(i)+": is :"+str(content2[i])
            difference = np.append(difference,type_out)
    return difference
```

In [ ]:
```
def consider_same(img1,img2):
    with open(img1, 'rb') as f1:
        content1 = f1.read()
        f1.close()
    with open(img2, 'rb') as f2:
        content2 = f2.read()
        f2.close()
    if content1 == content2:
        out = "same"
    else:
        out = "not same"
    return out
```

In [ ]:
```
img = cv2.imread('./test.png',0) #reading the image in grey scale
img_f = np.float32(img) #float convertion
print("img_f: \n"+str(img_f))
DCT = cv2.dct(img_f) #applying the DCT
```

```
print("DCT: \n"+str(DCT))
IDCT = cv2.idct(DCT)
print("IDCT: \n"+ str(IDCT))
```

```
img_f:
[[255. 255. 255. ... 255. 255. 255.]
 [255. 255. 255. ... 255. 255. 255.]
 [255. 255. 255. ... 255. 255. 255.]
 ...
 [255. 255. 255. ... 255. 255. 255.]
 [255. 255. 255. ... 255. 255. 255.]
 [255. 255. 255. ... 255. 255. 255.]]
DCT:
[[ 1.05646195e+05  2.54367432e+03  2.53769668e+04 ...  7.30974054e+00
  -2.69341869e+01 -2.12951660e+00]
 [-3.42387988e+03  4.23082520e+03  1.89980969e+03 ...  1.35087311e+00
  -1.38156309e+01  3.22126675e+00]
 [ 9.62704395e+03 -1.71177588e+03 -9.75689355e+03 ...  1.28259525e+01
  -1.06176615e+00  1.45622969e+01]
 ...
 [ 1.52236834e+01  1.87208697e-01 -8.25145626e+00 ...  1.61212826e+00
  -3.77964687e+00  6.98530102e+00]
 [ 1.05191526e+01 -6.95934725e+00 -6.63289022e+00 ... -7.01942635e+00
  -5.45922709e+00 -4.77775764e+00]
 [ 1.47783947e+01 -1.19657815e+00 -6.49815178e+00 ... -1.32078695e+01
   1.15399370e+01  7.71744108e+00]]
IDCT:
[[254.99988 254.99985 254.9998  ... 254.99994 254.99988 254.99988]
 [255.00003 255.00006 254.99997 ... 255.00014 255.       255.00002]
 [254.99992 254.99994 254.99988 ... 254.99994 254.99994 254.99988]
 ...
 [255.00002 254.99992 254.99992 ... 255.00006 254.99994 254.99997]
 [255.00009 255.00006 254.99997 ... 255.00009 255.00009 255.00002]
 [254.99995 254.99995 254.9999  ... 255.00003 254.99998 254.99998]]
```

In [ ]:
```
def batch_compare(img1, img2, batch_size = 256000):
    with open(img1, 'rb') as f:
        content1 = f.read()
        f.close()
    with open(img2, 'rb') as f2:
        content2 = f2.read()
        f2.close()
    difference = ""
    difference_log = np.empty
    content1 = binascii.hexlify(content1)
    content1 = strip(content1)
    content2 = binascii.hexlify(content2)
    content2 = strip(content2)
    i = 0
    while 1==1:
        left = content1[i:(i+batch_size)]
        right = content2[i:(i+batch_size)]
        if (left!=right):
            type_out_1 = "at :"+str(i)+": is :"+str(content2[i:i+batch_si
            difference_log = np.append(difference_log,type_out_1)
            type_out_2 = "4d61726b"+str(i)+"4d61726b6f7574"+str(content2[
            difference = difference + type_out_2
        i = i+batch_size
        if (i>=len(content1)):
            break
    return difference_log, difference
```

In [ ]:
```
img1 = './1.jpg'
img2 = './2.jpg'
output_log, output = batch_compare(img1,img2,1000)
print(output_log)
```

In [ ]:
```
print(output)
print(len(output))
```

In [ ]:
```
output = bytes.fromhex(output)
with open('22.jpg','wb') as f:
    f.write(output)
```