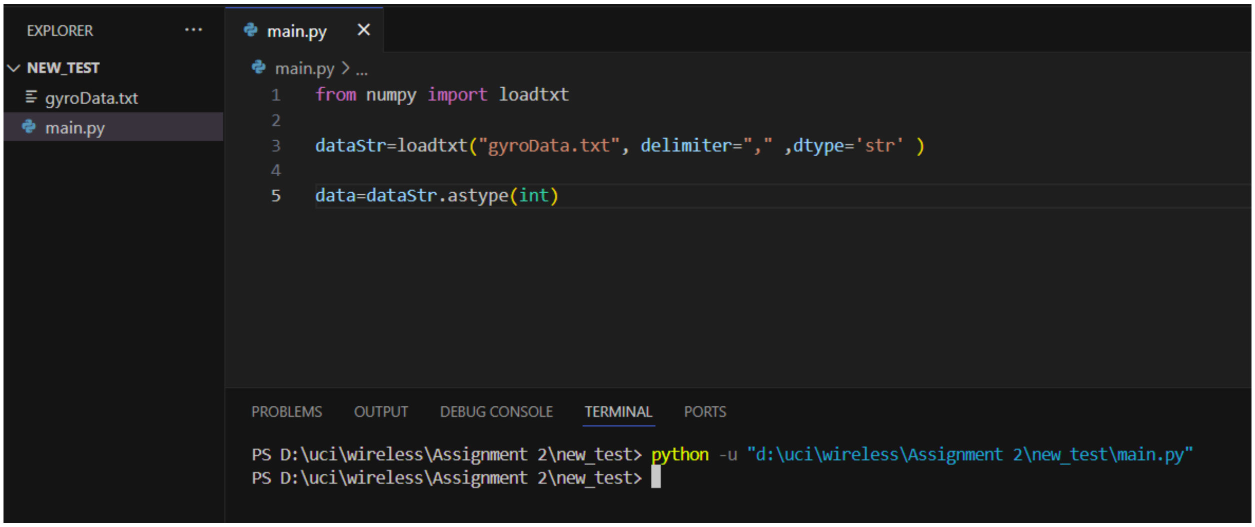
Part1:

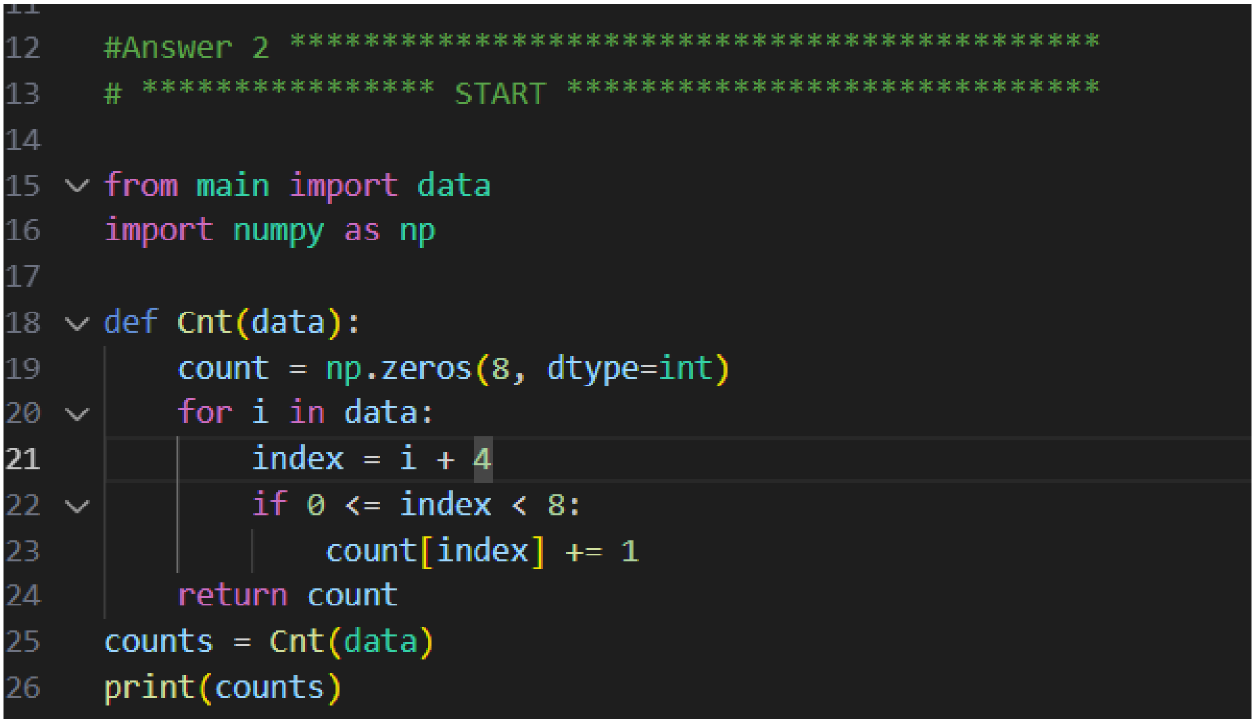
Answer1: Used the mentioned code and it compiled successfully



```python
from numpy import loadtxt

dataStr=loadtxt("gyroData.txt", delimiter="," ,dtype='str' )

data=dataStr.astype(int)
```

```
PS D:\uci\wireless\Assignment 2\new_test> python -u "d:\uci\wireless\Assignment 2\new_test\main.py"
PS D:\uci\wireless\Assignment 2\new_test>
```

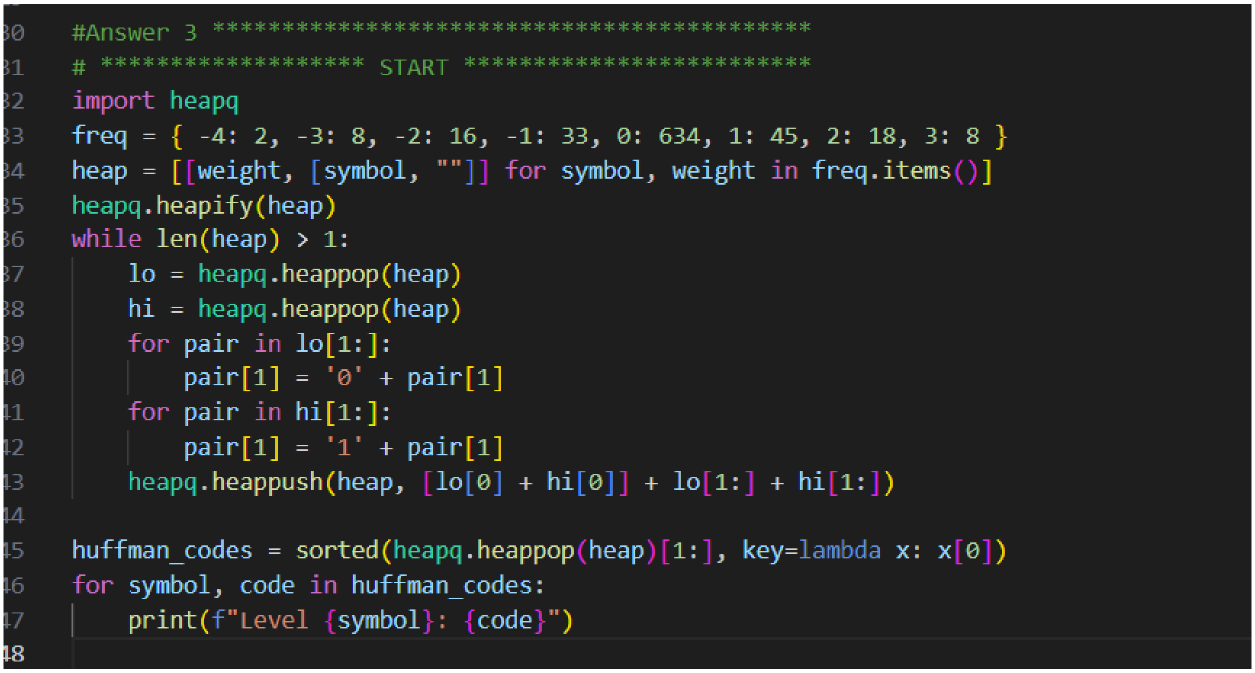
Answer 2:



```python
#Answer 2 ******************************************
# **************** START ****************************

from main import data
import numpy as np

def Cnt(data):
    count = np.zeros(8, dtype=int)
    for i in data:
        index = i + 4
        if 0 <= index < 8:
            count[index] += 1
    return count
counts = Cnt(data)
print(counts)
```

level	-4	-3	-2	-1	0	1	2	3
number	2	8	16	33	634	45	18	8

Answer 3:



```python
#Answer 3 *******************************************
# ******************* START *************************
import heapq
freq = { -4: 2, -3: 8, -2: 16, -1: 33, 0: 634, 1: 45, 2: 18, 3: 8 }
heap = [[weight, [symbol, ""]] for symbol, weight in freq.items()]
heapq.heapify(heap)
while len(heap) > 1:
    lo = heapq.heappop(heap)
    hi = heapq.heappop(heap)
    for pair in lo[1:]:
        pair[1] = '0' + pair[1]
    for pair in hi[1:]:
        pair[1] = '1' + pair[1]
    heapq.heappush(heap, [lo[0] + hi[0]] + lo[1:] + hi[1:])

huffman_codes = sorted(heapq.heappop(heap)[1:], key=lambda x: x[0])
for symbol, code in huffman_codes:
    print(f"Level {symbol}: {code}")
```

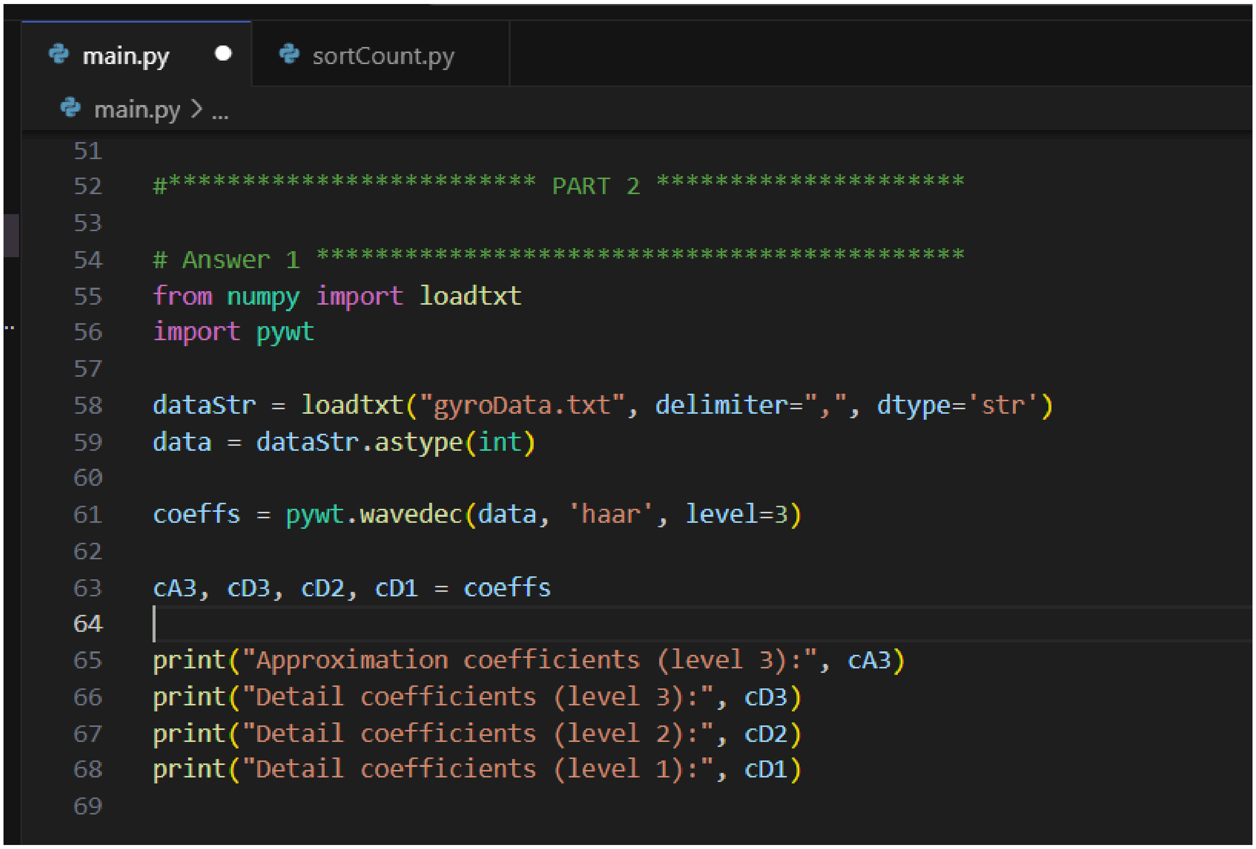
Answer 4:

level	-4	-3	-2	-1	0	1	2	3
bit representation	1111111	1111110	11110	110	0	10	1110	111110

Part2:

Answer 1:

Code:



```python

#************************* PART 2 *********************

# Answer 1 ********************************************
from numpy import loadtxt
import pywt

dataStr = loadtxt("gyroData.txt", delimiter=",", dtype='str')
data = dataStr.astype(int)

coeffs = pywt.wavedec(data, 'haar', level=3)

cA3, cD3, cD2, cD1 = coeffs

print("Approximation coefficients (level 3):", cA3)
print("Detail coefficients (level 3):", cD3)
print("Detail coefficients (level 2):", cD2)
print("Detail coefficients (level 1):", cD1)

```

Result:

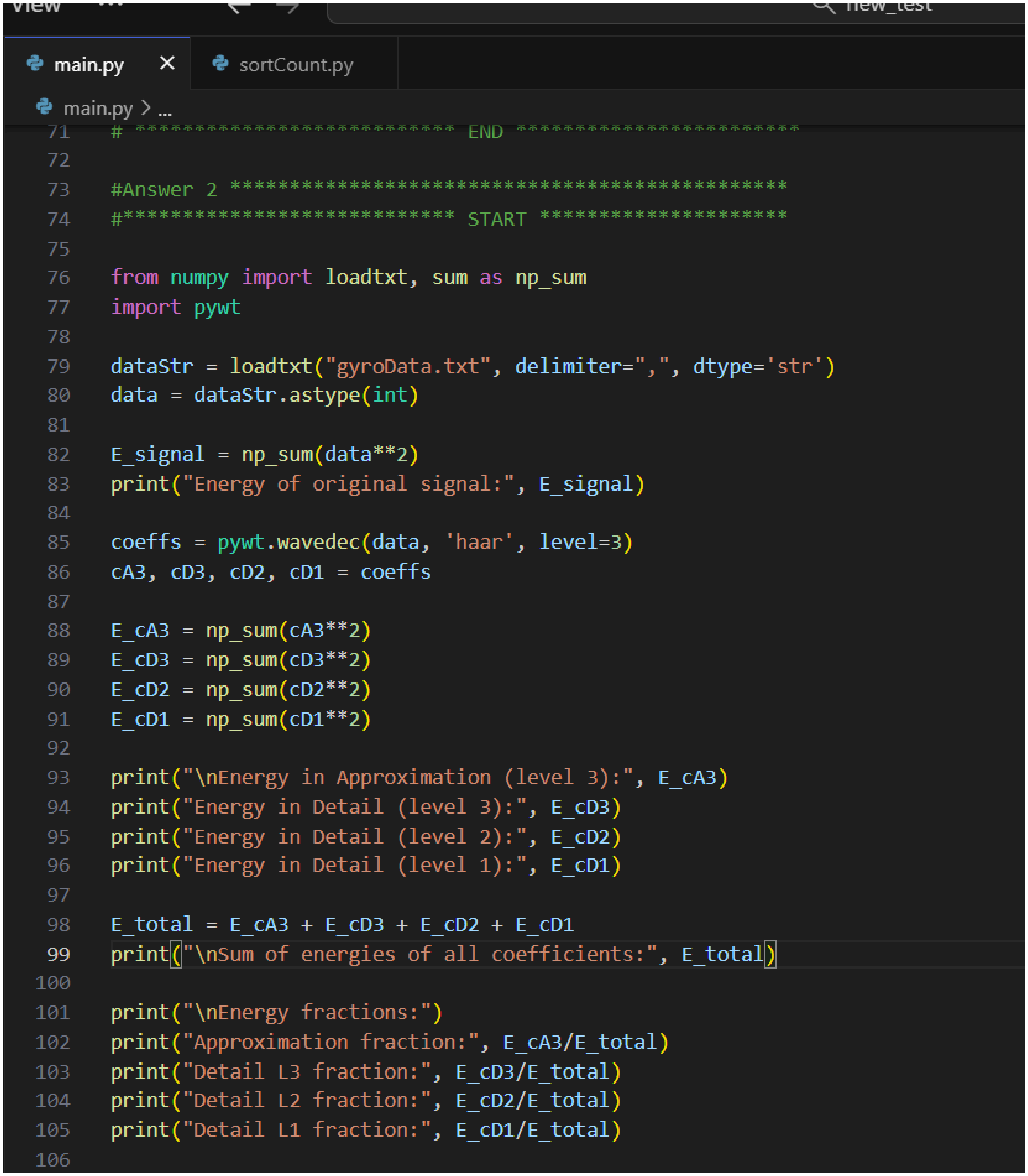
```
Approximation coefficients (level 3): [ 0.00000000e+00  0.00000000e+00  7.07106781e-01
0.00000000e+00
  3.53553391e-01 -3.53553391e-01  3.53553391e-01  0.00000000e+00
  0.00000000e+00  7.07106781e-01 -3.53553391e-01 -7.07106781e-01
  0.00000000e+00  3.53553391e-01  0.00000000e+00  0.00000000e+00
  3.53553391e-01 -3.53553391e-01  3.53553391e-01  0.00000000e+00
  0.00000000e+00  0.00000000e+00  0.00000000e+00 -7.07106781e-01
  0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
  0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
  0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
  0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
  0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
 -3.53553391e-01  0.00000000e+00 -3.53553391e-01  1.06066017e+00
 -1.06066017e+00  2.22044605e-16  7.07106781e-01  3.53553391e-01
 -3.53553391e-01  0.00000000e+00  0.00000000e+00  1.41421356e+00
  3.53553391e-01  0.00000000e+00 -7.07106781e-01  0.00000000e+00
  0.00000000e+00  3.53553391e-01 -7.07106781e-01  7.07106781e-01
  0.00000000e+00 -7.07106781e-01 -1.41421356e+00 -1.11022302e-16
  1.06066017e+00 -2.12132034e+00  3.53553391e-01  3.53553391e-01
  1.06066017e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
  0.00000000e+00  0.00000000e+00  0.00000000e+00 -3.53553391e-01
  0.00000000e+00  0.00000000e+00  0.00000000e+00  3.53553391e-01
  3.53553391e-01  3.53553391e-01  0.00000000e+00  0.00000000e+00
  0.00000000e+00  7.07106781e-01  0.00000000e+00  0.00000000e+00
  0.00000000e+00  7.07106781e-01  0.00000000e+00  0.00000000e+00]
```

```
Detail coefficients (level 3): [ 0.          0.          0.70710678  0.         -0.35355339  0.35355339
 0.35355339  0.          0.         -0.70710678 -1.06066017 -0.70710678
 0.         -0.35355339  0.70710678  0.          1.06066017  0.35355339
 0.35355339  0.          0.          0.          0.         -0.70710678
 0.          0.          0.          0.          0.          0.
 0.          0.          0.          0.          0.          0.
 0.          0.          0.          0.          0.          0.
 0.          0.          0.35355339  0.         -0.35355339 -1.76776695
-0.35355339  2.82842712  0.70710678 -0.35355339  0.35355339  0.
 0.         -1.41421356 -2.47487373 -2.12132034 -0.70710678  0.
-2.12132034  0.35355339  0.70710678  0.70710678  0.          1.41421356
 2.82842712  0.70710678 -2.47487373 -2.12132034  1.06066017  3.18198052
 1.06066017 -1.41421356  0.          0.          0.          0.
 0.          0.35355339  0.          0.          0.         -0.35355339
-0.35355339  0.35355339  0.          0.          0.         -0.70710678
 0.          0.          0.         -0.70710678  0.          0.        ]
```

```
Detail coefficients (level 2): [ 0.   0.   0.   0.  -1.   0.   0.   0.   0.  -0.5  0.   0.5  0.5  0.
 0.   0.   0.   0.   0.  -1.  -1.  -0.5  0.   0.   0.   0.   0.   0.5
 0.5 -0.5  0.   0.   0.  -0.5  0.   0.5 -0.5 -1.   1.   1.   0.   0.
 0.   0.   0.   0.   1.   0.   0.   0.   0.   0.   0.   0.   0.   0.
 0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.
 0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.
 0.   0.   0.   0.   0.   0.5  0.   0.  -0.5  0.   0.5  1.  -2.   2.5
-1.  -4.   1.   0.   0.  -0.5  0.  -0.5  0.   0.   0.   0.   0.  -1.
 0.5 -1.   1.5 -1.5  2.   0.   0.   0.   0.5  0.5  1.5  0.   0.   1.
 0.   0.   0.   0.  -0.5  0.5  1.  -1.   0.5 -3.5  4.  -2.5 -1.   1.
-3.   3.5 -0.5 -2.   3.5 -4.   4.   0.   0.   0.   0.   0.   0.   0.
 0.   0.   0.   0.   0.  -0.5  0.   0.   0.   0.   0.   0.   0.  -0.5
 0.  -0.5  0.5  0.   0.   0.   0.   0.   0.   0.   0.   1.   0.   0.
 0.   0.   0.   0.   0.   1.   0.   0.   0. ]
```

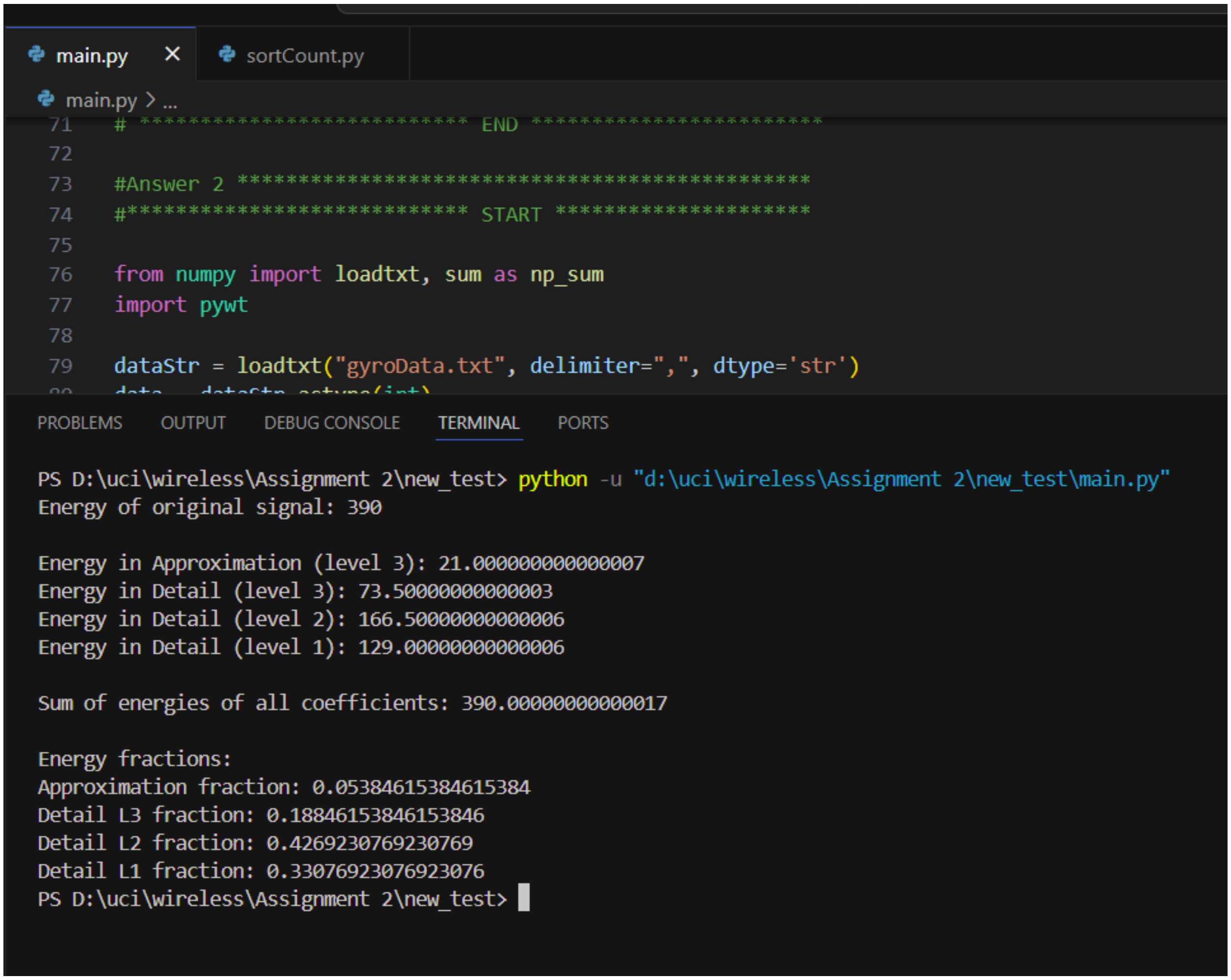
```
Detail coefficients (level 1): [ 0.          0.          0.          0.          0.          0.
  0.          0.          0.          1.41421356  0.          0.
  0.          0.          0.          0.          0.          0.
  0.         -0.70710678  0.          0.          0.          0.70710678
 -0.70710678  0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.
  0.          0.          0.          1.41421356  1.41421356  0.
  0.         -0.70710678  0.70710678 -0.70710678  0.          0.
  0.          0.          0.          0.          0.          0.
 -0.70710678  0.          0.70710678  0.          0.70710678  0.
  0.          0.          0.          0.         -0.70710678  0.70710678
 -0.70710678  0.          0.          0.          0.          0.70710678
  0.          0.70710678 -0.70710678 -0.70710678  0.70710678 -0.70710678
 -0.70710678  0.70710678  0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.
  0.          0.          0.         -1.41421356  0.          0.
  0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.         -0.70710678
  0.          0.          0.          0.         -0.70710678  0.
  0.          0.          0.         -0.70710678 -0.70710678  0.70710678
  0.70710678 -0.70710678  1.41421356  2.12132034 -2.12132034  0.70710678
  1.41421356 -1.41421356  1.41421356  0.          0.          0.
  0.          0.          0.          0.70710678  0.          0.
  0.70710678  0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.
 -0.70710678  0.70710678  0.70710678  0.         -0.70710678  0.70710678
  1.41421356 -0.70710678  0.          0.70710678  0.70710678 -0.70710678
  0.          0.          0.          0.          0.          0.
  0.70710678  1.41421356 -1.41421356 -0.70710678  0.          0.70710678
  0.          0.          0.          0.          0.         -1.41421356
  0.70710678  0.70710678  0.          0.          0.          0.
  0.          0.          0.         -0.70710678  0.70710678 -1.41421356
 -1.41421356  0.          0.         -2.82842712 -0.70710678  0.
 -1.41421356 -2.12132034  2.12132034 -0.70710678 -1.41421356  0.70710678
  1.41421356 -1.41421356 -0.70710678  0.70710678 -1.41421356 -1.41421356
  2.12132034 -1.41421356 -1.41421356  2.12132034  1.41421356 -1.41421356
 -0.70710678  1.41421356 -2.82842712 -1.41421356  0.70710678  0.70710678
 -0.70710678  0.70710678  0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.
 -0.70710678  0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          1.41421356  0.70710678
  0.          0.          0.          0.70710678  0.70710678  0.
  0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.
  0.          0.          1.41421356  0.          0.          0.
  0.          0.          0.          0.        ]
```

Answer 2:
Code



```python
# *************************** END ************************

#Answer 2 ***********************************************
#**************************** START *********************

from numpy import loadtxt, sum as np_sum
import pywt

dataStr = loadtxt("gyroData.txt", delimiter=",", dtype='str')
data = dataStr.astype(int)

E_signal = np_sum(data**2)
print("Energy of original signal:", E_signal)

coeffs = pywt.wavedec(data, 'haar', level=3)
cA3, cD3, cD2, cD1 = coeffs

E_cA3 = np_sum(cA3**2)
E_cD3 = np_sum(cD3**2)
E_cD2 = np_sum(cD2**2)
E_cD1 = np_sum(cD1**2)

print("\nEnergy in Approximation (level 3):", E_cA3)
print("Energy in Detail (level 3):", E_cD3)
print("Energy in Detail (level 2):", E_cD2)
print("Energy in Detail (level 1):", E_cD1)

E_total = E_cA3 + E_cD3 + E_cD2 + E_cD1
print("\nSum of energies of all coefficients:", E_total)

print("\nEnergy fractions:")
print("Approximation fraction:", E_cA3/E_total)
print("Detail L3 fraction:", E_cD3/E_total)
print("Detail L2 fraction:", E_cD2/E_total)
print("Detail L1 fraction:", E_cD1/E_total)

```

Result:



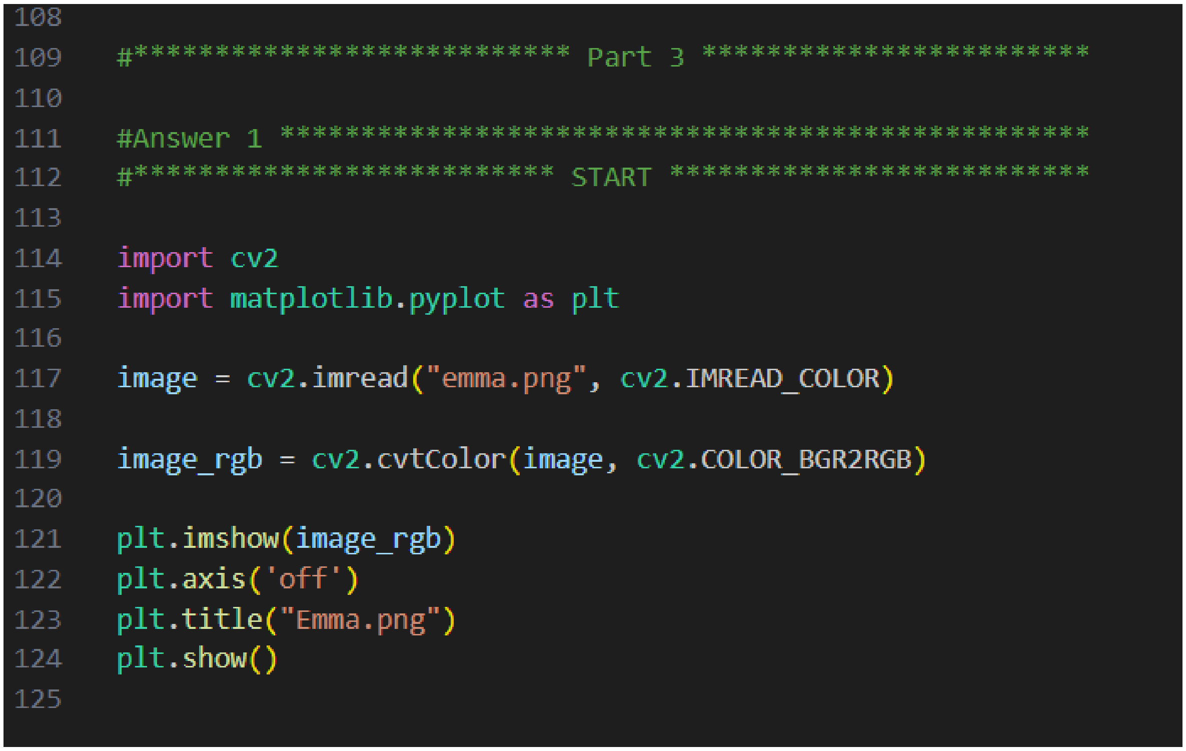
```
# *************************** END ************************

#Answer 2 ***********************************************
#**************************** START *********************

from numpy import loadtxt, sum as np_sum
import pywt

dataStr = loadtxt("gyroData.txt", delimiter=",", dtype='str')
data = dataStr.astype(int)
```

```
PS D:\uci\wireless\Assignment 2\new_test> python -u "d:\uci\wireless\Assignment 2\new_test\main.py"
Energy of original signal: 390

Energy in Approximation (level 3): 21.000000000000007
Energy in Detail (level 3): 73.50000000000003
Energy in Detail (level 2): 166.50000000000006
Energy in Detail (level 1): 129.00000000000006

Sum of energies of all coefficients: 390.00000000000017

Energy fractions:
Approximation fraction: 0.05384615384615384
Detail L3 fraction: 0.18846153846153846
Detail L2 fraction: 0.4269230769230769
Detail L1 fraction: 0.33076923076923076
PS D:\uci\wireless\Assignment 2\new_test> 
```

Part 3:

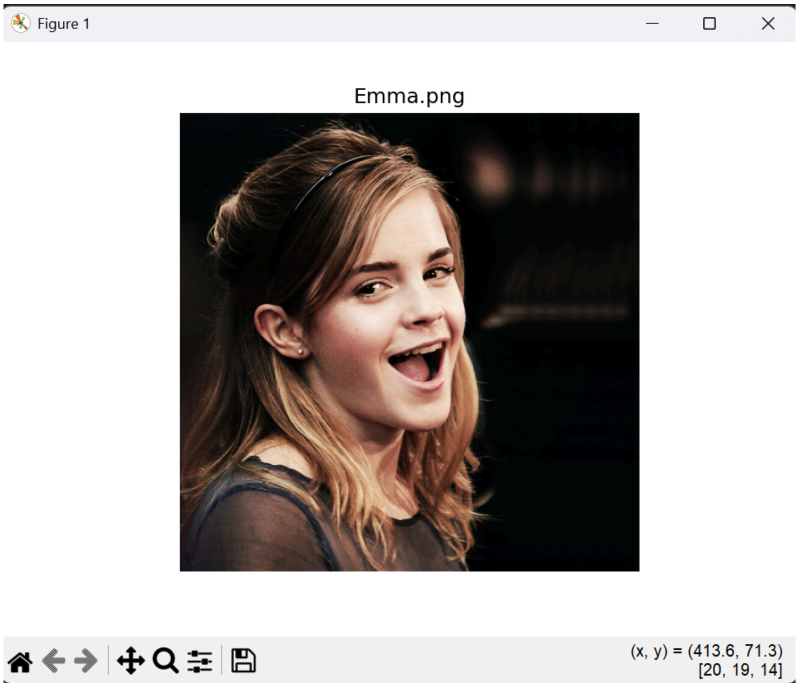
Answer 1:

Code:



```python

#*************************** Part 3 ************************

#Answer 1 **************************************************
#************************** START **************************

import cv2
import matplotlib.pyplot as plt

image = cv2.imread("emma.png", cv2.IMREAD_COLOR)

image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

plt.imshow(image_rgb)
plt.axis('off')
plt.title("Emma.png")
plt.show()

```

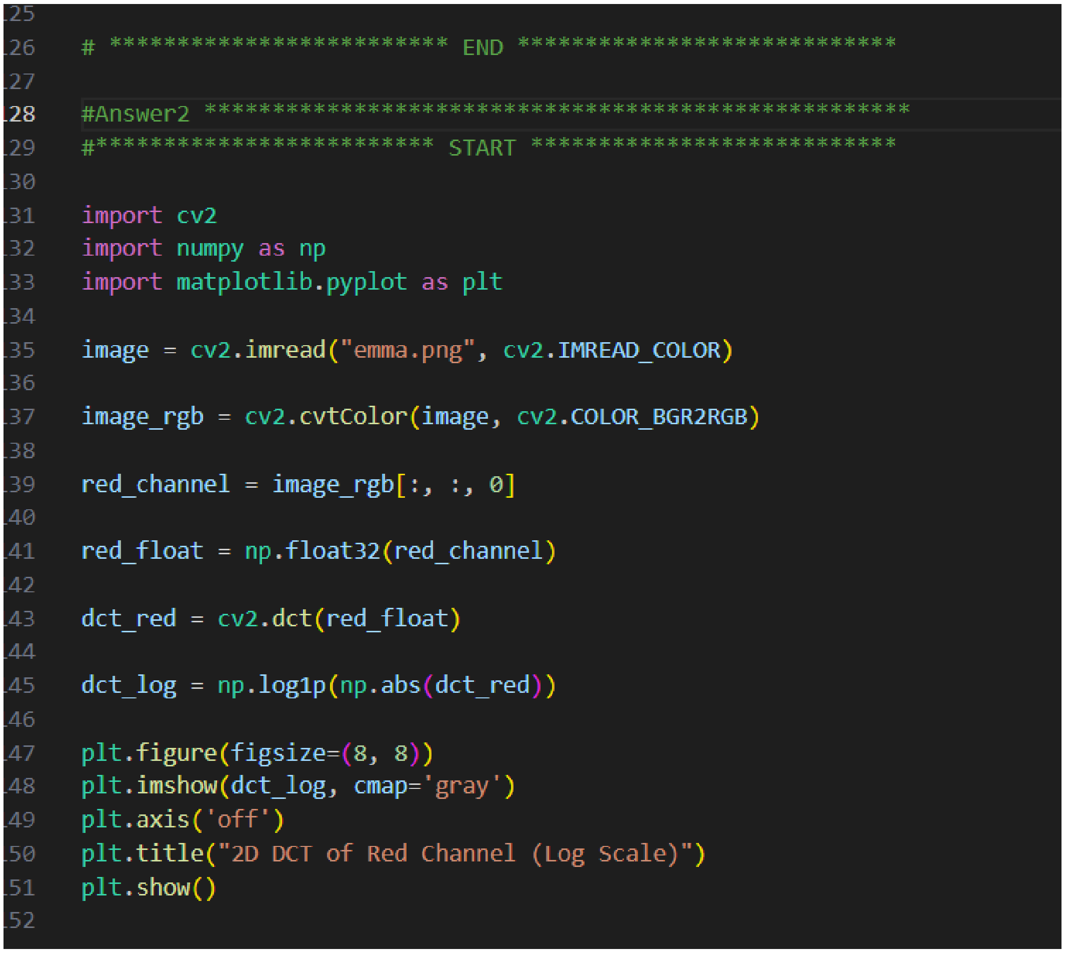
Result:



```
(x, y) = (413.6, 71.3)
[20, 19, 14]
```

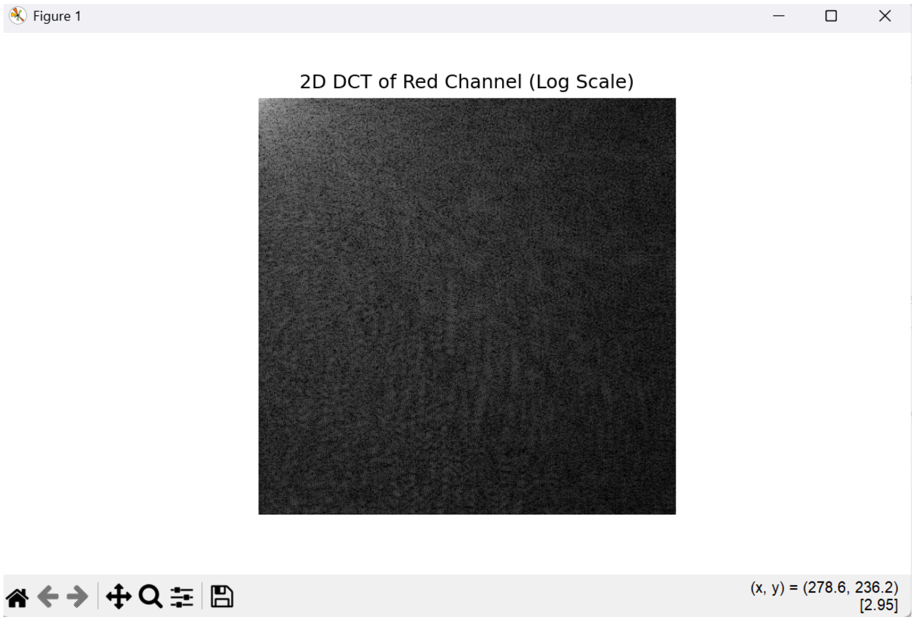
Answer 2:

Code:



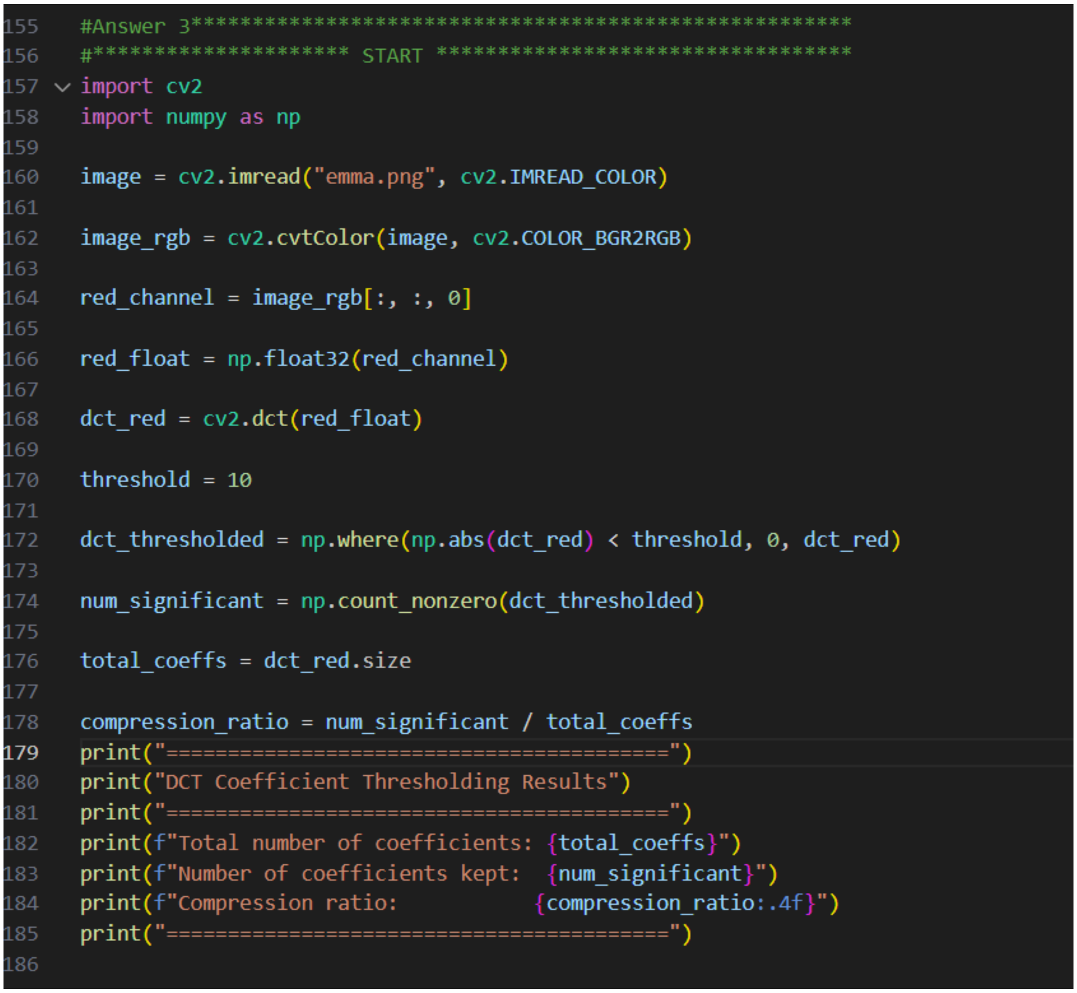
```

# ************************* END ****************************

#Answer2 ****************************************************
#************************* START ***************************

import cv2
import numpy as np
import matplotlib.pyplot as plt

image = cv2.imread("emma.png", cv2.IMREAD_COLOR)

image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

red_channel = image_rgb[:, :, 0]

red_float = np.float32(red_channel)

dct_red = cv2.dct(red_float)

dct_log = np.log1p(np.abs(dct_red))

plt.figure(figsize=(8, 8))
plt.imshow(dct_log, cmap='gray')
plt.axis('off')
plt.title("2D DCT of Red Channel (Log Scale)")
plt.show()

```

Result:



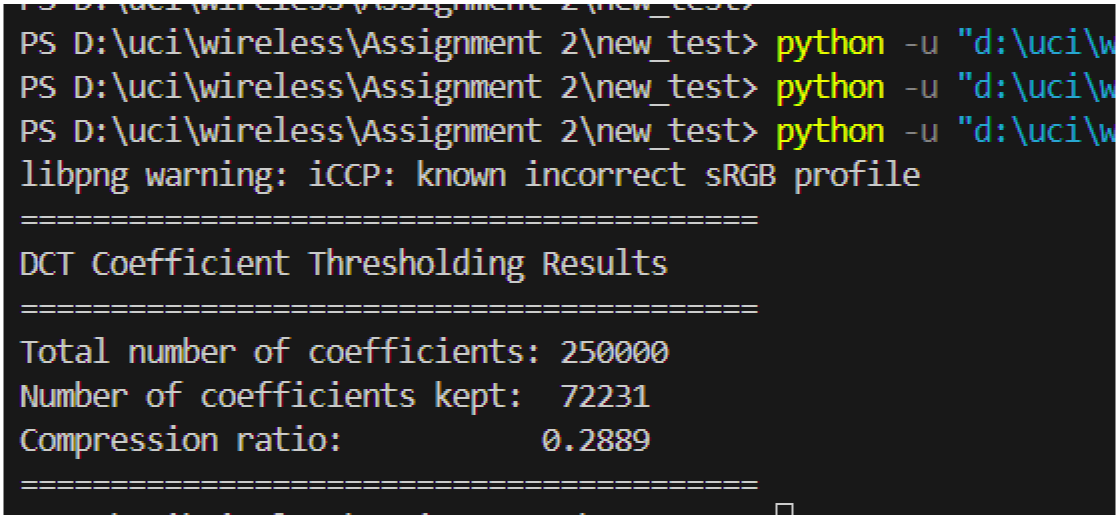
Answer3:

Code:



```python
#Answer 3******************************************************
#********************* START **********************************
import cv2
import numpy as np

image = cv2.imread("emma.png", cv2.IMREAD_COLOR)

image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

red_channel = image_rgb[:, :, 0]

red_float = np.float32(red_channel)

dct_red = cv2.dct(red_float)

threshold = 10

dct_thresholded = np.where(np.abs(dct_red) < threshold, 0, dct_red)

num_significant = np.count_nonzero(dct_thresholded)

total_coeffs = dct_red.size

compression_ratio = num_significant / total_coeffs
print("=========================================")
print("DCT Coefficient Thresholding Results")
print("=========================================")
print(f"Total number of coefficients: {total_coeffs}")
print(f"Number of coefficients kept:  {num_significant}")
print(f"Compression ratio:           {compression_ratio:.4f}")
print("=========================================")

```

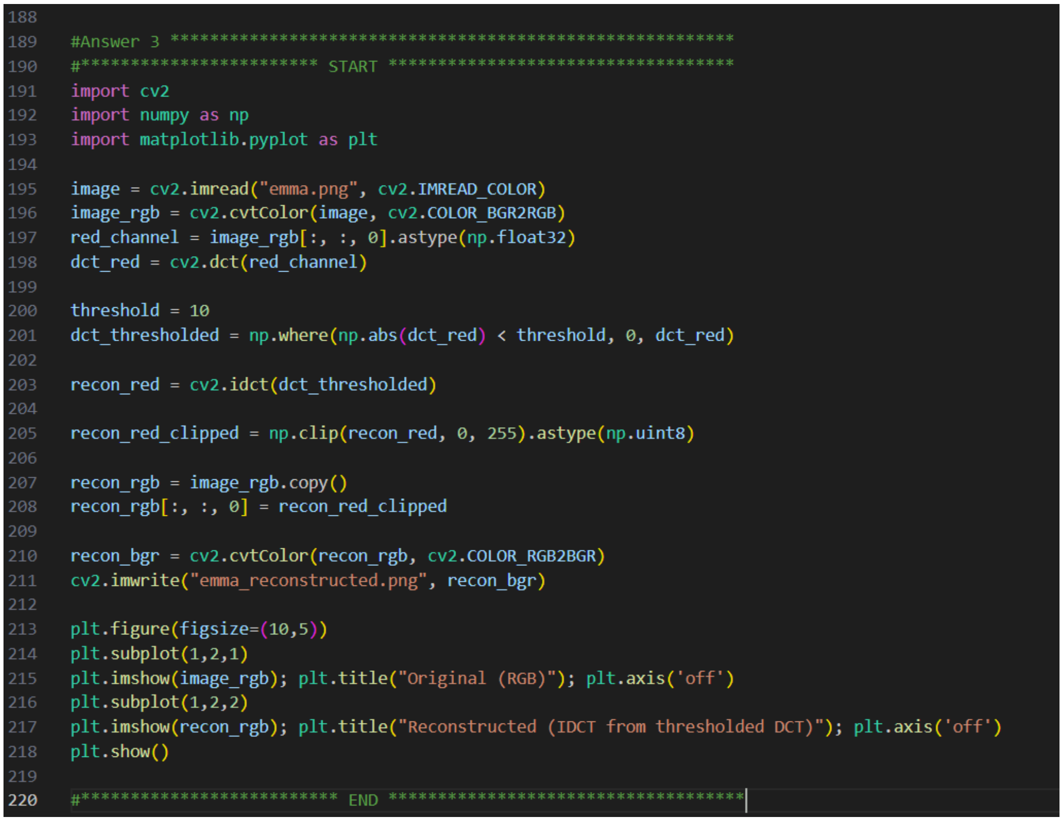
Result:



```
PS D:\uci\wireless\Assignment 2\new_test>
PS D:\uci\wireless\Assignment 2\new_test> python -u "d:\uci\w
PS D:\uci\wireless\Assignment 2\new_test> python -u "d:\uci\w
PS D:\uci\wireless\Assignment 2\new_test> python -u "d:\uci\w
libpng warning: iCCP: known incorrect sRGB profile
=========================================
DCT Coefficient Thresholding Results
=========================================
Total number of coefficients: 250000
Number of coefficients kept:  72231
Compression ratio:           0.2889
=========================================
```

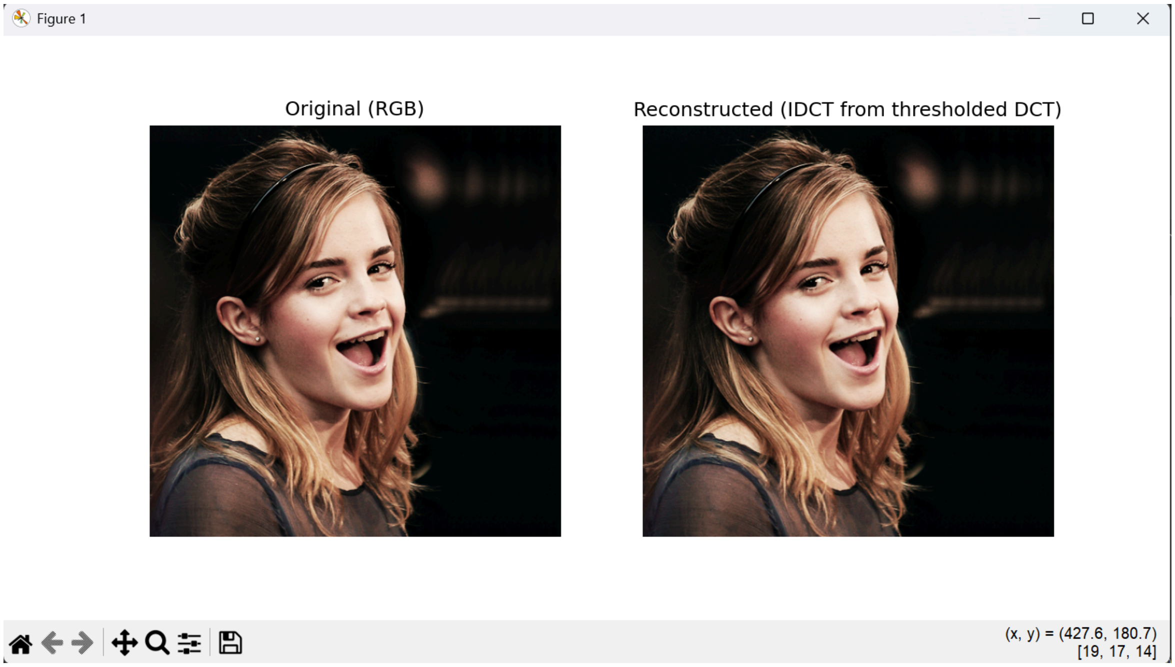
Answer4:

Code:



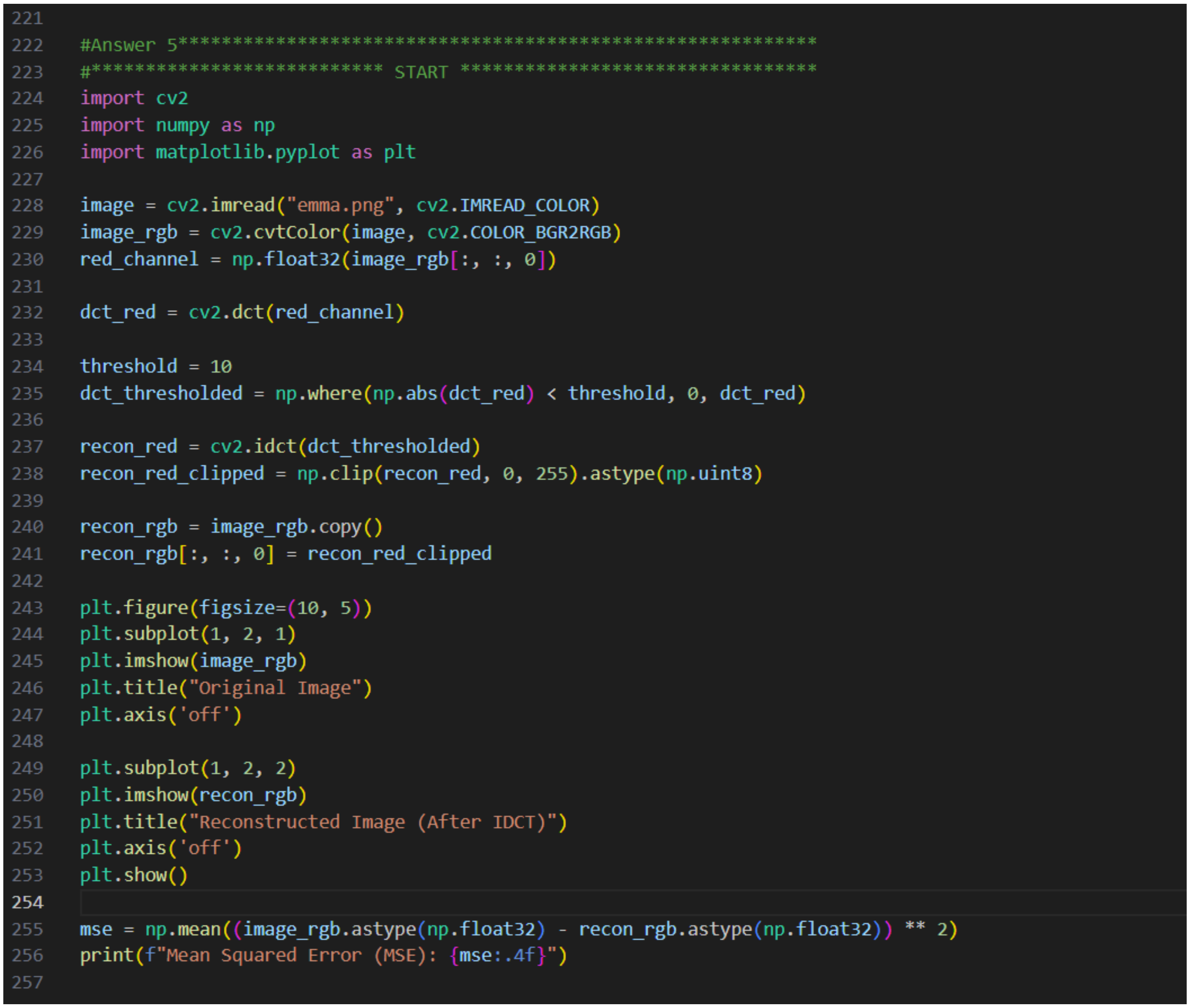
```
#Answer 3 *************************************************************
#************************ START *************************************
import cv2
import numpy as np
import matplotlib.pyplot as plt

image = cv2.imread("emma.png", cv2.IMREAD_COLOR)
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
red_channel = image_rgb[:, :, 0].astype(np.float32)
dct_red = cv2.dct(red_channel)

threshold = 10
dct_thresholded = np.where(np.abs(dct_red) < threshold, 0, dct_red)

recon_red = cv2.idct(dct_thresholded)

recon_red_clipped = np.clip(recon_red, 0, 255).astype(np.uint8)

recon_rgb = image_rgb.copy()
recon_rgb[:, :, 0] = recon_red_clipped

recon_bgr = cv2.cvtColor(recon_rgb, cv2.COLOR_RGB2BGR)
cv2.imwrite("emma_reconstructed.png", recon_bgr)

plt.figure(figsize=(10,5))
plt.subplot(1,2,1)
plt.imshow(image_rgb); plt.title("Original (RGB)"); plt.axis('off')
plt.subplot(1,2,2)
plt.imshow(recon_rgb); plt.title("Reconstructed (IDCT from thresholded DCT)"); plt.axis('off')
plt.show()

#************************** END ***********************************
```

Result:



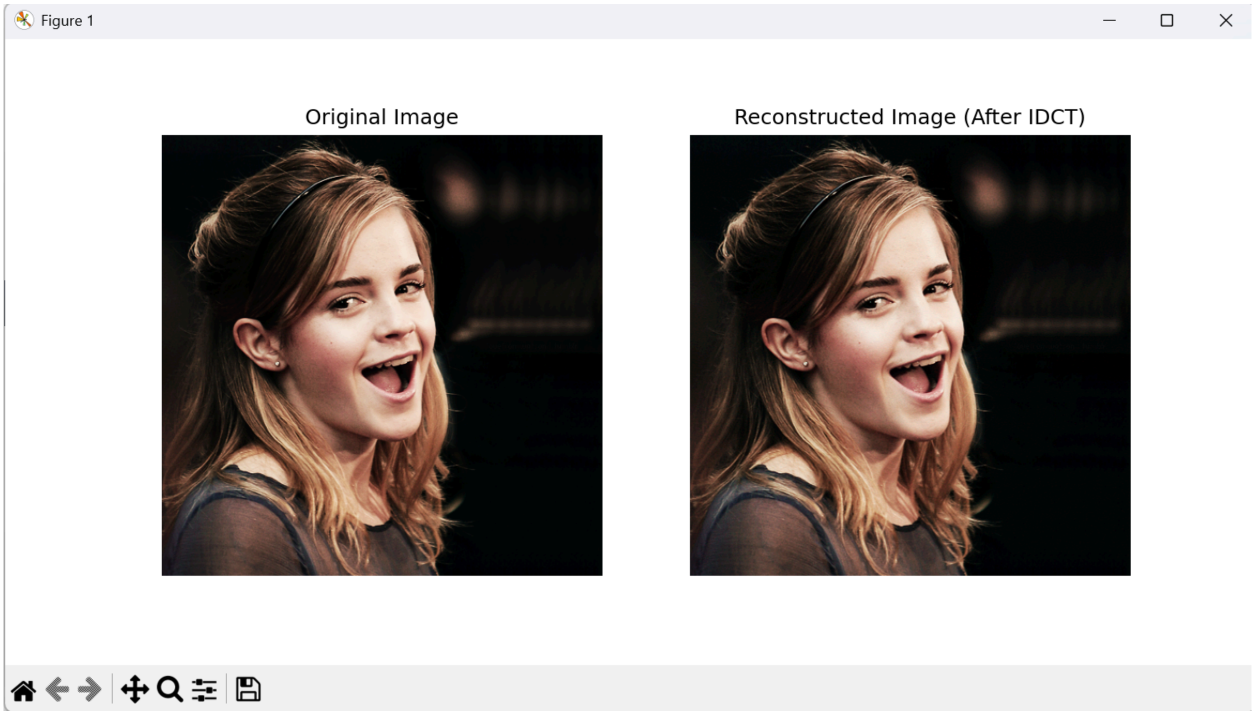
Answer5:

Code:



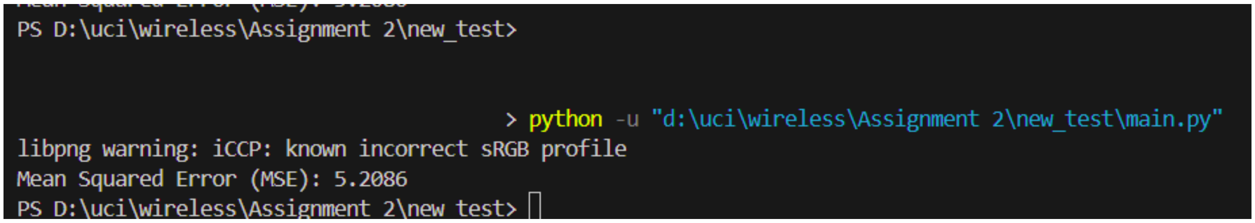
```
#Answer 5***********************************************************
#*************************** START *********************************
import cv2
import numpy as np
import matplotlib.pyplot as plt

image = cv2.imread("emma.png", cv2.IMREAD_COLOR)
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
red_channel = np.float32(image_rgb[:, :, 0])

dct_red = cv2.dct(red_channel)

threshold = 10
dct_thresholded = np.where(np.abs(dct_red) < threshold, 0, dct_red)

recon_red = cv2.idct(dct_thresholded)
recon_red_clipped = np.clip(recon_red, 0, 255).astype(np.uint8)

recon_rgb = image_rgb.copy()
recon_rgb[:, :, 0] = recon_red_clipped

plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.imshow(image_rgb)
plt.title("Original Image")
plt.axis('off')

plt.subplot(1, 2, 2)
plt.imshow(recon_rgb)
plt.title("Reconstructed Image (After IDCT)")
plt.axis('off')
plt.show()

mse = np.mean((image_rgb.astype(np.float32) - recon_rgb.astype(np.float32)) ** 2)
print(f"Mean Squared Error (MSE): {mse:.4f}")

```

Result:



Original Image

Reconstructed Image (After IDCT)



```
Mean Squared Error (MSE): 5.2086
PS D:\uci\wireless\Assignment 2\new_test>


                                       > python -u "d:\uci\wireless\Assignment 2\new_test\main.py"
libpng warning: iCCP: known incorrect sRGB profile
Mean Squared Error (MSE): 5.2086
PS D:\uci\wireless\Assignment 2\new_test>
```