MANGALORE UNIVERSITY

MASTER OF SCIENCE

IN

COMPUTER SCIENCE

23CSP203: IMAGE PROCESSING LAB

SUBMITTED

BY

II SEMESTER MSC

Computer Science Students

SUBMITTED

TO

Dr. B.H. Shekar

Department of Computer Science

Lecturers, In-charge:

1.

2.

Mangalore University

Dept. of Post-Graduate Studies and Research in Computer Science

Mangalagangothri - 574199

IMAGE PROCESSING LAB PROGRAMS		
SL.NO	PROGRAMS	PAGE NO
1.	Write a python program to implement transform(inverse) of an image without built in functions.	4-5
2.	Write a python program to implement an average of 5 images without built in functions.	6-7
3.	Write a python program to calculate histogram of an image with and without built- in function.	8
4.	Write a python program to implement histogram equalization.	9
5.	Write a python program to implement arithmetic operations on two images.	10-11
6.	Write a python program implement bitwise operations on two images.	12-13
7.	Write a python program to implement blurring of an image with built in functions.	14
8.	Write a python program to implement Laplacian sharpening of an image.	15
9.	Write a python program to implement cropping of an image.	16
10.	Write a python program to implement rotation of an image with built in functions.	17
11.	Write a python program to implement image translation with built in function.	18
12.	Write a python program to implement Euclidean distance between two images without using built in functions.	19
13.	Write a python program to implement Canny Edge Detection using built-in functions.	20

1. Write a python program to implement transform(inverse) of an image without built in functions.

```
import matplotlib.pyplot as plt
import cv2
import os

def read_image_file(root_directory):
   filename = input("Enter a image file name :: ")
   filepath = os.path.join(root_directory, filename)
   img = cv2.imread(filepath)
   return img

root_directory = os.getcwd()
original_image = read_image_file(root_directory)

if original_image is None:
   print("Failed to load image")
else:
   rgb_image = cv2.cvtColor(original_image, cv2.COLOR_BGR2RGB)
   gray_image = cv2.cvtColor(original_image, cv2.COLOR_BGR2GRAY)
   _, binary_image = cv2.threshold(gray_image, 127, 255, cv2.THRESH_BINARY)

   negated_rgb_image = 255 - rgb_image
   negated_gray_image = 255 - gray_image
   negated_binary_image = 255 - binary_image


   plt.figure(figsize=(15, 7))

   plt.subplot(2,3,1)
   plt.imshow(rgb_image)
   plt.title("Coloured Image")

   plt.subplot(2,3,2)
   plt.imshow(gray_image, cmap = 'gray')
   plt.title("Grayscale Image")
```

```
plt.subplot(2,3,3)
plt.imshow(binary_image, cmap = 'gray')
plt.title("Binary Image")

plt.subplot(2,3,4)
plt.title("Negation of Coloured Image")
plt.imshow(negated_rgb_image)

plt.subplot(2,3,5)
plt.title("Negation of Grayscale Image")
plt.imshow(negated_gray_image, cmap = 'gray')

plt.subplot(2,3,6)
plt.title("Negation of Binary Image")
plt.imshow(negated_binary_image, cmap = 'gray')

plt.axis('off')
```

2. Write a python program to implement an average of 5 images without built in functions.

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

def read_image_file(root_directory):
    filename = input("Enter a image file name :: ")
    filepath = os.path.join(root_directory, filename)
    img = cv2.imread(filepath)
    return img

def average_images(image_list, target_size):
    avg_image = np.zeros(target_size, dtype=np.float32)
    # Sum all images
    for img in image_list:
        img = cv2.resize(img, target_size).astype(np.float32)
        avg_image += img
    # Divide by the number of images to get the average
    avg_image /= len(image_list)
    # Convert back to uint8
    avg_image = np.clip(avg_image, 0, 255).astype(np.uint8)
    return avg_image

root_directory = os.getcwd()
imgList = []
n = 5
for i in range(n):
    original_image = read_image_file(root_directory)
    if original_image is None:
        print("Failed to load image")
    else:
        gray_image = cv2.cvtColor(original_image, cv2.COLOR_BGR2GRAY)
        imgList.append(gray_image)
```

```
target_size = (512, 512)  #(rows,cols)- used to resize all the images to the same size. can be set to any number.

average_image1 = average_images(imgList, target_size)
imgList.append(average_image1)

plt.figure(figsize=(15, 5))
try:
    for i in range(len(imgList)):
        plt.subplot(1, len(imgList), i+1)
        plt.imshow(imgList[i], cmap = 'gray')
        plt.axis('off')
        if i != len(imgList) - 1 :
            plt.title('Original Image')
        else:
            plt.title('Average Image')
except IndexError:
    print("5 images are not read properly.")
```

3. Write a python program to calculate histogram of an image with and without built-in function.

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

def calculateHist(img):
    hist = np.zeros(256)
    rows, cols = img.shape
    for i in range(rows):
        for j in  range(cols):
            intensity = img[i,j]
            hist[intensity] += 1
    return hist

filepath = input("Enter a image file name :: ")
img = cv2.imread(filepath, 0)

if img is None:
    print("Failed to load image")
else:
    hist1 = cv2.calcHist([img], [0], None, [256], [0, 256])

    plt.figure(figsize=(12, 8))
    plt.subplot(1, 3, 1)
    plt.imshow(img, cmap='gray')
    plt.title('Original Image')

    plt.subplot(1, 3, 2)
    plt.plot(hist1)
    plt.title('Histogram(using built-in)')

    hist2 = calculateHist(img)
    plt.subplot(1, 3, 3)
    plt.plot(hist2)
    plt.title('Histogram(without using built-in)')
```

4. Write a python program to implement histogram equalization.

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

# Read the image
image = cv2.imread('sunflower.jpg', 0)

# Equalize the histogram
equalized_image = cv2.equalizeHist(image)

# Calculate the histogram for the original image
hist_orig = cv2.calcHist([image], [0], None, [256], [0, 256])
# Calculate the histogram for the equalized image
hist_eq = cv2.calcHist([equalized_image], [0], None, [256], [0, 256])

plt.figure(figsize=(12, 8))
# Display the original and equalized  grayscale image
plt.subplot(2, 2, 1)
plt.title("Original Image")
plt.imshow(image, cmap='gray')
plt.axis('off')

plt.subplot(2, 2, 2)
plt.title("Equalized Image")
plt.imshow(equalized_image, cmap='gray')
plt.axis('off')

# Display the histogram for the original and equalized image
plt.subplot(2, 2, 3)
plt.title("Histogram (Original)")
plt.plot(hist_orig)

plt.subplot(2, 2, 4)
plt.title("Histogram (Equalized)")
plt.plot(hist_eq)
```

5. Write a python program to implement arithmetic operations on two images.

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

# Load images
image1 = cv2.imread('sunflower.jpg')
image2 = cv2.imread('f1.jpg')

#Convert the images to RGB
rgb_img1 = cv2.cvtColor(image1, cv2.COLOR_BGR2RGB)
rgb_img2 = cv2.cvtColor(image2, cv2.COLOR_BGR2RGB)
rows, cols = image2.shape[:2]

# Ensure the images are of the same size
rgb_img1 = cv2.resize(rgb_img1, (cols,rows))

added_image = cv2.add(rgb_img1, rgb_img2)
sub_image = cv2.subtract(rgb_img1, rgb_img2)
multiplied_image = cv2.multiply(rgb_img1, rgb_img2)

image1_float = rgb_img1.astype(np.float32)
image2_float = rgb_img2.astype(np.float32) + 1  # Adding 1 to avoid division by zero

divided_image = cv2.divide(image1_float, image2_float)

# Convert back to uint8
divided_image = cv2.normalize(divided_image, None, 0, 255, cv2.NORM_MINMAX)
divided_image = divided_image.astype(np.uint8)

plt.figure(figsize = (8,12))
plt.subplot(3,2,1)
plt.imshow(rgb_img1)
plt.title('Original Image 1')

plt.subplot(3,2,2)
plt.imshow(rgb_img2)
```

```
plt.title('Original Image 2')

plt.subplot(3,2,3)
plt.imshow(added_image)
plt.title('Added Image')

plt.subplot(3,2,4)
plt.imshow(sub_image)
plt.title('Subtracted Image')

plt.subplot(3,2,5)
plt.imshow(multiplied_image)
plt.title('Multiplied Image')

plt.subplot(3,2,6)
plt.imshow(divided_image)
plt.title('Divided Image')
```

6. Write a python program implement bitwise operations on two images.

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

# Load two images
image1 = cv2.imread('f1.jpg')
image2 = cv2.imread('dog.jpg')

#Convert the images to RGB
rgb_img1 = cv2.cvtColor(image1, cv2.COLOR_BGR2RGB)
rgb_img2 = cv2.cvtColor(image2, cv2.COLOR_BGR2RGB)

# Resize images to the same dimensions if needed
rgb_img2 = cv2.resize(rgb_img2, (rgb_img1.shape[1], rgb_img1.shape[0]))

# Perform bitwise AND operation
bitwise_and = cv2.bitwise_and(rgb_img1, rgb_img2)

# Perform bitwise OR operation
bitwise_or = cv2.bitwise_or(rgb_img1, rgb_img2)

# Perform bitwise XOR operation
bitwise_xor = cv2.bitwise_xor(rgb_img1, rgb_img2)

# Perform bitwise NOT operation
bitwise_not_img1 = cv2.bitwise_not(rgb_img1)
bitwise_not_img2 = cv2.bitwise_not(rgb_img2)

# Display results
plt.figure(figsize = (8,15))
plt.subplot(4,2,1)
plt.imshow(rgb_img1)
plt.title('Original Image')

plt.subplot(4,2,2)
plt.imshow(rgb_img2)
```

```
plt.title('Original Image')

plt.subplot(4,2,3)
plt.imshow(bitwise_and)
plt.title('Bitwise AND')

plt.subplot(4,2,4)
plt.imshow(bitwise_or)
plt.title('Bitwise OR')

plt.subplot(4,2,5)
plt.imshow(bitwise_not_img1)
plt.title('Bitwise NOT Image 1')

plt.subplot(4,2,6)
plt.imshow(bitwise_not_img2)
plt.title('Bitwise NOT Image 2')

plt.subplot(4,2,7)
plt.imshow(bitwise_xor)
plt.title('Bitwise XOR')
```

7. Write a python program to implement blurring of an image with built in functions.

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

image =  cv2.imread('sunflower.jpg')
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

blurred_img = cv2.GaussianBlur(image_rgb, (15, 15), 0)

plt.figure(figsize=(7, 4))

plt.subplot(1, 2, 1)
plt.imshow(image_rgb)
plt.title('Original Image')

plt.subplot(1, 2, 2)
plt.imshow(blurred_img)
plt.title('Blurred Image')
```

8. Write a python program to implement Laplacian sharpening of an image.

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

# Read the image in grayscale
image = cv2.imread('flower.jpg')
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

# Apply the Laplacian operator
laplacian = cv2.Laplacian(image_rgb, cv2.CV_64F)

# Since the result might have negative values, we need to convert it to a suitable format
laplacian = np.uint8(np.absolute(laplacian))

# Sharpen the image by adding the Laplacian result to the original image
sharpened_image = cv2.addWeighted(image_rgb, 1.0, laplacian, -1.0, 0)

# Display the original and sharpened images
plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)
plt.title("Original Image")
plt.imshow(image_rgb)
plt.axis('off')

plt.subplot(1, 2, 2)
plt.title("Sharpened Image")
plt.imshow(sharpened_image)
plt.axis('off')

plt.tight_layout()
plt.show()
```

9. Write a python program to implement cropping of an image.

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

def crop_image(image, x, y, w, h):
    # Extract the region of interest (ROI) from the input image
    cropped_image = image[y:y+h, x:x+w]

    return cropped_image

# Load the input image
image = cv2.imread('dog.jpg')

# Convert BGR image to RGB
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

# Define the region of interest (ROI) for cropping
x = 100   # X-coordinate of the top-left corner of the ROI
y = 100   # Y-coordinate of the top-left corner of the ROI
w = 200   # Width of the ROI
h = 200   # Height of the ROI

# Crop the image using the custom function
cropped_image = crop_image(image_rgb, x, y, w, h)
cv2.imwrite('cropped_image.jpg',cv2.cvtColor(cropped_image, cv2.COLOR_RGB2BGR))

# Display the original and cropped images
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.imshow(image_rgb)
plt.title('Original Image')

plt.subplot(1, 2, 2)
plt.imshow(cropped_image)
plt.title('Cropped Image')
```

10. Write a python program to implement rotation of an image with built in functions.

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

img = cv2.imread('sunflower.jpg')
rgb_img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

rows, cols = img.shape[:2]
M = np.float32([[1, 0, 0], [0, -1, rows], [0, 0, 1]])
img_rotation = cv2.warpAffine(rgb_img,cv2.getRotationMatrix2D((cols/2, rows/2),30, 0.6),(cols,
rows))
cv2.imwrite('rotation_image.jpg', cv2.cvtColor(img_rotation, cv2.COLOR_RGB2BGR))

plt.figure(figsize = (10, 13))
plt.subplot(1,2,1)
plt.imshow(rgb_img)
plt.title("Original")

plt.subplot(1,2,2)
plt.imshow(img_rotation)
plt.title("Rotated")
```

11. Write a python program to implement image translation with built in function.

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

img = cv2.imread('dog.jpg')
rgb_img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

rows, cols = img.shape[:2]
M = np.float32([[1, 0, 100], [0, 1, 50]])
translated_img = cv2.warpAffine(rgb_img, M, (cols, rows))
cv2.imwrite('translated_image.jpg', cv2.cvtColor(translated_img, cv2.COLOR_RGB2BGR))

plt.figure(figsize = (10, 13))
plt.subplot(1,2,1)
plt.imshow(rgb_img)
plt.title("Original")

plt.subplot(1,2,2)
plt.imshow(translated_img)
plt.title("translated_img")
```

12. Write a python program to implement Euclidean distance between two images without using built in functions.

```
import math
import cv2
import numpy as np

img1 = cv2.imread("sunflower.jpg", 0)
img2 = cv2.imread("flower.jpg", 0)

rows1,cols1 = img1.shape

# Resize img2 to match img1 if dimensions are different
if img1.shape != img2.shape:
   img2 = cv2.resize(img2, (cols1, rows1))

# Convert images to float64 to prevent overflow
img1 = img1.astype(np.float64)
img2 = img2.astype(np.float64)

sum1 = 0
for i in range(rows1):
   for j in range(cols1):
      sum1 += (img1[i,j] - img2[i,j]) ** 2
ED = math.sqrt(sum1)

print("Eucidean distance :: ",ED)
```

13. Write a python program to implement Canny Edge Detection using built-in functions.

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

# Read image.
img = cv2.imread('sunflower.jpg')
# Convert BGR image to RGB
image_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

# Apply Canny edge detection
edges = cv2.Canny(image= image_rgb, threshold1=100, threshold2=700)

# Create subplots
plt.figure(figsize=(7, 4))

# Plot the original image
plt.subplot(1, 2, 1)
plt.imshow(image_rgb)
plt.title('Original Image')

# Plot the blurred image
plt.subplot(1, 2, 2)
plt.imshow(edges)
plt.title('Image edges')
```

14. Write a python program to implement Laplacian of Gaussian edge detection using built-in functions.

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

# Load the image
image = cv2.imread('flower.jpg', 0)

# Apply Gaussian Blur
blurred = cv2.GaussianBlur(image, (5, 5), 0)

# Apply the Laplacian operator
laplacian = cv2.Laplacian(blurred, cv2.CV_64F)

# Convert the Laplacian to an 8-bit image
laplacian_8u = cv2.convertScaleAbs(laplacian)

# Display the results
plt.figure(figsize=(15, 8))

plt.subplot(1, 3, 1)
plt.title('Original Image')
plt.imshow(image, cmap='gray')


plt.subplot(1, 3, 2)
plt.title('Gaussian Blurred Image')
plt.imshow(blurred, cmap='gray')


plt.subplot(1, 3, 3)
plt.title('Laplacian of Gaussian')
plt.imshow(laplacian_8u, cmap='gray')
```

15. Write a python program to implement Difference of Gaussian edge detection using built-in functions.

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

# Load the image
image = cv2.imread('flower.jpg', 0)

# Apply Gaussian Blurs with different sigma values
blur1 = cv2.GaussianBlur(image, (5, 5), 1)
blur2 = cv2.GaussianBlur(image, (5, 5), 2)
# Subtract the two blurred images
dog = blur1 - blur2
# Normalize the DoG image to the range 0-255
dog = cv2.normalize(dog, None, 0, 255, cv2.NORM_MINMAX)
dog = np.uint8(dog)

# Display the results
plt.figure(figsize=(10, 8))
plt.subplot(2, 2, 1)
plt.title('Original Image')
plt.imshow(image, cmap='gray')

plt.subplot(2, 2, 2)
plt.title('Gaussian Blur 1 (sigma=1)')
plt.imshow(blur1, cmap='gray')

plt.subplot(2, 2, 3)
plt.title('Gaussian Blur 2 (sigma=2)')
plt.imshow(blur2, cmap='gray')

plt.subplot(2, 2, 4)
plt.title('DoG Segmentation')
plt.imshow(dog, cmap='gray')
```

16. Write a python program to implement global thresholding without built in functions.

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

def global_thresholding(image, threshold_value):
    binary_image = np.zeros_like(image)
    height, width = image.shape
    for i in range(height):
        for j in range(width):
            if image[i, j] > threshold_value:
                binary_image[i, j] = 255
            else:
                binary_image[i, j] = 0
    return binary_image

image_path = 'sunflower.jpg'
manual_threshold = 127

# Load the image
original_image = cv2.imread(image_path, 0)

# Apply global thresholding
binary_image = global_thresholding(original_image, manual_threshold)

plt.figure(figsize=[10, 5])
plt.subplot(1, 2, 1)
plt.title("Original Grayscale Image")
plt.imshow(original_image, cmap='gray')
plt.axis('off')

plt.subplot(1, 2, 2)
plt.title("Binary Image after Thresholding")
plt.imshow(binary_image, cmap='gray')
plt.axis('off')
```

17. Write a python program to implement local thresholding with built in functions.

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

# Block size for local thresholding (must be odd and greater than 1)
block_size = 16  # You can adjust this value as needed
if block_size % 2 == 0:  # Ensure block size is odd
    block_size += 1

# Load the image
image_path = 'sunflower.jpg'
original_image = cv2.imread(image_path, 0)

C = 2

# Apply local thresholding
binary_image =
cv2.adaptiveThreshold(original_image,255,cv2.ADAPTIVE_THRESH_MEAN_C,cv2.THRESH_BINA
RY,block_size,C)

# Display the images
plt.figure(figsize=[10, 5])

plt.subplot(1, 2, 1)
plt.title("Original Grayscale Image")
plt.imshow(original_image, cmap='gray')
plt.axis('off')

plt.subplot(1, 2, 2)
plt.title("Binary Image after Local Thresholding")
plt.imshow(binary_image, cmap='gray')
plt.axis('off')
```

18. Write a python program to implement Sobel edge detection using built – in function.

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

# Read the image in grayscale
image = cv2.imread('sunflower.jpg', 0)

# Apply Sobel edge detection
sobel_x = cv2.Sobel(image, cv2.CV_64F, 1, 0, ksize=3)
sobel_y = cv2.Sobel(image, cv2.CV_64F, 0, 1, ksize=3)
sobel_combined = np.sqrt(sobel_x**2 + sobel_y**2)

# Display the original and Sobel edge-detected images
plt.figure(figsize=[10, 5])
plt.subplot(1, 2, 1)
plt.imshow(image, cmap='gray')
plt.title('Original Image')
plt.axis('off')

plt.subplot(1, 2, 2)
plt.imshow(sobel_combined, cmap='gray')
plt.title('Sobel Edge-detected Image')
plt.axis('off')
plt.show()
```

19. Write a python program to implement Robert edge detection using built-in functions.

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

# Read the image in grayscale
image = cv2.imread('sunflower.jpg', 0)
# Convert the image to float32 for precision in computations
image = np.float32(image)

# Define the Roberts Cross kernels
kernel_x = np.array([[1, 0],[0, -1]], dtype=np.float32)
kernel_y = np.array([[0, 1],[-1, 0]], dtype=np.float32)

# Apply convolution with the Roberts Cross kernels
roberts_x = cv2.filter2D(image, -1, kernel_x)
roberts_y = cv2.filter2D(image, -1, kernel_y)
# Calculate the magnitude of gradients
roberts_combined = np.sqrt(roberts_x**2 + roberts_y**2)

# Normalize the result to the range [0, 255]
roberts_combined = cv2.normalize(roberts_combined, None, 0, 255, cv2.NORM_MINMAX)
roberts_combined = np.uint8(roberts_combined)

# Display the original and Roberts Cross edge-detected images
plt.figure(figsize=[10, 5])
plt.subplot(1, 2, 1)
plt.imshow(image, cmap='gray')
plt.title('Original Image')
plt.axis('off')

plt.subplot(1, 2, 2)
plt.imshow(roberts_combined, cmap='gray')
plt.title('Roberts Cross Edge-detected Image')
plt.axis('off')
```

20. Write a python program to implement Prewitt edge detection using built-in functions.

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

# Read the image in grayscale
image = cv2.imread('sunflower.jpg', 0)
# Convert the image to float32 for precision in computations
image = np.float32(image)

# Define the Prewitt kernels
kernel_x = np.array([[-1, 0, 1],[-1, 0, 1],[-1, 0, 1]], dtype=np.float32)
kernel_y = np.array([[-1, -1, -1],[0, 0, 0],[1, 1, 1]], dtype=np.float32)
# Apply convolution with the Prewitt kernels
prewitt_x = cv2.filter2D(image, -1, kernel_x)
prewitt_y = cv2.filter2D(image, -1, kernel_y)

# Calculate the magnitude of gradients
prewitt_combined = np.sqrt(prewitt_x**2 + prewitt_y**2)

# Normalize the result to the range [0, 255]
prewitt_combined = cv2.normalize(prewitt_combined, None, 0, 255, cv2.NORM_MINMAX)
prewitt_combined = np.uint8(prewitt_combined)

# Display the original and Prewitt edge-detected images
plt.figure(figsize=[10, 5])
plt.subplot(1, 2, 1)
plt.imshow(image, cmap='gray')
plt.title('Original Image')
plt.axis('off')

plt.subplot(1, 2, 2)
plt.imshow(prewitt_combined, cmap='gray')
plt.title('Prewitt Edge-detected Image')
plt.axis('off')
```

21. Write a python program to implement mean, median, standard deviation and correlation coefficient of an image.

```
import cv2
import numpy as np

# Read the image
img = cv2.imread("img2.jpg", 0)

# Calculate mean, median, standard deviation and correlation coefficient
mean_value = np.mean(img)
median_value = np.median(img)
std_value = np.std(img)
correlation_coefficient = np.corrcoef(img)[0, 1]

# Display results
print("Mean: {:.2f}".format(mean_value))
print("Median: {:.2f}".format(median_value))
print("Standard Deviation: {:.2f}".format(std_value))
print("Correlation Coefficient: {:.2f}".format(correlation_coefficient))
```