**SORU #1**

*import* pandas

*import* matplotlib.pyplot *as* plt

data = pandas.read\_csv('covid\_data.csv')

df = pandas.DataFrame(*data*=data, *columns*=['gender', 'age', 'death', 'country'])

*# a)Yaşları 50 ile 60 arasında olan kadın(female) hasta sayısı nedir*

dict1 = {'female': [], 'male': []}

*for* gender, age *in* zip(df.gender, df.age):

*if* gender == 'female' and (age > 50 and age < 60):

        dict1["female"].append(age)

\_1a = len(dict1['female'])

print(\_1a)

*# b) Ölen ve yaşayan hastaların yaş ortalaması nedir*

dead\_count = 0

alive\_count = 0

*for* death *in* df.death:

*if* death == '1':

        dead\_count += 1

*elif* death == '0':

        alive\_count += 1

print(dead\_count)

print(alive\_count)

*# c) Ölen kadın  ve Ölen erkeklerin yaş ortalaması nedir*

dict2 = {'female': [], 'male': []}

*for* gender, age, dead *in* zip(df.gender, df.age, df.death):

*if* gender == 'female' and dead == '1':

        dict2["female"].append(age)

*elif* gender == 'male' and dead == '1':

        dict2["male"].append(age)

*# female avegare*

female\_sum = 0

*for* i *in* dict2['female']:

    female\_sum += i

female\_avg = female\_sum / len(dict2['female'])

print(female\_avg)

*# male average*

male\_sum = 0

*for* i *in* dict2['male']:

    male\_sum += i

male\_avg = male\_sum / len(dict2['male'])

print(male\_avg)

*# d)20 yaşın altında Çin de (China) bulunan hastaların sayısı nedir*

count = 0

*for* age, country *in* zip(df.age, df.country):

*if* age < 20:

        count += 1

print(count)

*# e) 50 yaşından büyük Çinde ölen hastaların , 50 yaşından büyük Almanyada ölen hastalara oranı nedir?*

*# NOT: kaynaklardaki csv dosyasina gore Almanya'da olen hasta sayisi olmadigi icin degerleri manuel olarak degistirdim.*

ch\_count = 0

de\_count = 0

*for* age, country, death *in* zip(df.age, df.country, df.death):

*if* age > 50 and country == 'China' and death == '1':

        ch\_count += 1

*elif* (age > 50) and (country == 'Germany') and (death == '1'):

        de\_count += 1

oran = ch\_count / de\_count

print(ch\_count)

print(de\_count)

print(oran)

**SORU #2**

*# 2. Bir dizinin içindeki en buyuk tek ve en küçük tek sayı arasındaki farkın karesini bulan  python  kodunu fonksiyon kullanarak yazın.*

*import* random

def hesap(*liste*):

*# Yontem 1*

*# liste = sorted(liste)*

*# return liste*

*# Yontem 2*

    ek = liste[0]

*for* i *in* range(0, len(liste)):

*if* liste[i] < ek:

            ek = liste[i]

*return* ek

*# Yontem 3*

*# return min(liste)*

liste = [random.randrange(1, 50, 1) *for* i *in* range(50)]

print(liste)

print(hesap(liste))

**SORU #3**

Verilen L = [1, 2, 3, 4, 5, 6,7,8,……….225] listesine, lambda, map, reduce, filter fonk kullanarak,

*from* functools *import* reduce

*from* math *import* sqrt

*from* statistics *import* variance, mean

liste = [i *for* i *in* range(1, 256)]

x = []

*# a)sayılardan 3’e tam bölünenleri*

uc = lambda *sayi*: sayi % 3

*for* i *in* liste:

*if* uc(i) == 0:

        print(i)

*# b)çift olan sayuların toplamını*

sonuc = list(filter(lambda *x*: x % 2 == 0, liste))

toplam = 0

*for* i *in* range(0, len(sonuc)):

    toplam = toplam + sonuc[i]

print(toplam)

*# c)ortalamının %10  üstüne olanları*

print(list(filter(lambda *sayi*: (sayi > (mean(liste) + mean(liste) \* 0.1)), liste)))

*# d) listede sayilarin kupleirni ve toplamini*

kup = list(map(lambda *x*: x \*\* 3, liste))

print(kup)

toplam = reduce(lambda *x*, *y*: x + y, liste)

print(toplam)

*# e)Sayıların varyans ve standard sapmasını bulan programı yazınız*

varyans = lambda *liste*, *ort*: sum([x \*\* 2 *for* x *in* [i - ort *for* i *in* liste]]) / float(

    len(liste)

)

print(varyans(liste, mean(liste)))

std\_sapma = lambda *liste*: sqrt(variance(liste, mean(liste)))

std\_sapma2 = lambda *liste*: sqrt(varyans(liste, mean(liste)))

print(std\_sapma(liste))

print(std\_sapma2(liste))

**SORU #4**

*# OpenCv kullanarak bir görüntü okuyun, görüntüyü siyah beyaz cevirin gösterin, filtre kullanarak: gaussian blur , canny kenar bulma, keskinleştirme filtresi(   ), sobel filtresi(       ), ortalama blur filtresi(  ) , Laplacian filtresi , dilation işlemi, erosion işlemi, olacak şekilde 8 işlemi, subplot kullanarak çizdirin. Görüntünün histogramını çıkarıp çizdirin, threshold  belirleyip, objeleri siyah-beyaz gösterin(basit segmentasyon) yapın*

*import* cv2 *as* cv

*import* numpy *as* np

*import* matplotlib.pyplot *as* plt

img = cv.imread('animals.jpg')

rgb\_img = cv.cvtColor(img, cv.COLOR\_BGR2RGB)

*# cv.imshow('Original', img)*

plt.subplot(1, 1, 1), plt.imshow(rgb\_img, 'gray')

plt.title('Original'), plt.xticks([]), plt.yticks([])

plt.show()

*# Siyah Beyaz*

gray = cv.cvtColor(rgb\_img, cv.COLOR\_BGR2GRAY)

*# cv.imshow('Black and White', gray)*

plt.subplot(1, 1, 1), plt.imshow(gray, 'gray')

plt.title('Black and White'), plt.xticks([]), plt.yticks([])

plt.show()

*# GaussianBlur*

blur = cv.GaussianBlur(rgb\_img, (5, 5), cv.BORDER\_DEFAULT)

*# cv.imshow('Blur', blur)*

plt.subplot(1, 1, 1), plt.imshow(blur, 'gray')

plt.title('GaussianBlur'), plt.xticks([]), plt.yticks([])

plt.show()

*# Canny Edges*

canny = cv.Canny(rgb\_img, 125, 175)

*# cv.imshow('Canny Edges', canny)*

plt.subplot(1, 1, 1), plt.imshow(canny, 'gray')

plt.title('Canny Edges'), plt.xticks([]), plt.yticks([])

plt.show()

*# Sharpening*

sharp\_filter = np.array(

    [

        [0, -1, 0],

        [-1, 5, -1],

        [0, -1, 0],

    ]

)

sharp\_img = cv.filter2D(rgb\_img, -1, sharp\_filter)

plt.subplot(1, 1, 1), plt.imshow(sharp\_img, 'gray')

plt.title('Sharpened'), plt.xticks([]), plt.yticks([])

plt.show()

*# cv.imshow('Sharpened', sharp\_img)*

*# cv.waitKey(0)*

*# cv.destroyAllWindows()*

*# # Sobel*

sobelx = cv.Sobel(rgb\_img, cv.CV\_64F, 1, 0)

sobely = cv.Sobel(rgb\_img, cv.CV\_64F, 0, 1)

plt.subplot(2, 2, 1), plt.imshow(rgb\_img, 'gray')

plt.title('Original'), plt.xticks([]), plt.yticks([])

plt.subplot(2, 2, 3), plt.imshow(sobelx, 'gray')

plt.title('Sobel X'), plt.xticks([]), plt.yticks([])

*# cv.imshow('Sobel X', sobelx)*

plt.subplot(2, 2, 4), plt.imshow(sobely, 'gray')

plt.title('Sobel Y'), plt.xticks([]), plt.yticks([])

*# cv.imshow('Sobel Y', sobely)*

plt.show()

*# Average Blur*

kernel = np.ones((5, 5), np.float32) / 9

dst = cv.filter2D(rgb\_img, -1, kernel)

plt.subplot(1, 2, 1), plt.imshow(rgb\_img, 'gray'), plt.title('Original')

plt.xticks([]), plt.yticks([])

plt.subplot(1, 2, 2), plt.imshow(dst), plt.title('Averaging')

plt.xticks([]), plt.yticks([])

*# cv.imshow('Median Blur', dst)*

plt.show()

*# Laplacion*

laplacion = cv.Laplacian(rgb\_img, cv.CV\_64F)

plt.subplot(1, 1, 1), plt.imshow(laplacion, 'gray'), plt.title('Laplacion')

plt.xticks([]), plt.yticks([])

*# cv.imshow('Laplacion', laplacion)*

plt.show()

*# Dilation*

dilated = cv.dilate(canny, (7, 7), *iterations*=1)

plt.subplot(1, 1, 1), plt.imshow(dilated, 'gray'), plt.title('Dilation')

plt.xticks([]), plt.yticks([])

*# cv.imshow('Dilation', dilated)*

plt.show()

*# Eroding*

eroded = cv.erode(dilated, (3, 3), *iterations*=1)

plt.subplot(1, 1, 1), plt.imshow(eroded, 'gray'), plt.title('Eroding')

plt.xticks([]), plt.yticks([])

*# cv.imshow('Eroding', eroded)*

plt.show()

*# Histogram*

color = ('b', 'g', 'r')

*for* i, col *in* enumerate(color):

    histr = cv.calcHist([rgb\_img], [i], None, [256], [0, 256])

    plt.plot(histr, *color*=col)

    plt.xlim([0, 256])

plt.hist(rgb\_img.ravel(), 256, [0, 256])

plt.show()

*# Threshold*

threshold, thresh = cv.threshold(gray, 150, 255, cv.THRESH\_BINARY)

plt.subplot(1, 1, 1), plt.imshow(thresh, 'gray'), plt.title('Threshold')

plt.xticks([]), plt.yticks([])

*# cv.imshow('Threshold', thresh)*

plt.show()

threshold, thresh\_inv = cv.threshold(gray, 150, 255, cv.THRESH\_BINARY\_INV)

plt.subplot(1, 1, 1), plt.imshow(thresh\_inv, 'gray'), plt.title('Threshold Inverted')

plt.xticks([]), plt.yticks([])

*# cv.imshow('Threshold Inverted', thresh\_inv)*

plt.show()

*# Adaptive Threshold*

adaptive\_threshold = cv.adaptiveThreshold(

    gray, 255, cv.ADAPTIVE\_THRESH\_MEAN\_C, cv.THRESH\_BINARY, 11, 3

)

plt.subplot(1, 1, 1), plt.imshow(adaptive\_threshold, 'gray'), plt.title(

    'Adaptive Threshold'

)

plt.xticks([]), plt.yticks([])

*# cv.imshow('Adaptive Threshold', adaptive\_threshold)*

plt.show()

**SORU #5**

*# 5. Matplotlib kutuphanesi kullanarak: y=x2 , y=x, y = mutlak(x), y=sin(x) 2d grafiklerini çizdiren, meshgrid kullanarak z=x2+y2 grafiğini çizdiren kodu yazın.*

*import* cv2 *as* cv

*import* numpy *as* np

*import* matplotlib.pyplot *as* plt

*# Creating vectors X and Y*

x = np.linspace(-5, 5, 255)

y = x \*\* 2

y2 = x

y3 = abs(x)

y4 = np.sin(x)

fig = plt.figure(*figsize*=(10, 8))

*# Create the plot*

plt.subplot(2, 2, 1), plt.plot(x, y), plt.title('y=x\*\*2')

plt.subplot(2, 2, 2), plt.plot(x, y2), plt.title('y=x')

plt.subplot(2, 2, 3), plt.plot(x, y3), plt.title('y=|x|')

plt.subplot(2, 2, 4), plt.plot(x, y4), plt.title('y=sin(x)')

*# z = x2 + y2*

x0 = np.linspace(-10, 10, 25)

y0 = np.linspace(-10, 10, 25)

x, y = np.meshgrid(x0, y0)

z = (x \*\* 2) + (y \*\* 2)

cmap = plt.get\_cmap('viridis')

fig = plt.figure()

ax = fig.add\_subplot(111, *projection*='3d')

ax.plot\_surface(x, y, z, *cmap*=cmap)

ax.set\_title('z = x^2 + y^2')

plt.show()