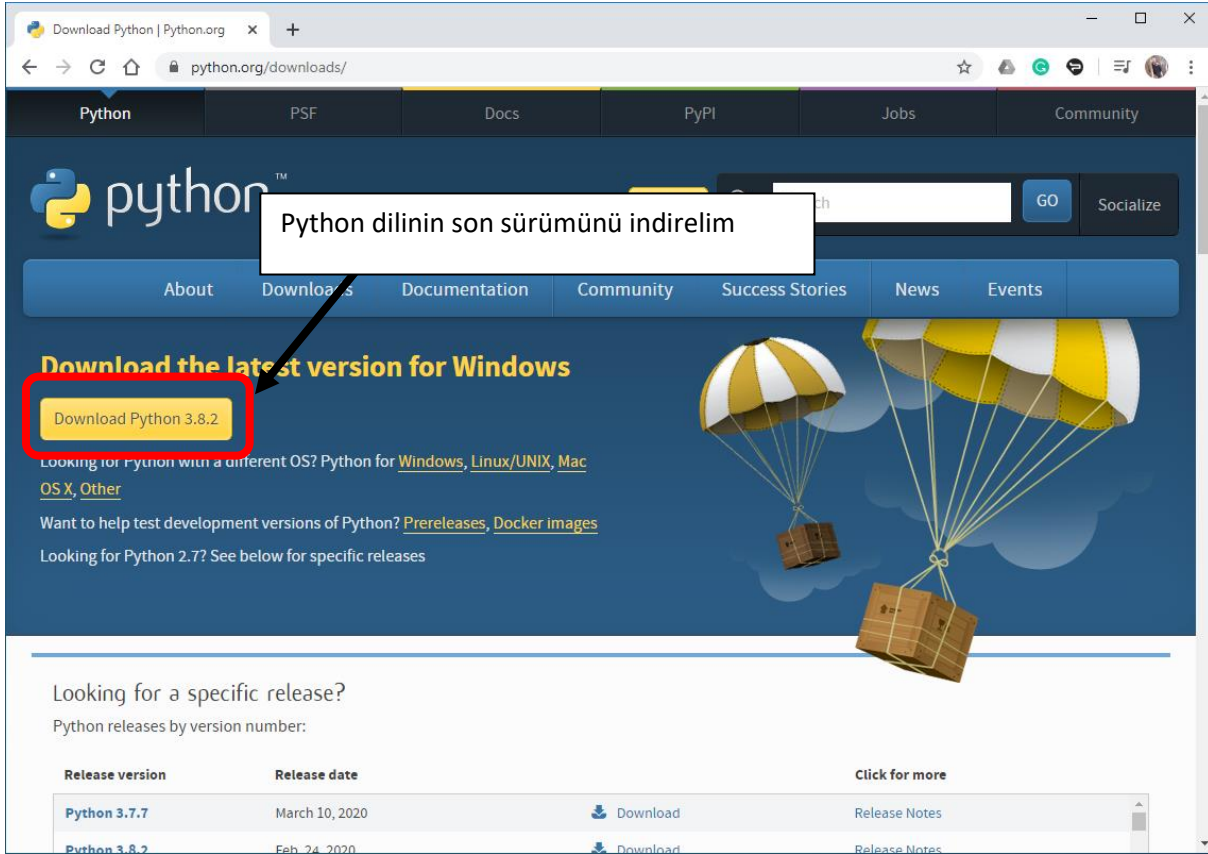


# BİLGİSAYAR GÖRMESİ UYGULAMA DERSİ 1

## Uygulamalar

<https://github.com/kayhanayar/bilgisayargormesi>

## PYTHON KURULUMU



The screenshot shows the Python.org website's download page. A callout box with the text "Python dilinin son sürümünü indirelim" (Let's download the latest version of the Python language) points to the "Download Python 3.8.2" button, which is highlighted with a red rectangle. The page features the Python logo, navigation links, and a section titled "Download the latest version for Windows". Below this, there are links for other operating systems and a table of Python releases.

Python dilinin son sürümünü indirelim

Download Python 3.8.2

Looking for Python with a different OS? Python for [Windows](#), [Linux/UNIX](#), [Mac OS X](#), [Other](#)

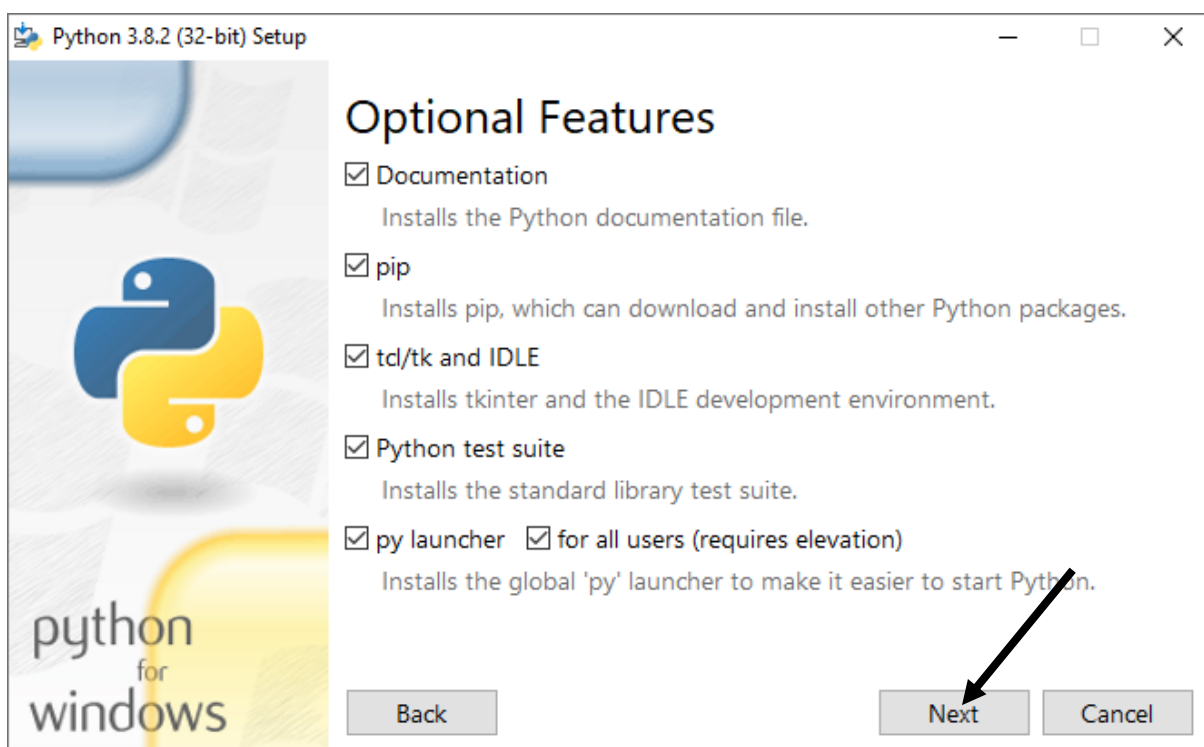
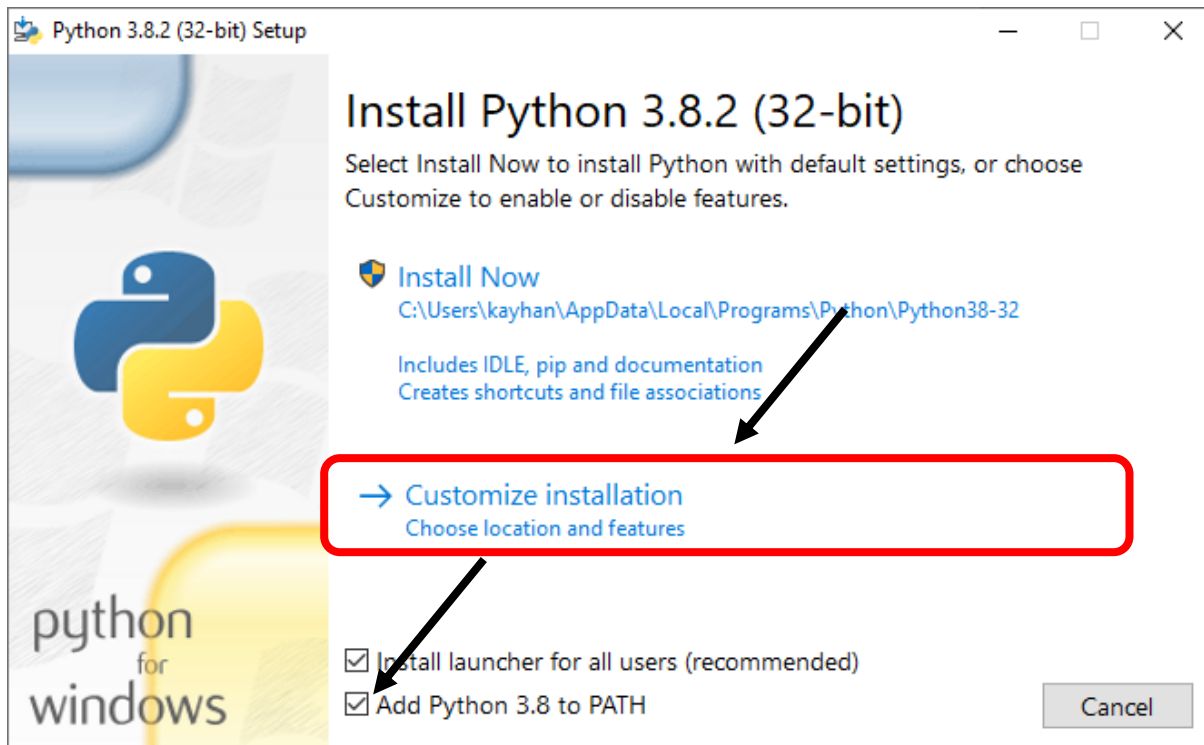
Want to help test development versions of Python? [Prereleases](#), [Docker images](#)

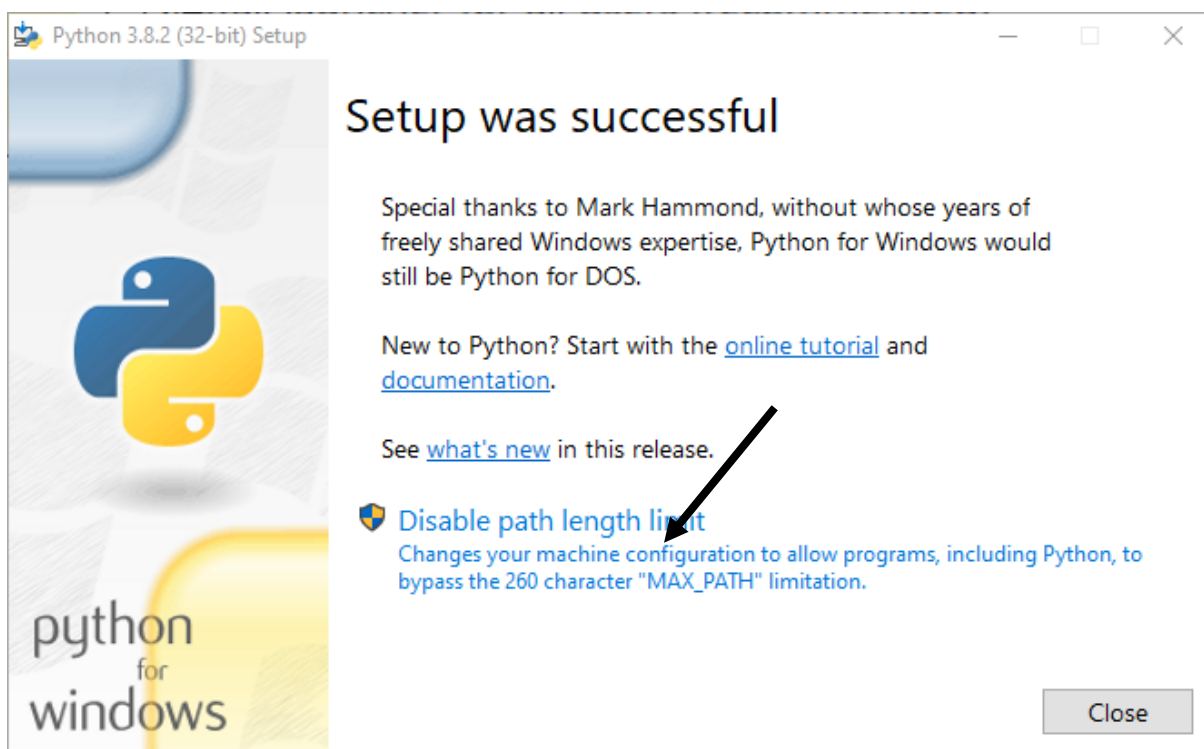
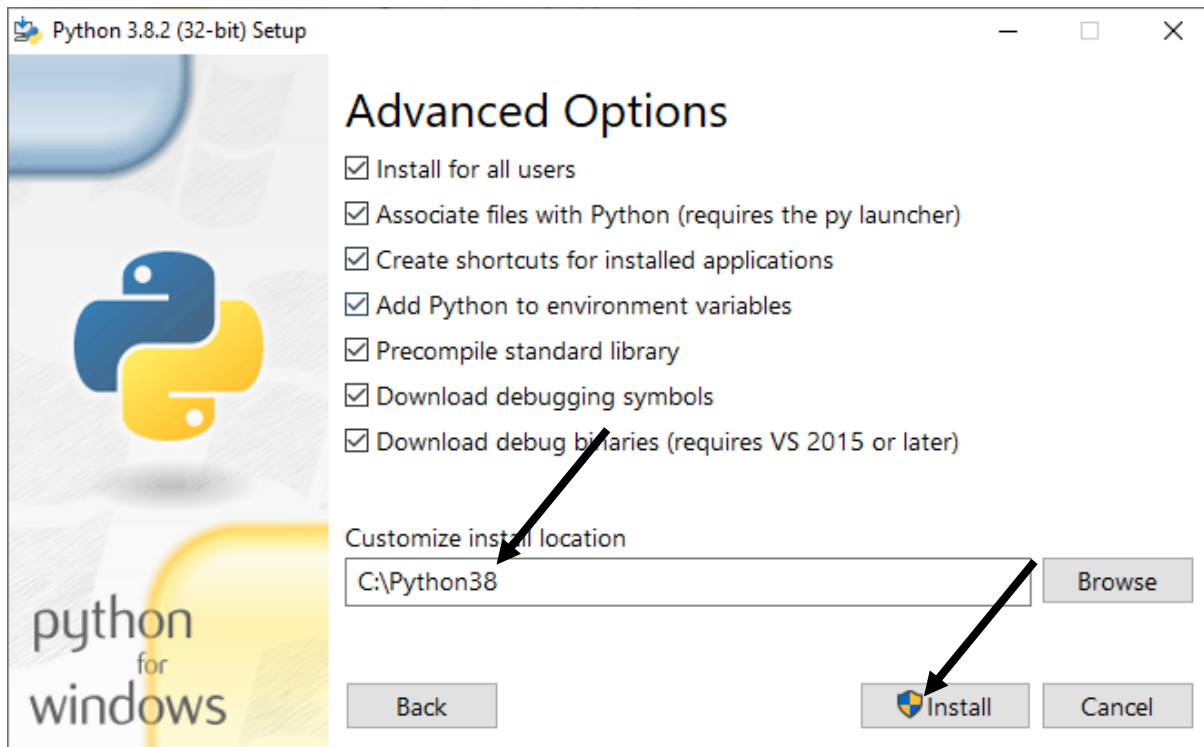
Looking for Python 2.7? See below for specific releases

Looking for a specific release?

Python releases by version number:

Release version	Release date		Click for more
<a href="#">Python 3.7.7</a>	March 10, 2020	<a href="#">Download</a>	<a href="#">Release Notes</a>
<a href="#">Python 3.8.2</a>	Feb. 24, 2020	<a href="#">Download</a>	<a href="#">Release Notes</a>

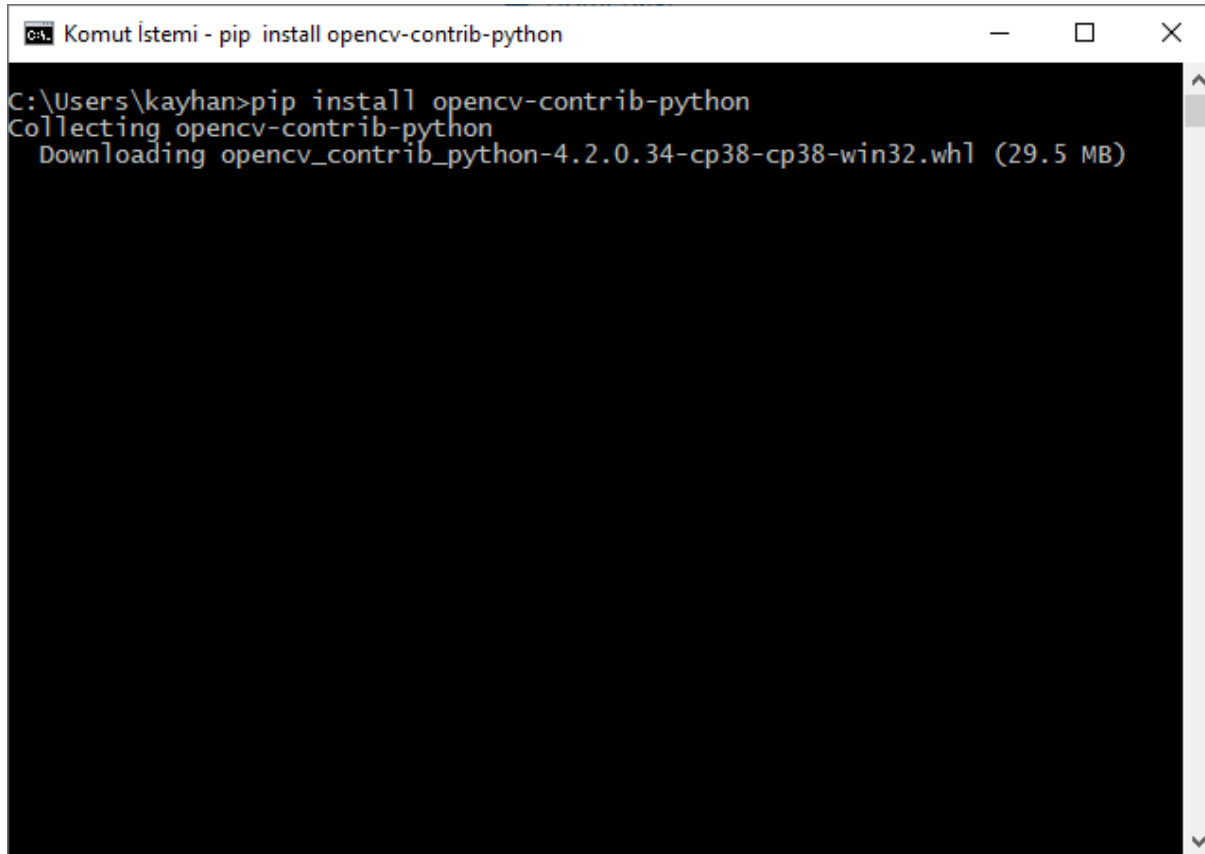




## OPENCV KURULUMU

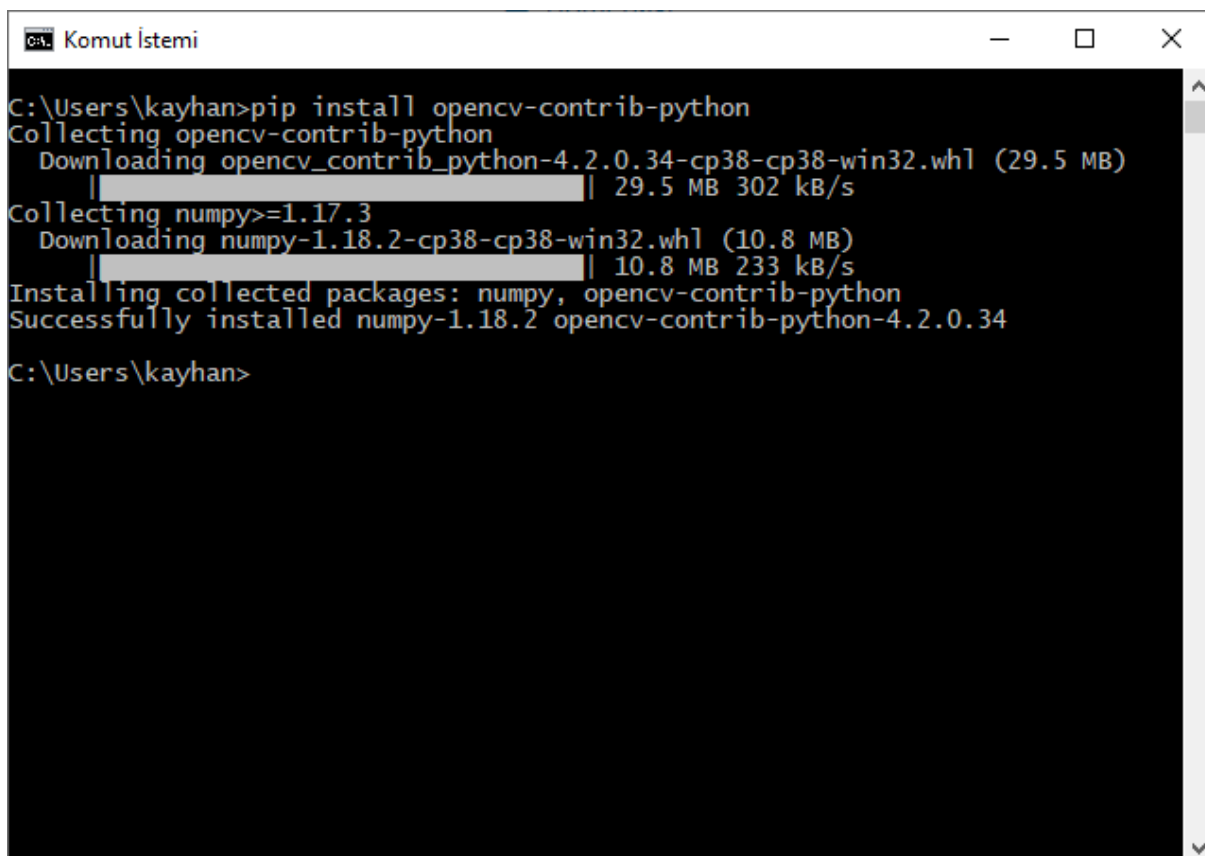
**Komut Satırına girilip aşağıdaki satır yazılır.**

**pip install opencv-contrib-python**



```
Komut İstemi - pip install opencv-contrib-python

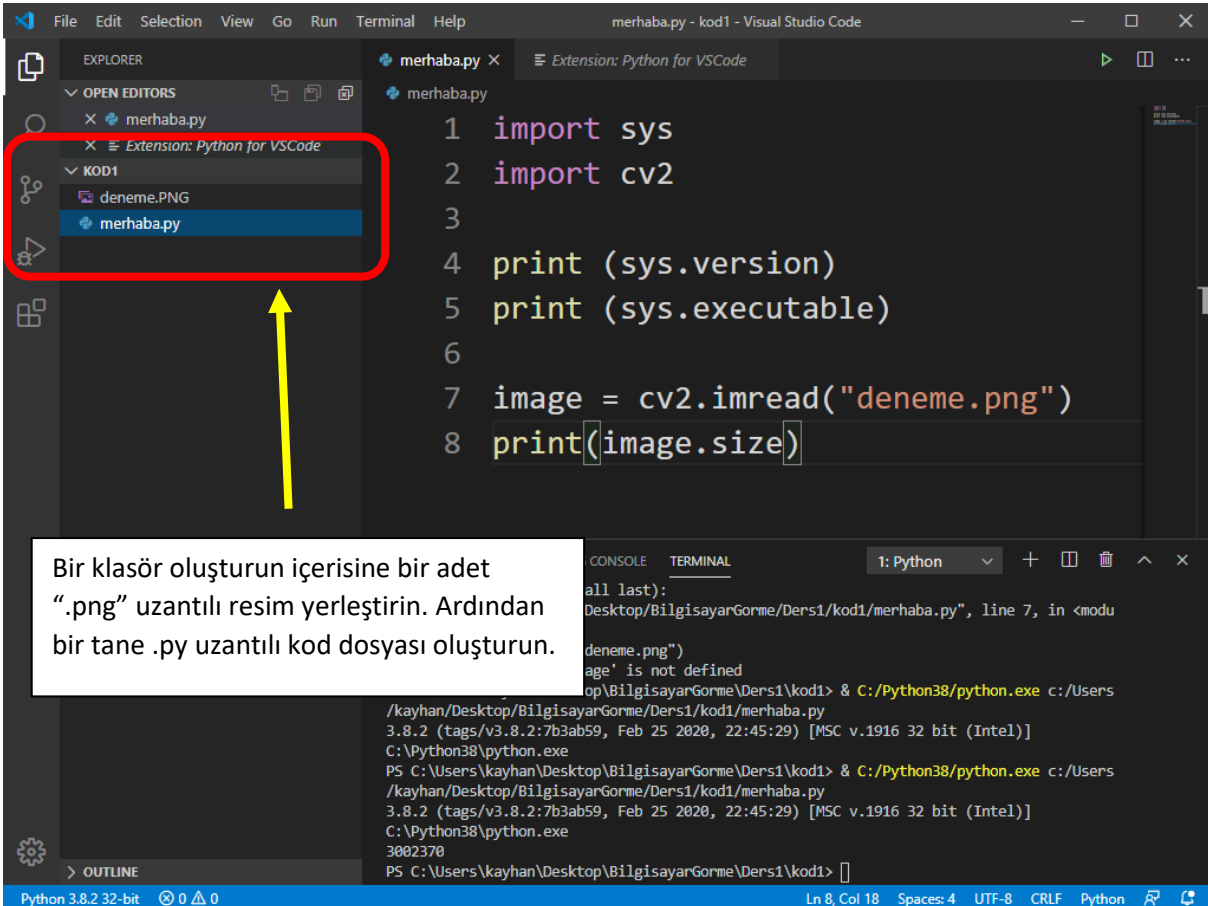
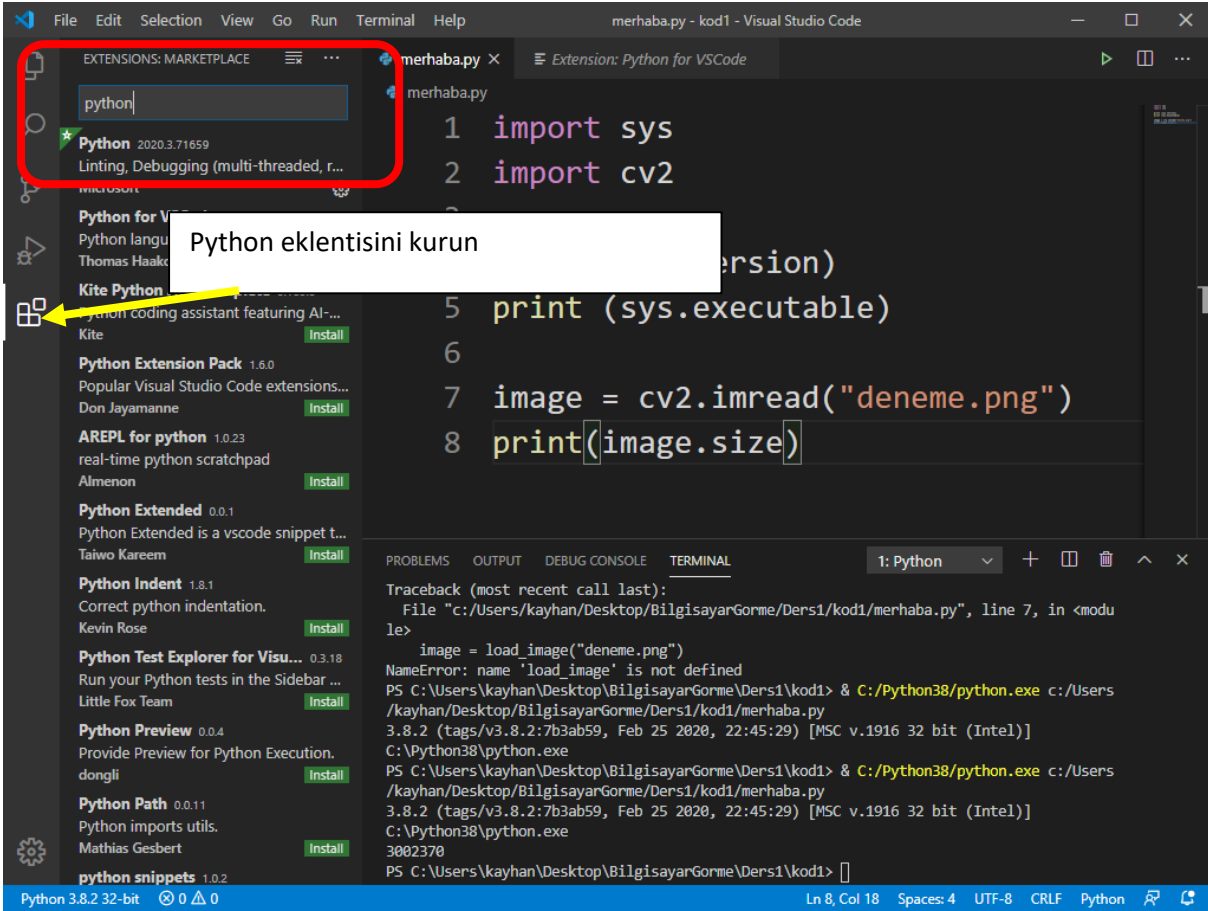
C:\Users\kayhan>pip install opencv-contrib-python
Collecting opencv-contrib-python
  Downloading opencv_contrib_python-4.2.0.34-cp38-cp38-win32.whl (29.5 MB)
```



```
Komut İstemi

C:\Users\kayhan>pip install opencv-contrib-python
Collecting opencv-contrib-python
  Downloading opencv_contrib_python-4.2.0.34-cp38-cp38-win32.whl (29.5 MB)
|-----| 29.5 MB 302 kB/s
Collecting numpy>=1.17.3
  Downloading numpy-1.18.2-cp38-cp38-win32.whl (10.8 MB)
|-----| 10.8 MB 233 kB/s
Installing collected packages: numpy, opencv-contrib-python
Successfully installed numpy-1.18.2 opencv-contrib-python-4.2.0.34
C:\Users\kayhan>
```

## Visual Studio Code (IDE) Kullanımı



## merhaba.py

```
import sys
```

Yardımcı kütüphane ekleniyor

```
import cv2
```

OpenCV ekleniyor

```
print (sys.version)
```

Python sürümü ekrana çıkartılıyor

```
print (sys.executable)
```

Python programının konumu  
Ekranı çıkartılıyor

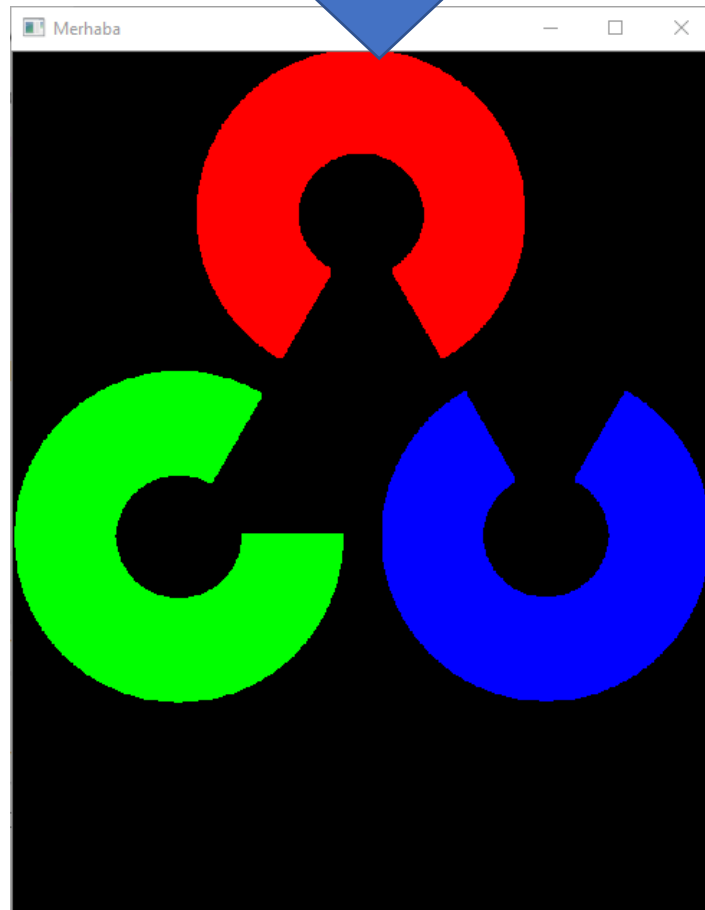
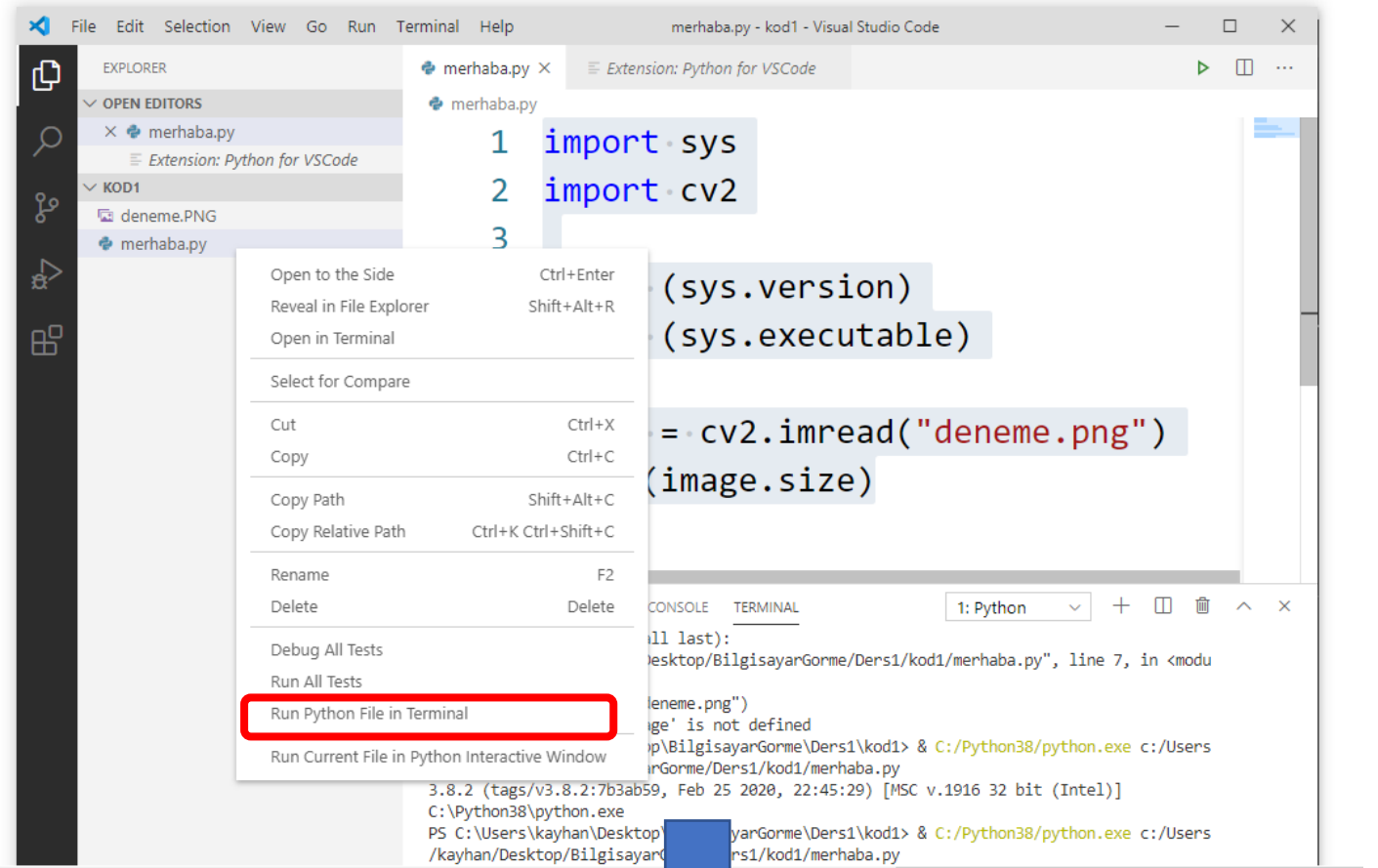
```
image = cv2.imread("deneme.png")
```

```
print(image.size)
```

Resmin boyutu ekrana çıkartılıyor

Opencv kullanılarak bir resim okunuyor ve  
image nesnesine yerleştiriliyor

Fareyi kod dosyasının üzerine getirip sağ tuşa tıkladığınızda aşağıdaki menü karşınıza çıkacaktır. Buradan seçili kısmı tıklayıp programınızı çalıştırabilirsiniz.



## RESİMİN BELİRLİ BİR BÖLGESİNİ ALMA(Kod2)

```
import sys
import cv2

resim = cv2.imread("deneme.png")

#resimdeki piksel bilgilerine bir dizi elemanına eriştiğimiz
#gibi erişebilmekteyiz. aşağıda 40 sütun 6. satır pikselinin
#renk değeri getirilmektedir. PNG formatında piksel renk
#değerleri BGR (blue,green,red) şeklinde saklandığı için
#diziden alınan elemanın da 3 farklı değişkene yerleştirilmesi gerekmektedir.
#python üç değer döndürmeye imkan tanımaktadır.

b,g,r = resim[6,40]

print(b,g,r)

#aynı şekilde bir piksele yeni değer atamak istediğimizde renk
#değerlerini ayrı ayrı vermeliyiz

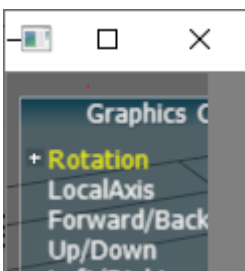
resim[6,40]=0,0,255

#bazen resimden bir piksel yerine bir bölge almak isteyebiliriz
#bunun içinde aşağıdaki gibi istediğimiz bölgeyi belirtmemiz gerekir.

sol_ust_kose = resim[0:100,0:100]

cv2.imshow("Merhaba",sol_ust_kose)

cv2.waitKey(0);
```





## Gri Tonlamalı Resim (Kod3)

```
import sys
import cv2

#Resimi siyah-beyaz olarak yükleme (GRayscale)
#bu formatta tek bir renk ve tonları bulunmaktadır.
resim = cv2.imread("deneme.png",cv2.IMREAD_GRAYSCALE)

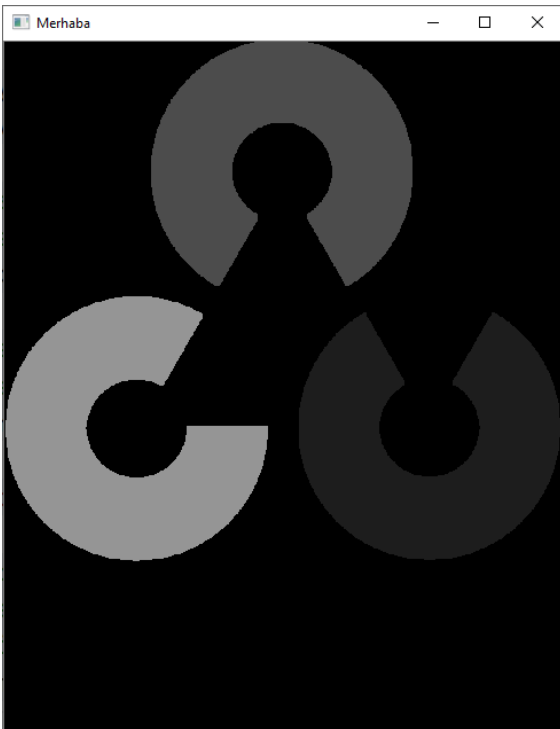
#boyut sadece genişlik ve yükseklik bilgilerini tutacaktır.
#siyah beyaz resimlerde kanal bilgisi bulunmaz.
boyut = resim.shape;

print("boyut:"+str(boyut))

#piksel bilgisi sadece beyaz renginin tonlarını tutmaktadır.
# o yüzden aşağıdaki işlem sonucun da tek bir değer dönecektir
#o değeri de i değişkeni içerisinde tuttuk
i = resim[6,40];

print("beyaz yoğunluğu:"+str(i));

cv2.imshow("Merhaba",resim)
cv2.waitKey(0);
```



## Uygulama Argümanlar Kontrolü (Kod4)

```
import sys
import cv2
#sys.argv dizisi içerisinde komut satırından gelen parametreler
i barındırır.
#ilk parametre her programlama dilinde olduğu gibi çalıştırılan
uygulamanın tam yoludur
print(sys.argv[0]);

#istersek ekrana biçimli çıktı alabiliriz. '{}' içerisine
# format fonksiyonun ilk parametresi yerleştirilir
#eğer birden fazla '{}' olsaydı format fonksiyonun da buna
#göre parametre sayısının artması gerekirdi.

print("calistirilan program:'{}'.format(sys.argv[0]))

#dışarıdan girilen parametre sayısı len(sys.argv) ile elde edil
ebilir

print("parametre sayısı:'{}'.format(len(sys.argv)))

#dizinin hepsinin ekrana dökümü

print("parametreler:'{}'.format(str(sys.argv)))

cv2.waitKey(0);
```

Programımız çalıştırılırken komut satırında ayrıca argümanlarda verilebilmektedir. Aşağıdaki kod aracılığıyla bu argümanları yönetebiliriz.

Programımızı çalıştırmak için komut satırında aşağıdaki bir komut yazılabilir.

**Py program.py ilkArguman ikinciArguman**

```
PS C:\Users\kayhan\Desktop\BilgisayarGorme\Ders1\kod4> py program.py ilkArguman ikinciArguman
program.py
calistirilan program:'program.py'
parametre sayısı:'3'
parametreler:['program.py', 'ilkArguman', 'ikinciArguman']
PS C:\Users\kayhan\Desktop\BilgisayarGorme\Ders1\kod4> █
```

## Uygulama Argümanlar Kontrolünü Otomatikleştirme (Kod5)

```
# Bu uygulamada argparse kütüphanesinin kullanımını gösteriyoruz
# programa gelen argümanları daha iyi kontrol etmek için kullanılıyor.

import argparse

#parser nesnesi oluşturuluyor
parser = argparse.ArgumentParser()

#parser nesnesine kullanıcıdan alınacak bir argüman tanıtılıyor
#ilk parametre argümanın alacağı değişken ismi
#ikinci parametre argümanın yanında çıkacak olan yardım mesajı

parser.add_argument("ilk_arguman",help="ilk arguman yardım");

#argümanlar yakalanıyor
args = parser.parse_args()

#ilk argümanın değeri ekrana çıkartılıyor
print(args.ilk_arguman)
```

Komut satırında aşağıdaki satır yazılırsa.

**Py program.py ilkArguman**

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\kayhan\Desktop\BilgisayarGorme\Ders1\kod5> py .\program.py
usage: program.py [-h] ilk_arguman
program.py: error: the following arguments are required: ilk_arguman
PS C:\Users\kayhan\Desktop\BilgisayarGorme\Ders1\kod5> py .\program.py ilkarguman
ilkarguman
PS C:\Users\kayhan\Desktop\BilgisayarGorme\Ders1\kod5> █
```

## Uygulama Argümanlar Kontrolünü Otomatikleştirme (Kod6)

```
import argparse

parser = argparse.ArgumentParser()

parser.add_argument("ilk_arguman",help="ilk arguman yardım");

#ilk parametremiz otomatik olarak string türünde değer alacaktır.
#ikinci parametrenin tamsayı alabilmesi için fonksiyona yeni bir parametre ekliyoruz
#bu parametre içerisinde type değerini kullanıyoruz
parser.add_argument("ikinci_arguman",help="ikinci arguman yardım",type=int);

parser.add_argument("ucuncu_arguman",help="ucuncu arguman yardım",type=int);

args = parser.parse_args()

print(args.ilk_arguman)
print(args.ikinci_arguman)
print(args.ucuncu_arguman)
```

Komut satırında aşağıdaki satır yazılırsa.

**Py program.py ilkArguman 123 456**

```
PS C:\Users\kayhan\Desktop\BilgisayarGorme\Ders1\kod6> py program.py --help
usage: program.py [-h] ilk_arguman ikinci_arguman ucuncu_arguman

positional arguments:
  ilk_arguman      ilk arguman yardım
  ikinci_arguman   ikinci arguman yardım
  ucuncu_arguman   ucuncu arguman yardım

optional arguments:
  -h, --help      show this help message and exit
PS C:\Users\kayhan\Desktop\BilgisayarGorme\Ders1\kod6> Py program.py ilkArguman 123 456
ilkArguman
123
456
PS C:\Users\kayhan\Desktop\BilgisayarGorme\Ders1\kod6> █
```

## Resim kanalları ayrıştırma (Kod7)

```
import argparse
import sys
import cv2

image = cv2.imread("deneme.png")
b,g,r = cv2.split(image)

new_image = cv2.merge((g,r,b))

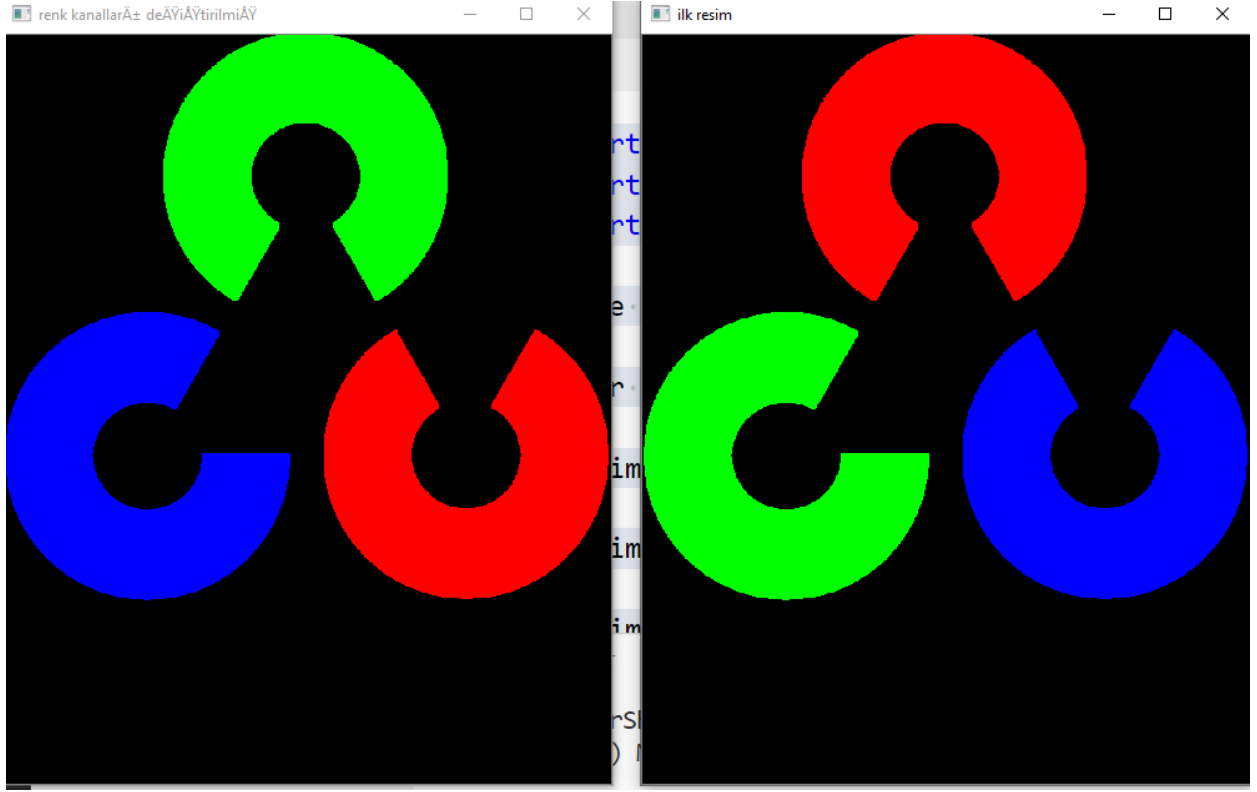
cv2.imshow("ilk resim",image)
cv2.imshow("renk kanalları değiştirilmiş",new_image)
cv2.waitKey(0)
cv2.destroyAllWindows()

#split fonksiyonu oldukça çok işlem yapmaktadır
#onun yerine Numpy kütüphanesinin sağladığı bir aracı kullanacağız
#aşağıda image dizisinin 0. kanalını yani mavi kanalını ayıracağız.
#şuan b sadece bir kanal temsil ediyor.
b= image[:, :,0]
#eğer bu kanal ekrana çıkartmak istersek siyah beyaz çıkar ve sadece mavi
#rengin olduğu yerler beyaz olacaktır.
cv2.imshow("sadece maviler",b)
cv2.waitKey(0)
cv2.destroyAllWindows()
#istenirse bir kanal komple sıfır yapılabilir
#bu seyede resimden ilgili kanal komple çıkartılmış olur
mavisizResim = image.copy()

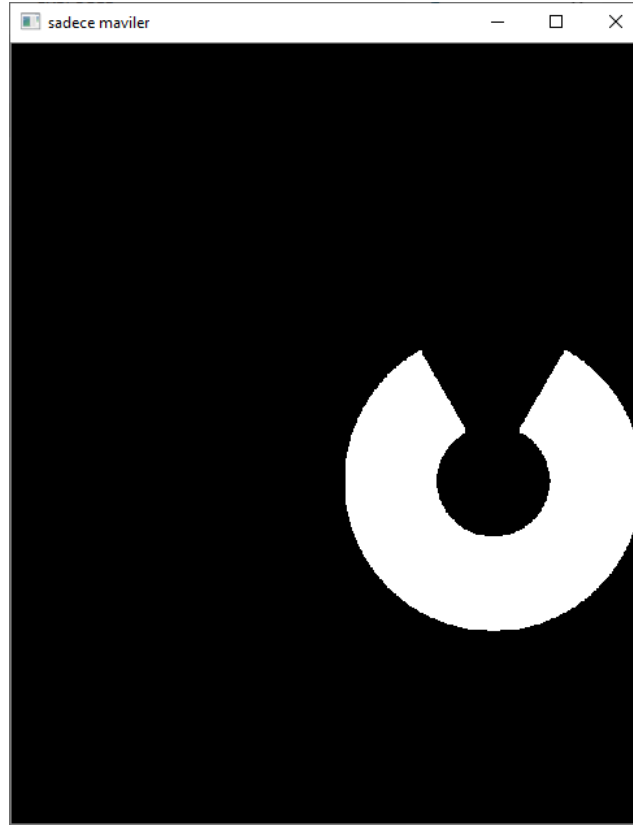
mavisizResim[:, :,0]=0

cv2.imshow("sadece maviler",mavisizResim)
cv2.waitKey(0)
```

## Resim kanalları ayırıştırma Ekran Çıktısı (Kod7)



İlk pencereler kapatıldıktan sonra ekrana aşağıdaki pencere çıkar



## Resim ölçekleme (Kod8)

```
import sys
import cv2

image = cv2.imread("deneme.png")

height,width,channels = image.shape
##resim büyötmek için kübik veya lineer enterpolasyon (ara değeri bulma)
## tekniğı kullanmak daha iyi sonuçlar verecektir.
## Kübik enterpolasyon daha fazla işlem gücü gerektirmektedir
buyukResim = cv2.resize(image,(width*2,height*2),interpolation=cv2.INTER_LINEAR)

cv2.imshow("ölçekli",buyukResim)
cv2.waitKey(0)
cv2.destroyAllWindows()

##küçöltme yaparken alan enterpolasyon kullanılmaktadır.
#piksel değeri bulundukları alana göre yeni değeri almadır.
kucukResim = cv2.resize(image,None,fx=0.5,fy=0.5,interpolation=cv2.INTER_AREA)

cv2.imshow("ölçekli",kucukResim)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

## Resim Öteleme (Kod9)

Resme ait pikselleri ötelemek için her pikselin resim içerisindeki koordinatlarını ötelememiz gerekir. Öteleme işlemi için x ve y koordinatları öteleme matrisi ile çarpılarak yeni koordinatlar elde edilebilmektedir. Matrisin formatı aşağıda verilmiştir. Burada tx ve ty öteleme miktarlarını temsil etmektedir.

$$M = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix}$$

Öteleme matrisi oluşturmak için NumPy kütüphanesini kullanacağız. Kütüphaneyi aşağıdaki gibi programımıza ekleyeceğiz. Kütüphane ismini program içerisinde de np olarak adlandırıyoruz.

```
import numpy as np
```

öteleme matrisi de aşağıdaki gibi oluşturulmaktadır.

```
M = np.float32([[1,0,200],[0,1,150]])
```

Matris iki satırdan oluşmaktadır her satırda da üç sütun bulunmaktadır.

Resme ait pikselleri ötelemek için de aşağıdaki satırı kullanıyoruz. İşlem sonucunda yeni bir resim oluşmaktadır.

```
yeniResim = cv2.warpAffine(image,M,(cols,rows))
```



## Resim Öteleme (Kod9)

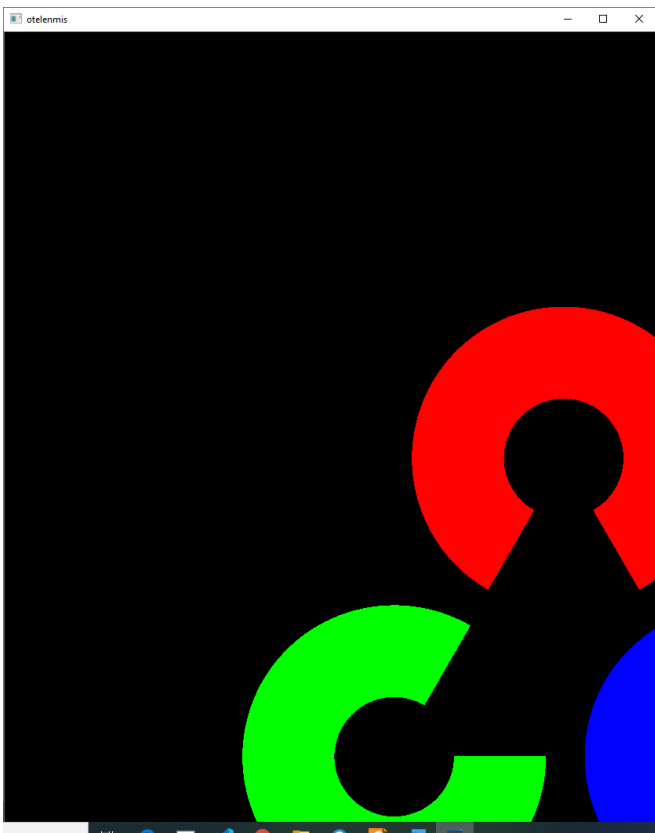
```
import sys
import cv2
import numpy as np
image = cv2.imread("deneme.png")

rows,cols,channels = image.shape

M = np.float32([[1,0,300],[0,1,350]])

yeniResim = cv2.warpAffine(image,M,(cols,rows))
cv2.imshow("otelenmis",yeniResim)

cv2.waitKey(0)
cv2.destroyAllWindows()
```



## Resim Döndürme (Kod10)

Döndürme matrisinin formu aşağıdaki gibi olacaktır.

$$M = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Eğer bütün dönüşüm matrislerini birleştirirsek(öteleme,döndürme,ölçekleme) aşağıdaki matris formunu elde ederiz.

$$\begin{bmatrix} \alpha & \beta & (1 - \alpha) \cdot center.x - \beta \cdot center.y \\ -\beta & \alpha & \beta \cdot center.x + (1 - \alpha) \cdot center.y \end{bmatrix}$$

Burada alfa ve beta değerleri aşağıdaki gibidir.

$$\alpha = scale \cdot \cos \theta,$$

$$\beta = scale \cdot \sin \theta$$

Bir resmi merkeze göre 90 derece döndürecek python kodu aşağıdaki gibi olacaktır.

```
image = cv2.imread("deneme.png")
```

```
height,width,channels = image.shape
```

Aşağıdaki fonksiyon döndürme işlemi yapmaktadır. İlk parametre resmin yükseklik ve genişliği bütün olarak verilmektedir. İkinci parametre döndürme açısı son parametre ölçek değeridir.

```
M = cv2.getRotationMatrix2D((height/2,width/2),90,1)
```

```
yeniResim = cv2.warpAffine(image,M,(width,height))
```

## Piksel operasyonları (Kod11)

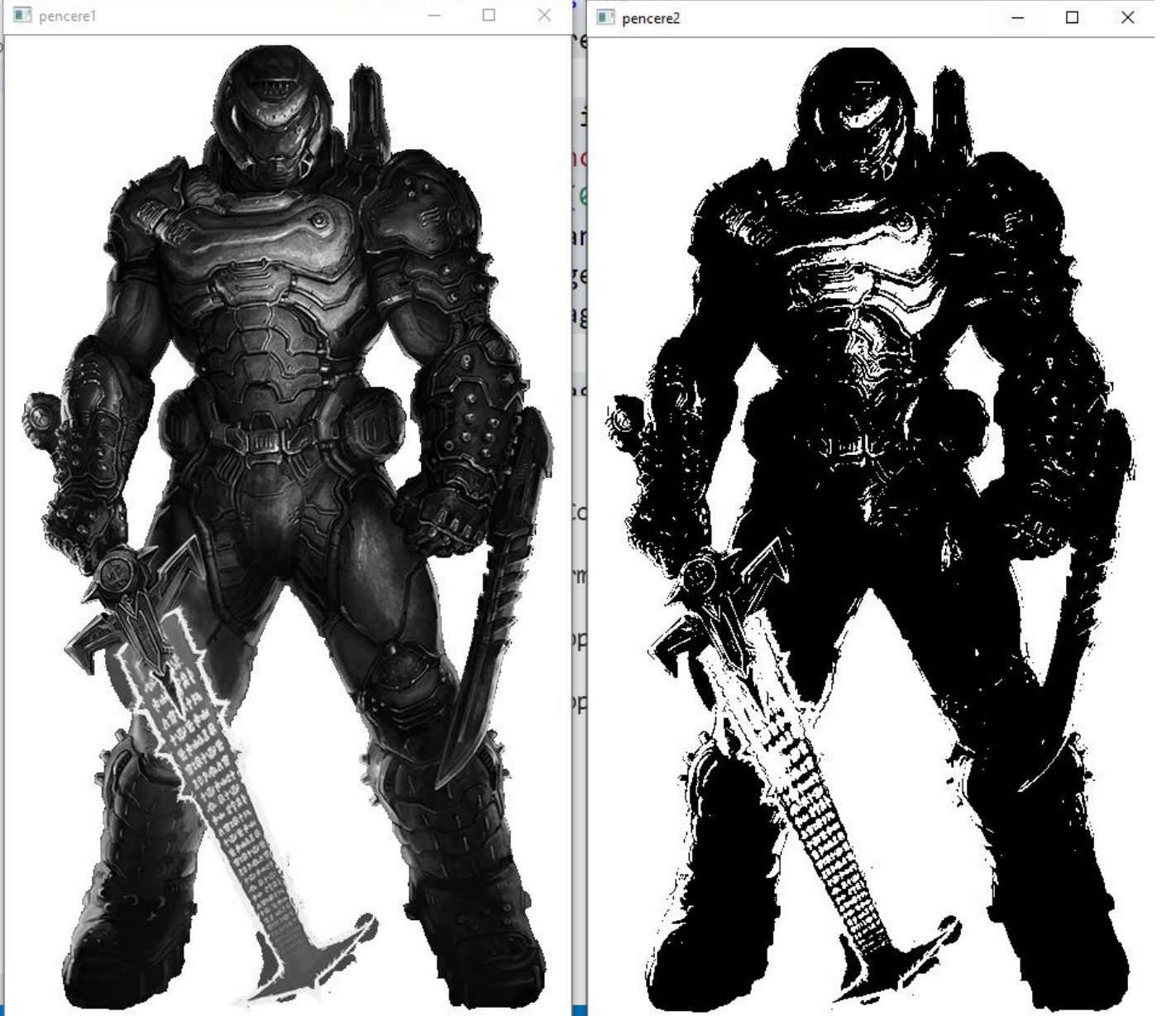
Bu uygulamada resme ait pikseller üzerinde işlem yapacağız. Öncelikle resim gri tonlamalı olacak şekilde açılıyor. Ardından bütün pikselleri dolaşacak bir döngü gerçekleştiriliyor. Bu döngü içerisinde her bir pikselin gri tonu kontrol ediliyor. Eğer ton 100 değerinden büyükse piksele 255 gri tonu atanıyor. Aksi durumda direk 0 değeri atanıyor.

```
import sys
import cv2
import numpy as np
image = cv2.imread("doomslayer.png",0)

height,width = image.shape
cv2.imshow("pencere1",image)
for y in range(0,height):
    for x in range(0,width):
        if image[y, x] >= 100:
            image[y, x] = 255
        else:
            image[y, x] = 0

cv2.imshow("pencere2",image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

## Piksel operasyonları sonuç (Kod11)



## Lineer Filtre sonuç (Kod12)

Filtreleme tekniklerinde genelde çekirdek adı verilen 3x3 veya 5x5 lik bir matris her bir piksel üzerine uygulanmaktadır. İşlem yapılacak olan pikselin etrafındaki piksellerde çekirdek matrisi ile çarpımda kullanılmaktadır. Burada resimdeki çerçeve içerisindeki her bir pikselin değeri çekirdek matrisindeki karşılığıyla çarpılır ve bütün sonuçlar toplanır. Elde edilen sayı seçili olan pikselin değeri olarak yazılmaktadır.

7	23	50	64	14
15	13	31	46	8
42	25	92	31	32
71	44	74	94	92
2	43	51	35	4

 $\times$ 

0	2	0
0	0	0
0	0	0

 $=$ 

-	-	-	-	-
-	46	100	128	-
-	26	62	92	-
-	50	184	62	-
-	-	-	-	-

## Basit bulanıklık filtresi

$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

```

import sys
import cv2
import numpy as np

image = cv2.imread("doomslayer.png")
height,width,channel = image.shape
cv2.imshow("pencere1",image)

box_kernel = [[1, 1 , 1 ],
               [1, 1 , 1 ],
               [1, 1 , 1 ]]

for y in range(0,height):
    for x in range(0,width):
        toplam = [0,0,0]
        kxmax = 1
        kymax = 1
        kymin = -1
        kxmin = -1
        if (x+1)>=width:
            kxmax = 0
        if (y+1)>=height:
            kymax = 0
        if (x-1)<0:
            kxmin = 0
        if (y-1)<0:
            kymin = 0
        for ky in range(kymin,kymax+1):
            for kx in range(kxmin,kxmax+1):
                lastX = x+kx
                lastY = y+ky
                toplam[0] = toplam[0]+image[lastY,lastX][0]
                toplam[1] = toplam[1]+image[lastY,lastX][1]
                toplam[2] = toplam[2]+image[lastY,lastX][2]
            toplam[0] = toplam[0]/9
            toplam[1] = toplam[1]/9
            toplam[2] = toplam[2]/9
            image[y,x] = toplam
cv2.imshow("pencere2",image)
cv2.waitKey(0)
cv2.destroyAllWindows()

```

Import CV2

pencere1



pencere2

