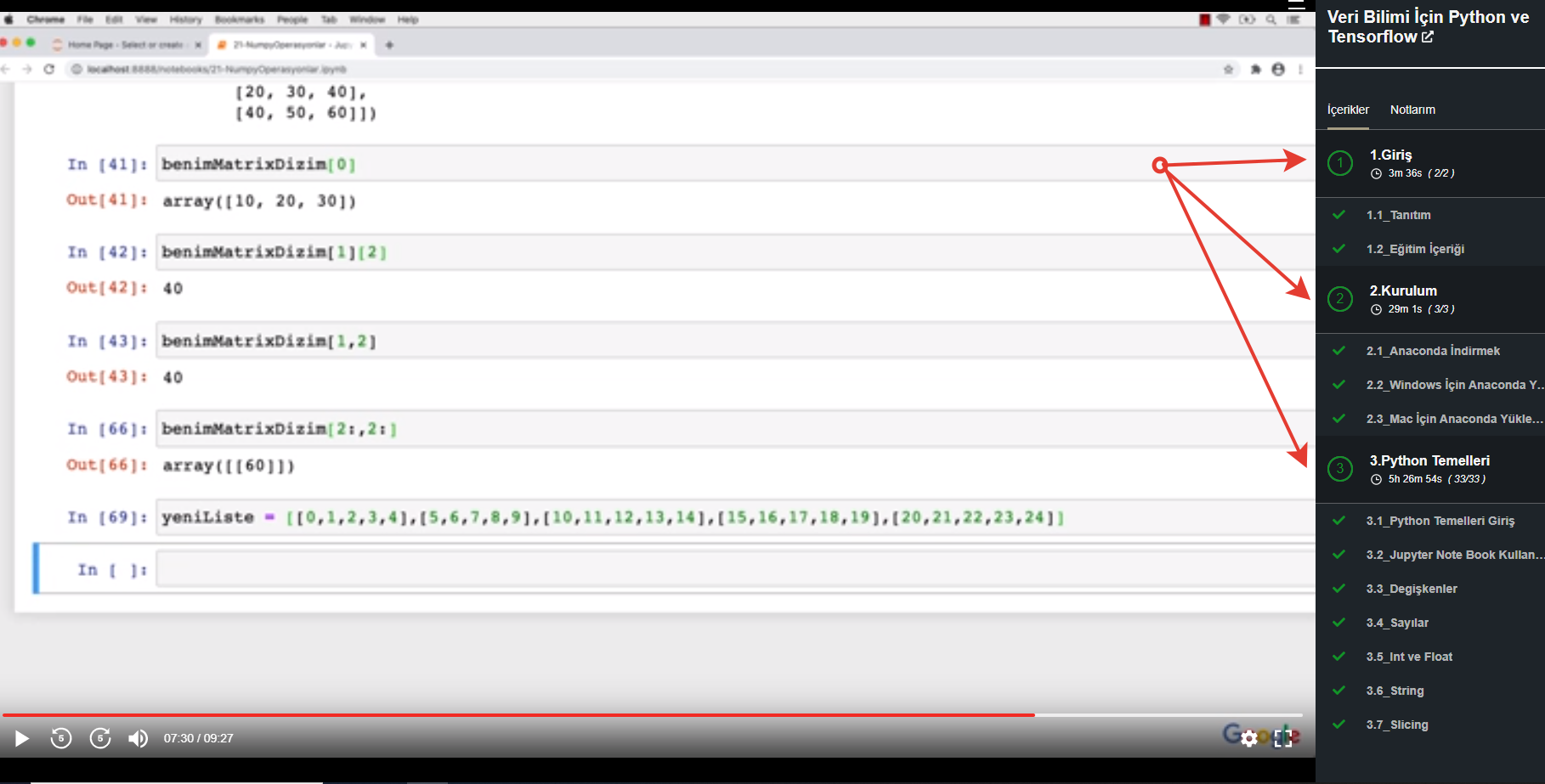
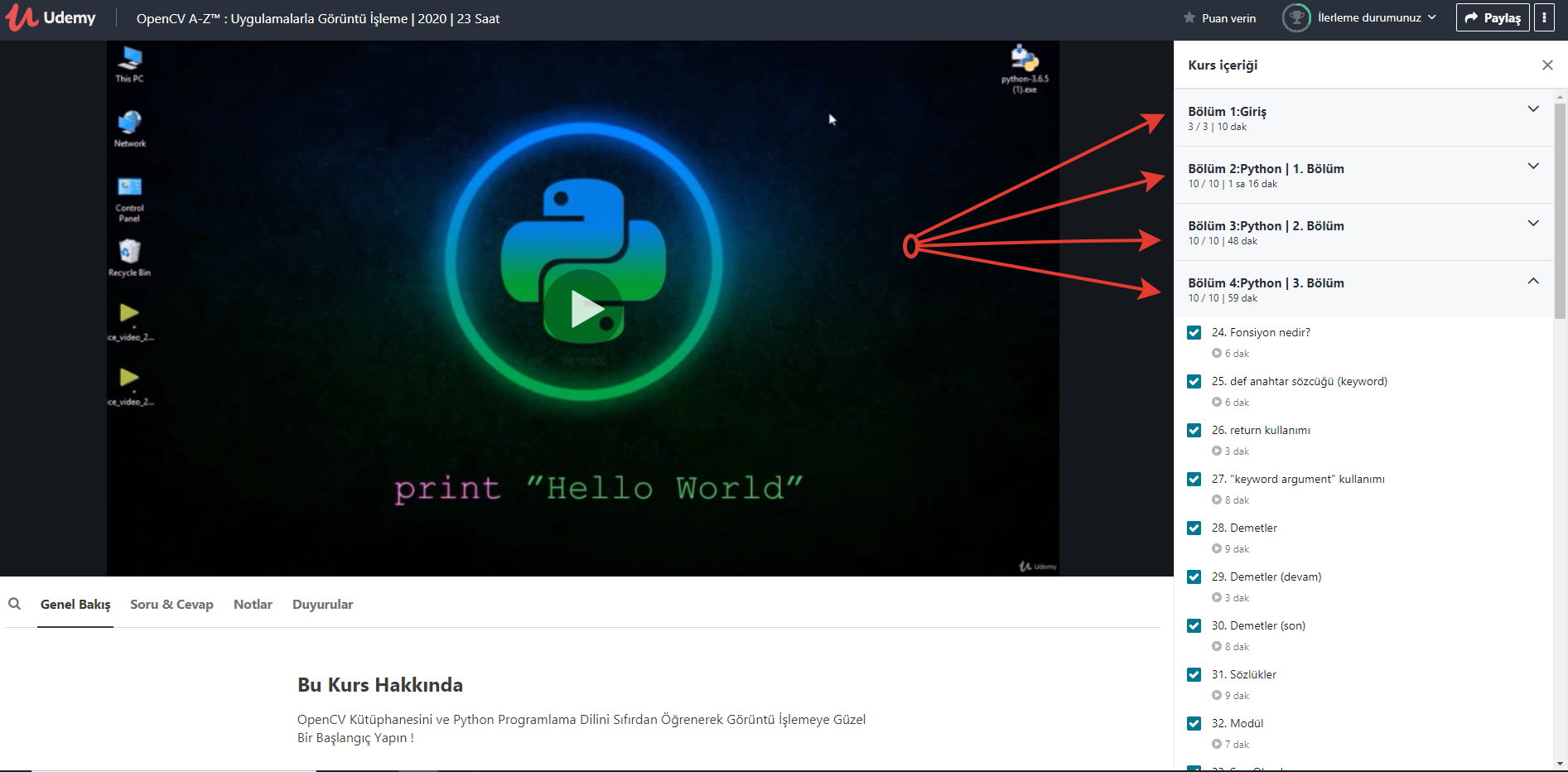
**Python - OpenCV**

**Başlamadan önce yapılan hazırlıklar**

Bu projeye başlamadan önce unutulan bilgilerin hatırlanması ve daha kapsamlı olarak projede kullanılan programlama diline hakim olunması için Udemy’ den Oğuzhan Gürbüz’ün Python-OpenCV ve BTK Akademi’den Atıl Samancıoğlu’ nun “Veri Bilimi İçin Python” kurslarından faydalandım.



Öncelikle BTK Akademi kursundaki Atıl hocanın tüm Python dersleri izlendi ve örnekler uygulandı daha sonrasında ise Udemy’den Oğuzhan Gürbüz’ün Python OpenCV kursuna başlanılarak, kursun başlangıcındaki 3 bölümden oluşan yaklaşık 4 saatlik Python kursu tamamlandı. Python dilinin temelleri ile alakalı eğitim bittikten sonra OpenCV kursuna devam etmekteyim.



Bu aşamada bilgisayarlar verileri ve girdileri nasıl anlarlar, yorumlarlar, bilgisayarlar nasıl görürler, resimler nasıl depolanır ve saklanır, bilgisayarlar gördükleri resim ve videoları nasıl yorumlar, karşılaştırır, sınıflandırır ve anlamlandırır gibi soruların cevabını öğrenmeye çalışacağım.

Bu çalışma esnasında da yapılan uygulamaları ekran görüntüleri ile açıklamaya çalışacağım.

**Neden Python ! Neden OpenCV ?**

Günümüzün hızla gelişen teknoloji çağında her alanda teknolojik ve bilimsel gelişmeler olabildiğince hızlı bir şekilde ilerlemektedir. Yazılım teknolojileri ise neredeyse her gün yeni bir buluş, eklenti, framework veya kütüphaneler ile adeta altın çağını yaşamaktadır.

Bir hevesle ucundan köşesinden tutunmaya çalışarak girmiş olduğum yazılım dünyasına hızlı bir başlangıç yaparak önce boğulmuş ve kafası karışmış olarak devam etsem de sonrasında birçok alanlarda makaleler okuyup, eğitim videoları izleyerek, heves ve merakım doğrultusunda Python programlama dili ve OpenCV kütüphanesi ile karşılaştım. Yapılan işler ve geliştirmeler bende farklı bir heyecan uyandırdığı için yazılım kariyerime bu yönde ilerlemek üzerine karar verdim.

Projem ile ilgili bilgileri vermeden önce bu kısa özeti yaparak neden Python ve OpenCV tercih ettiğimi belirtmek istedim. Sonrasında ise kısa bilgiler vererek, Python dili ve OpenCV kütüphanesi hakkında elde ettiğim bilgi ve deneyimleri paylaşarak devam edeceğim.



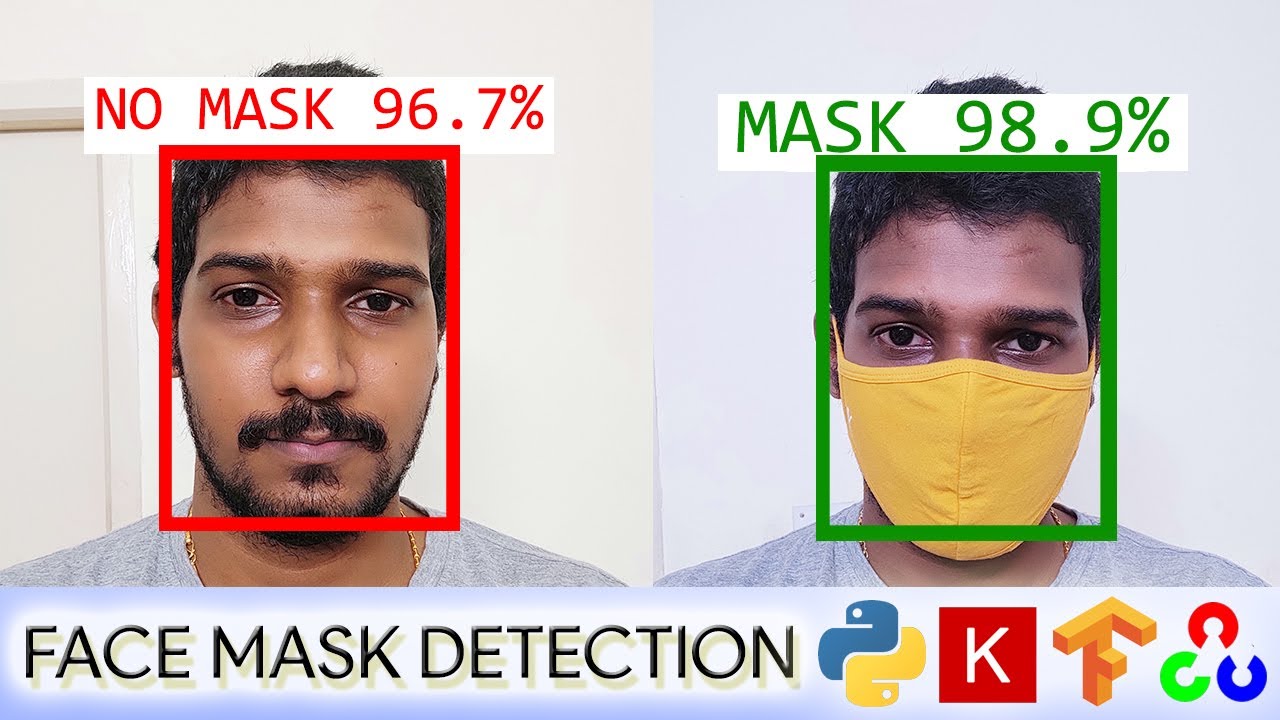
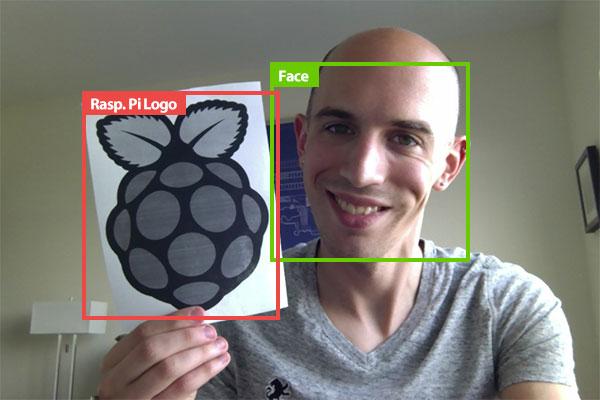
Python dili nesne yönelimli, basit söz dizimine sahip, öğrenme eğrisi kolay ve yüksek seviyeli bir programlama dilidir. Python 1990 yılında Guido Van Rossum tarafından geliştirilmiştir. Özellikle son zamanlarda Python’un bu kadar popüler olmasındaki asıl neden ise, diğer c söz dizimi dillere nazaran daha sade bir syntax yapısının olması, cross platform desteği ve kütüphanelerin zenginliği olarak gösterebiliriz.

OpenCV kütüphanesi ise1999 yılında Intel’den Gary Bradsky tarafından başlatıldı ve ilk sürümü 2000 yılında yayınlandı. Daha sonra bu çalışmalara Vadim Pisarevsky’de katıldı. Halen OpenCV’nin Bilgisayar Görmesi ve Makina Öğrenmesi çalışmalarına yönelik desteği artarak devam etmektedir.

OpenCV kütüphanesi hem C++, Python ve Java gibi dilleri, hem de Linux, Windows, OS X, Android ve iOS gibi platformları desteklemektedir. Bu geniş yelpaze olanakları sayesinde gelişimi oldukça hızlı ilerlemektedir ve en önemlisi kütüphane çalışmaları açık kaynaklıdır.

OpenCV, yüzleri ve nesneleri algılama ve tanımlama, videolarda insani eylemleri sınıflandırma, kamera hareketlerini ve hareketli nesneleri izleme, nesneleri 3 boyutlu modellerine ayıklama, stereo kameralardan 3D nokta bulutları üretme, görüntüleri yüksek çözünürlükte birleştirme gibi alanlarda başarılı alanlarda kullanılabilir.

Örnek olarak aşağıdaki resimlerde gözüktüğü gibi, kamera aracılığı ile yüz ve logo ayrımını yapılabilir veya maskenin takılıp takılmadığını algılayan bir uygulama yazılabilir. OpenCV’nin kullanımı artık hayal gücümüzün yettiği kadar olanak sağlamaktadır.



Tüm yeni geliştirmeler ve haberler www.opencv.org sitesi üzerinden paylaşılmaktadır. Son sürüm OpenCV 4.5.0, 16 Ekim 2020 tarihinde duyurulmuştur.

**Kullanılan Araçlar Ve Kurulumlar**

**-Anaconda Navigator**

**-PyCharm**

**Anaconda Navigator** bir Python dağıtımıdır ve içerisinde birçok kütüphane barındırır. Ve bu kütüphanelere kolayca erişmemizi sağlıyor. Uygulama yaparken, kod yazarken kullanacağımız kütüphaneleri projemize eklememiz bazen gerçekten yorucu ve zor olabiliyor.

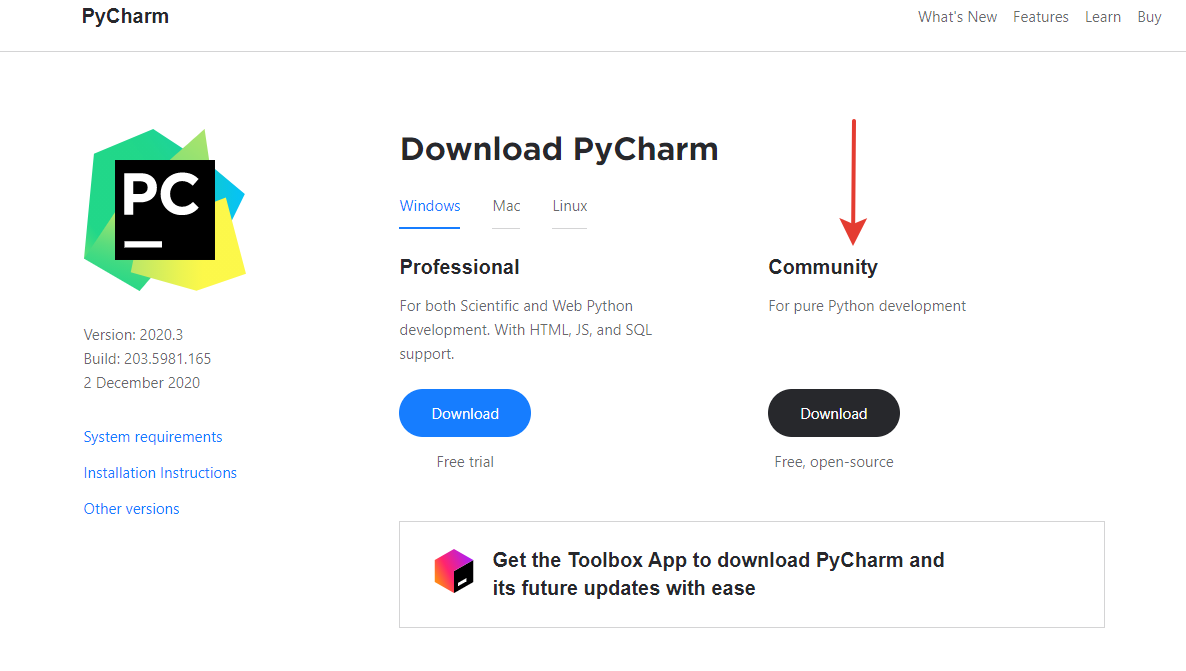
Sürüm uyuşmazlıkları ve hatalı komutlar gibi nedenler dolayısıyla projemize dahil edilemeyen kütüphaneler, **stackoverflow** gibi sitelerde saatlerce çözüm yolu aramamıza neden olabiliyor.

Kütüphaneleri komut satırı ile yüklemek de bir seçenek fakat ben uygulamalarımı komut istemi yerine hem kolaylığı hem de sunduğu zengin kütüphane seçeneğinden dolayı **Anaconda Navigator** kullanarak dahil ettim.

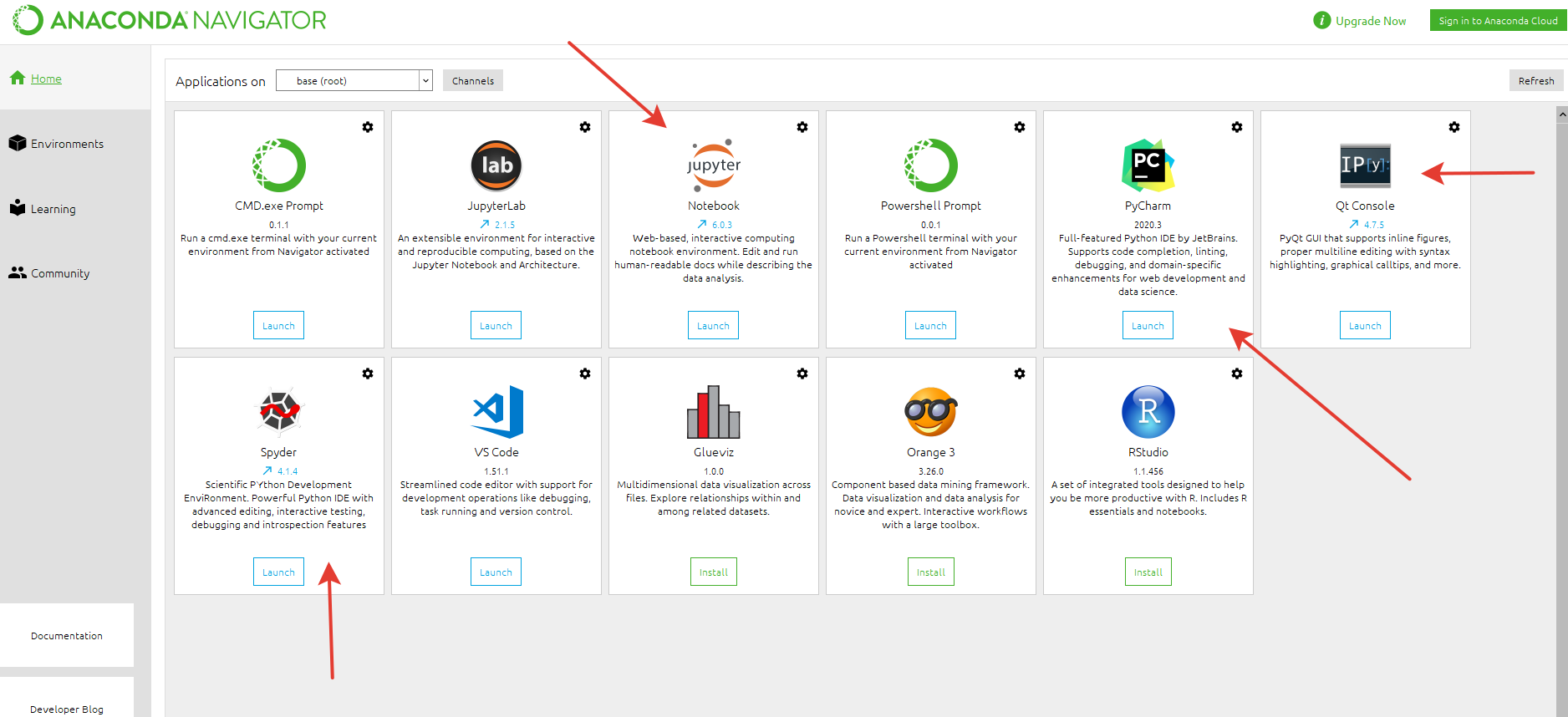
https://www.anaconda.com/products/individual sitesine gidilerek bilgisayar özelliklerine ve platforma göre uygun **Anaconda Navigator** indirilir.

Projelerimde aşağıdaki ekran görüntüsünde kırmızı ok ile gösterilen installer indirilerek kuruldu.

**PyCharm** kurulumu için ise https://www.jetbrains.com/pycharm/download sitesine gidilerek aşağıdaki paket indirilir.



Program kurulumundan sonra ekran görüntüsü aşağıdaki gibidir



Kırmızı oklar ile gösterdiklerim, günümüzde çoğunlukla kullanılan Python geliştirme ortamlarıdır.

**Jupyter Notebook** ve **Spyder** çoğunlukla veri bilimi ve yapay zeka uygulamalarında kullanılmaktadır.

GUI tasarlamak, masaüstü programlar yapmak için ise **Qt Console** kullanılır.

Ben projelerimde yukarıdaki resimde gözüktüğü üzere **PyCharm** kullanarak devam edeceğim. Kullanılan **Python** sürümü ise 3.8.5’dir.

**Proje 1 - OpenCV İle Resim Okuma, Gösterme Ve Kaydetme İşlemleri**

Bu başlık altında ilk olarak resim ve pixel arasındaki ilişkiyi ve OpenCV kütüphanesini kullanarak bir resmi okuma, gösterme ve kaydetme işlemleri yapılacaktır.

Pixel’ler, dijital görüntünün yapı taşlarıdır. Her görüntüde belirli miktarda pixel taneciği bulunur. Bu pixel’ler bilgisayarımızdaki nokta işareti kadardır. İşte bu denli küçük taneciklerin yan yana ve alt alta dizilmesi ile resim dediğimiz yapı oluşur.

Her bir pixel belli bir özelliğe sahiptir. Bu özellik onların rengidir.

Bilgisayarların ve insanların resimleri, görüntüleri algılaması farklıdır. Bilgisayarlar sayıları anlar, insanlar ise gördükleri görsel ögelere anlamlar yükleriz.

Bilgisayarlar ise sayılara anlamlar yüklerler. Resimleri kaydet demek ise aslında pixel ve renk değerlerini kaydetmek demektir.

**Yapılan adımlar ve kodlamalar**

**import cv2** *#Projeye OpenCV kütüphanesi eklenir*

**img = cv2.imread("gunbatimi.jpg")**

*#img değişkeni oluşturulur*

*#Ve cv2.imread fonksiyonu ile projede kullanılacak resimlerin matematiksel değerleri okunur #Fonksiyon içerisine işlenecek olan resim yolu yazılır*

*#Resim ve .py aynı klasörde olduğu için resim adı.uzatısı yazılabilir*

**cv2.namedWindow("image",cv2.WINDOW\_NORMAL)**

*#Bu fonksiyon ile açılan pencerenin boyutu fare ile manuel olarak istenilen*

*#boyuta göre ölçeklenebilir, aksi halde açılan pencere sabit olarak kalır*

*#imshow ile namedWindow fonksiyonlarındaki pencere isimleri aynı olmalıdır*

**cv2.imshow("image",img)**

*#İçerisine iki adet argüman alır, ilki resmin gösterileceği pencerenin adı*

*#ikincisi ise resmimin tutulduğu değişken*

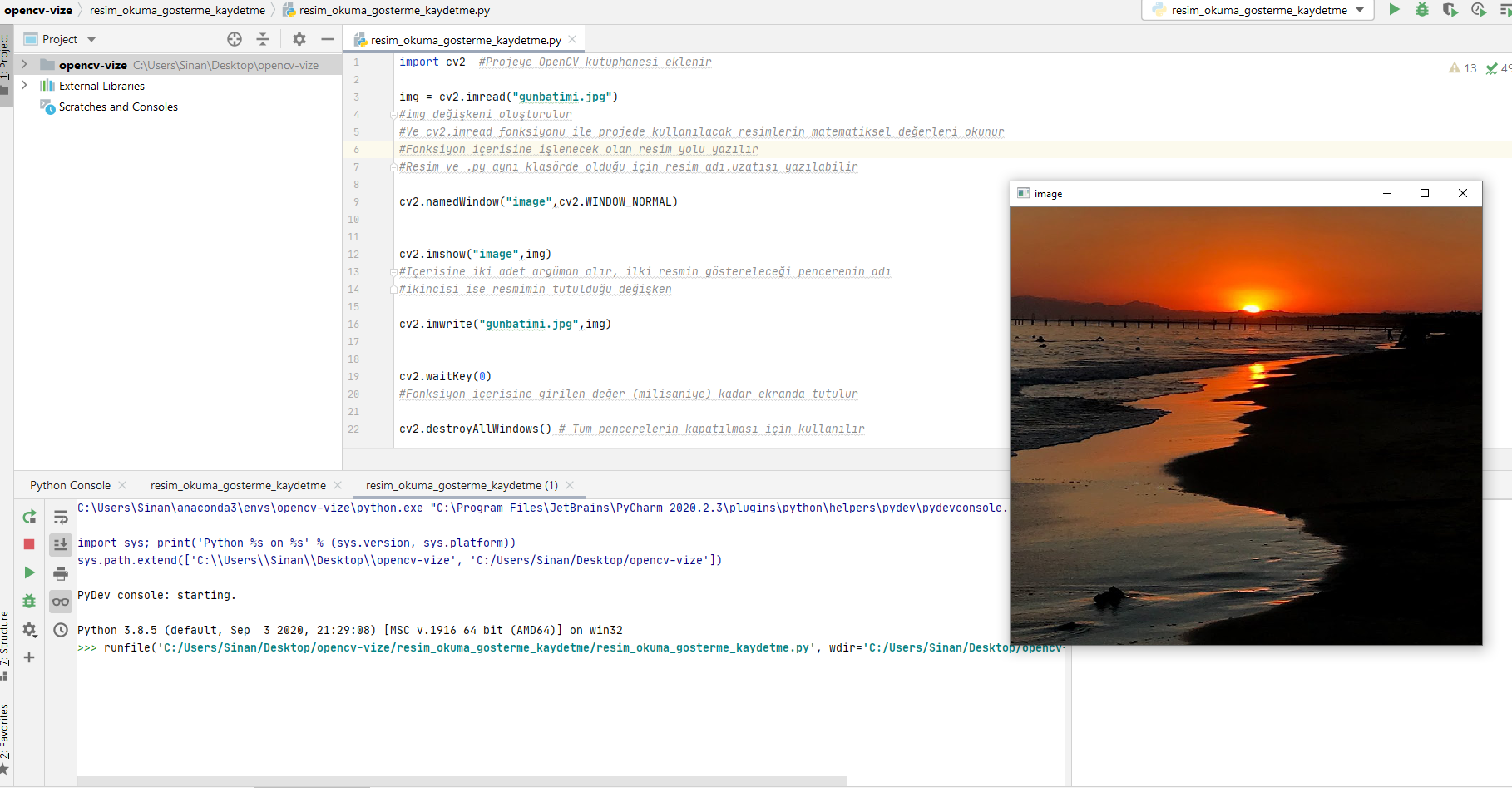
**cv2.imwrite("gunbatimi.jpg",img)**

**cv2.waitKey(0)**

*#Fonksiyon içerisine girilen değer (milisaniye) kadar ekranda tutulur*

**cv2.destroyAllWindows()** *# Tüm pencerelerin kapatılması için kullanılır.*

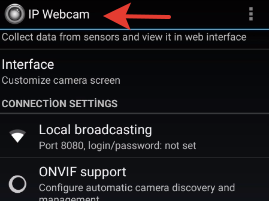
**Programın ekran çıktısı ise şu şekildedir**

****

**Proje 2 - Android Kamera Kullanarak Resim Penceresi Oluşturmak**

Bu projede Python kullanarak Android işletim sistemli telefonu bir web kamerası olarak kullanacağım. Ve telefondan alınan görüntüyü OpenCV kütüphanesi ile alarak, resim formatında bilgisayara kaydedeceğim.

Program android telefona yüklendikten sonra:





Start server aktif edildikten sonra program şu şekilde bir adres verdi 192.168.1.104:8080, bu herkeste aynı değildir ve farklı farklı adresler olabilir.

Daha sonra bu adresi birazdan da açıklayacağım kodlar içerisine yazarak, telefonun kamerasından aktif olarak anlık resim ve video almamı sağlayacaktır.

**Yapılan adımlar ve kodlamalar**

*#Kütüphaneler eklenir*

**import cv2**

**import numpy as np**

**import requests**

**url ="http://192.168.1.104:8080//shot.jpg"**

*#url değişkeni oluşturularak içerisine android kameradan alınan adres yazılır*

**while True:**

**img\_resp = requests.get(url)** *#requests kütüphanesinin fonksiyonu kullanılarak görüntü adresi kullanılır*

**img\_arr = np.array(bytearray(img\_resp.content), dtype=np.uint8)**

*#Alınan görüntü bir array içerisinde tutulur ve bytearray'e çevirilir*

**img = cv2.imdecode(img\_arr, cv2.IMREAD\_COLOR)**

*#Hafızadan çekilen görüntüyü görüntülenebilir hale getirir*

**img = cv2.resize(img,(640,480))** *# Görüntü boyutlandırılır*

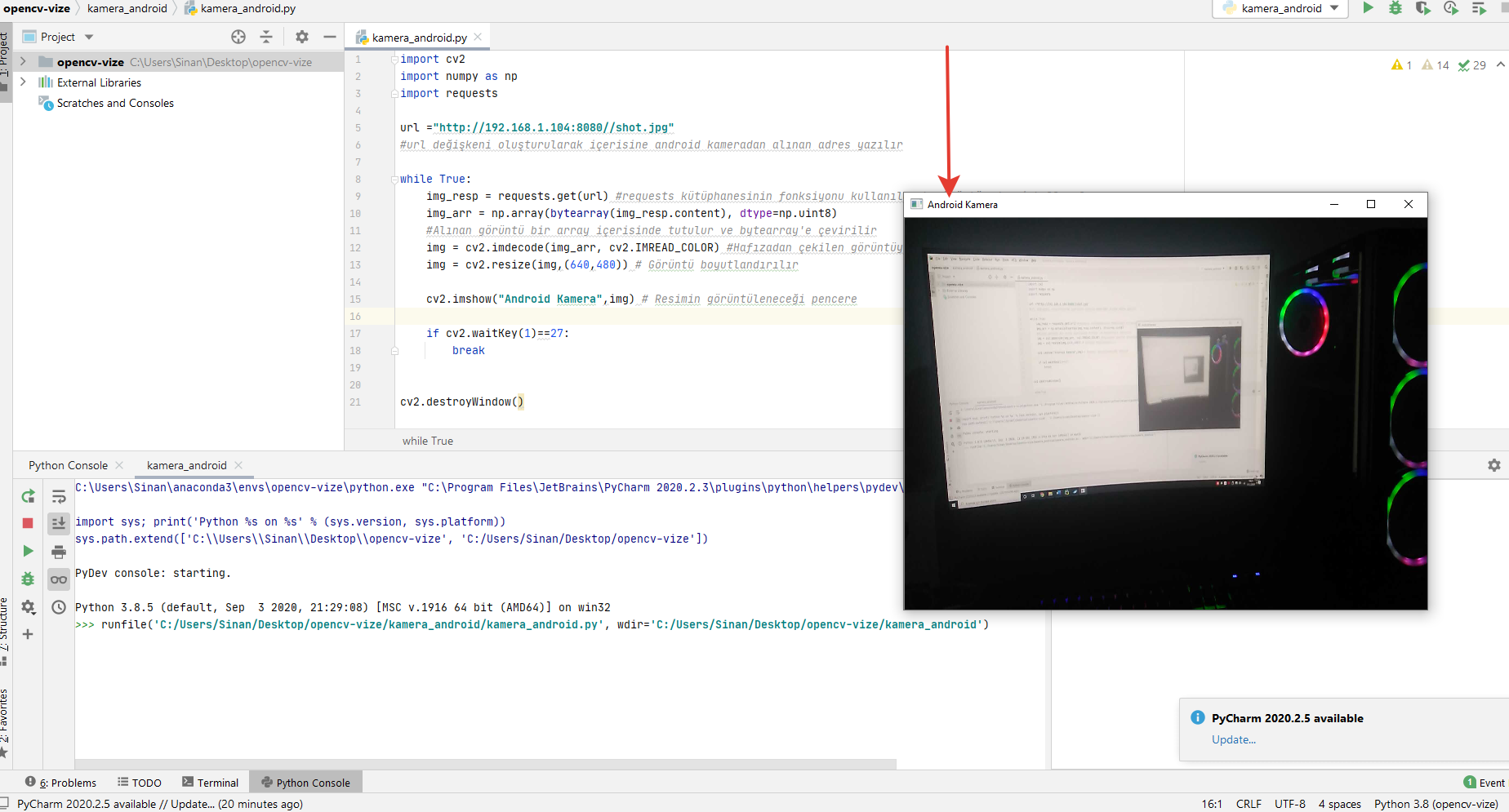
**cv2.imshow("Android Kamera",img)** *# Resimin görüntüleneceği pencere*

**if cv2.waitKey(1)==27:**

**break**

**cv2.destroyWindow()**

**Programın ekran çıktısı ise şu şekildedir**

****

**Proje 3 - Görüntülerle Toplama İşlemi**

• İki adet beyaz tuval oluşturularak , merkezlerine iki adet çember çizilir.

• Oluşturulan bir adet kırmızı ve mavi çemberin renkleri toplanır.

• Tekrardan yeni bir tuval oluşturularak, toplama işlemi sonucu oluşan renk yeni bir tuval ve çember ile gösterilir.

**Yapılan adımlar ve kodlamalar**

**import cv2**

**import numpy as np**

**circle=np.zeros((512,512,3),np.uint8) + 255** *#Beyaz renkte bir tuval oluşturur*

**cv2.circle(circle, (256,256), 60 , (255,0,0) , -1)** *#Oluşturulan beyaz tuvalin merkezine mavi renkte çember çizilir*

**circle2=np.zeros((512,512,3),np.uint8) + 255** *#Tekrardan beyaz renkte bir tuval oluşturur*

**cv2.circle(circle2, (256,256), 60 , (0,0,255) , -1)** *#Oluşturulan beyaz tuvalin merkezine kırmızı renkte çember çizilir*

**add=cv2.add(circle,circle2)** *#add değişkeni oluşturularak çemberlerin değeri toplanır* **print(add)** *#Consol ekranında oluşturulan toplama işleminin sayısal değerini görmek istediğim için ekledim*

**cv2.imshow("Circle",circle)**

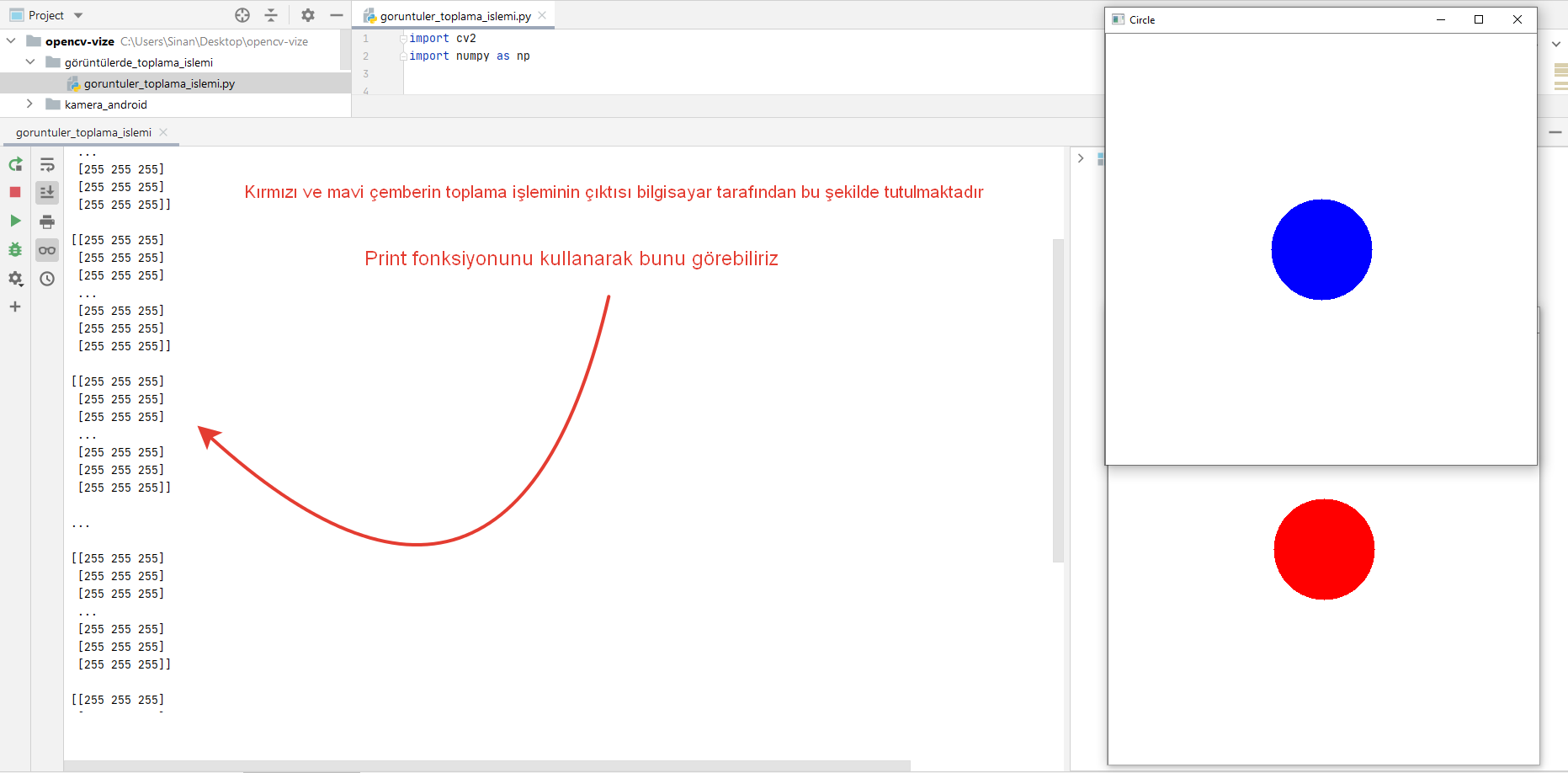
**cv2.imshow("Circle2",circle2)**

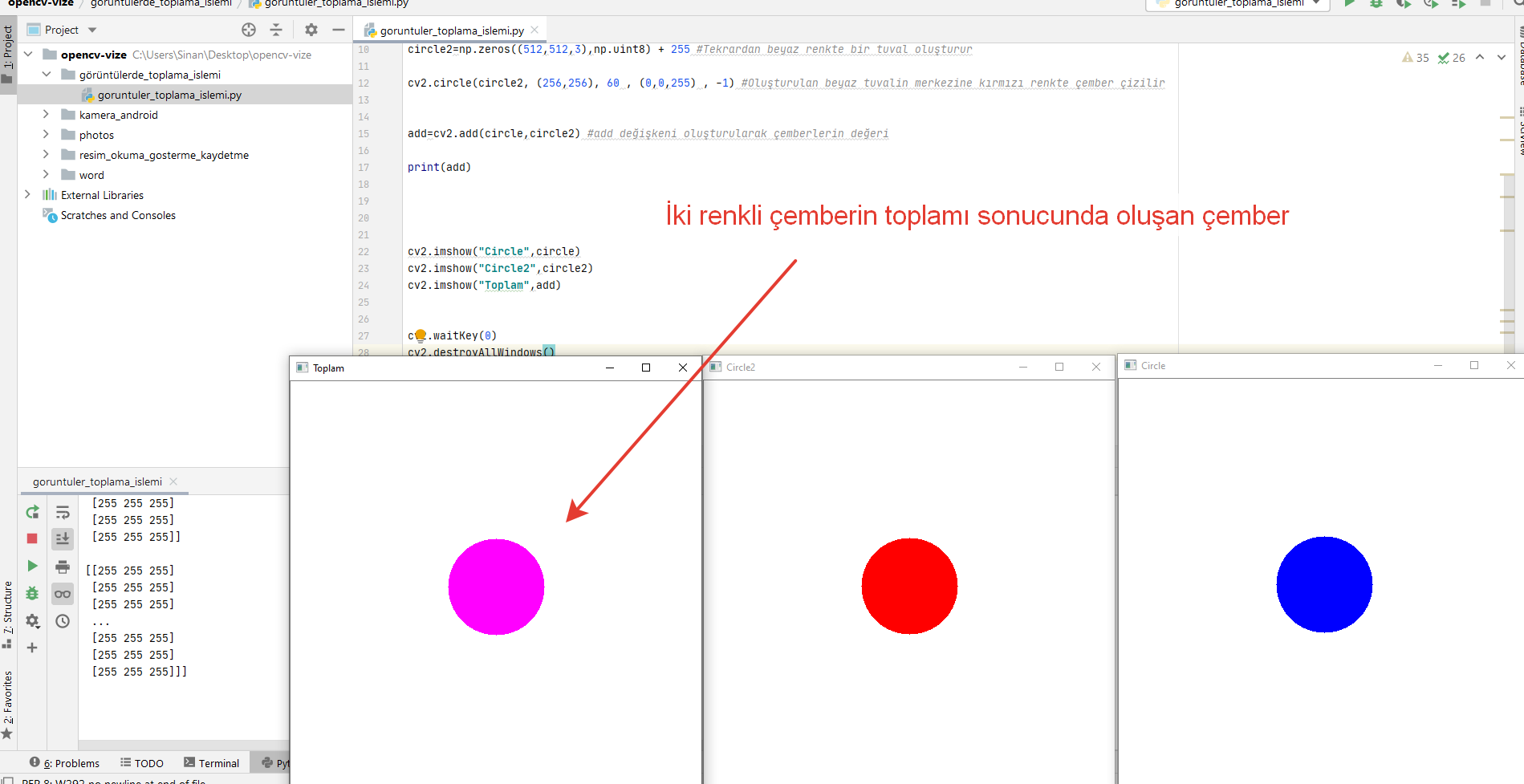
**cv2.imshow("Toplam",add)**

**cv2.waitKey(0)**

**cv2.destroyAllWindows()**

**Programın ekran çıktısı ise şu şekildedir**

****

****

**Proje 4 - Resimden Yüz Algılama İşlemleri**

Burada önce gerekli kütüphaneleri import ettikten sonra Haar cascade classifier dosyalarının yerini belirtiyoruz. Sonra programa verilen resim dosyasını OpenCV ile okuyup siyah-beyaz yapıyoruz, yüz tanıma işlemi renkli grafiklerle değil siyah-beyaz grafiklerle çalışır.

Bu kütüphane ve kodlar  ile Python programlarımızın resimler üzerindeki nesneleri tespit etmesini ve tanımasını sağlayabiliriz, örneğin insan resimlerindeki yüzleri.

**Yapılan adımlar ve kodlamalar**

# 1. Kullanacağımız kütüphane çalışmamıza dahil edilir.

**import cv2**

# 2. Kullanacağımız resim çalışmamıza dahil edilir.

**img = cv2.imread("C:\\Users\\Sinan\\Desktop\\opencv-final\\photos\\sinan3.jpg")**

# 3. Kullanacağımız cascade dosyası çalışmamıza dahil edilir.

**face\_cascade = cv2.CascadeClassifier("C:\\Users\\Sinan\\Desktop\\opencv-final\\frontalface.xml")**

# 4. Haar-like özellikleri kolay algılayabilmek için resmimizi boz(gri) tonlara çevirelim.

**gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)**

# 5. Şimdi de cascade dosyamızı kullanarak resim üzerindeki yüzlerin koordinarlarını bulalım.

**faces = face\_cascade.detectMultiScale(gray, 1.3, 7)**

# 6. "faces" değişkeninde tuttuğumuz koordinatları kullanarak yüzleri dikdörtgen içerisine alalım.

**for (x,y,w,h) in faces:**

**cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255),2)**

# 7. İşlediğimiz resmi görelim.

**cv2.imshow('Sinan Saricayir - Final Projesi ',img)**

# 8. Son olarak programı kapatacak kodu yazalım.

**cv2.waitKey(0)**

**cv2.destroyAllWindows()**

**Programın ekran çıktısı ise şu şekildedir**

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Proje 5 - Videodan Yüz Algılama İşlemleri**

Aynı şekilde gerekli kütüphaneler dahil edildikten sonra bu sefer video kullanılarak yüzlerin yakalanması sağlanmıştır.

Ekran görüntüsü ve gif resimleri aşağıdaki gibidir.

**Yapılan adımlar ve kodlamalar**

*# 1. Kullanacağımız kütüphaneyi çalışmamıza dahil edelim.***import cv2**   
  
  
*# 2. Kullanacağımız videoyu çalışmamıza dahil edelim.***vid = cv2.VideoCapture("C:\\Users\\Sinan\\Desktop\\opencv-final\\videos\\yuzler.mp4")**  
*# 3. Kullanacağımız cascade dosyasını çalışmamıza dahil edelim.***face\_cascade = cv2.CascadeClassifier("C:\\Users\\Sinan\\Desktop\\opencv-final\\frontalface.xml")**  
*#4. Sonsuz bir döngü ile her kareyi(frame) tek tek inceleyelim.***while 1:** *# 5. Her kareyi tek tek okuyalım.* **\_, frame = vid.read()** *# 6. Haar-like özellikleri kolay algılayabilmek için her bir kareyi boz(gri) tonlara çevirelim.***gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)** *# 7. Şimdi de cascade dosyamızı kullanarak her bir kare üzerindeki yüzlerin koordinarlarını bulalım.* **faces = face\_cascade.detectMultiScale(gray, 1.4, 1)** *# 7. "faces" değişkeninde tuttuğumuz koordinatları kullanarak yüzleri dikdörtgen içerisine alalım.***for (x,y,w,h) in faces:  
 cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,0),2)** *# 8. İşlediğimiz kareleri görelim.***cv2.imshow('image',frame)**  
 *# 9. Programı kapatacak kodu yazalım.***if cv2.waitKey(1) & 0xFF == ord('q'):  
 break**  
*# 10. Son olarak videoyu serbest bırakalım.***vid.release()  
cv2.destroyAllWindows()**

**Programın ekran çıktısı ise şu şekildedir**

**metin içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**metin içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**Proje 6 – Resimden Göz Algılama İşlemleri**

Yüz ve göz cascade doyalarımızı kodlaramıza dahil ettikten sonra , resimlerden yüz ve göz algılma işlemi yapılacaktır.

**Yapılan adımlar ve kodlamalar**

*# 1. Kullanacağımız kütüphaneyi çalışmamıza dahil edelim.***import cv2**  
  
*# 2. Kullanacağımız videoyu çalışmamıza dahil edelim.***img = cv2.imread("C:\\Users\\Sinan\\Desktop\\opencv-final\\photos\\goz.png")**  
*# 3. Kullanacağımız cascade dosyalarını çalışmamıza dahil edelim.***face\_cascade = cv2.CascadeClassifier("C:\\Users\\Sinan\\Desktop\\opencv-final\\frontalface.xml")  
eye\_cascade = cv2.CascadeClassifier("C:\\Users\\Sinan\\Desktop\\opencv-final\\eye.xml")***# 6. Haar-like özellikleri kolay algılayabilmek için her bir kareyi boz(gri) tonlara çevirelim.***gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)**  
   
*# 7. Şimdi de cascade dosyamızı kullanarak her bir kare üzerindeki yüzlerin koordinarlarını bulalım.***faces = face\_cascade.detectMultiScale(gray)***# 8. "faces" değişkeninde tuttuğumuz koordinatları kullanarak yüzleri dikdörtgen içerisine alalım.***for (x,y,w,h) in faces:  
 cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),2)***# 9. Şimdi de, bulduğum yüzler içinde göz arayacağım.***gray2 = gray[y:y+h, x:x+w]  
img2 = img[y:y+h, x:x+w]***# 10. eye cascade dosyasını kullanarak gözlerin koordinatlarını bulalım.***eyes = eye\_cascade.detectMultiScale(gray2)**  
  
*# 11. bu koordinatlara dikdörtgen çizelim.***for (ex,ey,ew,eh) in eyes:  
 cv2.rectangle(img2,(ex,ey),(ex+ew,ey+eh),(0,0,255),2)***# 12. İşlediğimiz resmi görelim.***cv2.imshow('image',img)**  
*# 13. Programı kapatacak kodu yazalım.***cv2.waitKey(0)  
cv2.destroyAllWindows()**

**Programın ekran çıktısı ise şu şekildedir**

**metin içeren bir resim

Açıklama otomatik olarak oluşturuldu**