* **Parking Space Counter Python Code**
* **Parking Space Picker Module 1 Code**

import cv2  
import pickle  
  
width, height = 107, 48  
  
try:  
 with open('CarParkPos', 'rb') as f:  
 posList = pickle.load(f)  
except:  
 posList = []  
  
  
def mouseClick(events, x, y, flags, params):  
 if events == cv2.EVENT\_LBUTTONDOWN:  
 posList.append((x, y))  
 if events == cv2.EVENT\_RBUTTONDOWN:  
 for i, pos in enumerate(posList):  
 x1, y1 = pos  
 if x1 < x < x1 + width and y1 < y < y1 + height:  
 posList.pop(i)  
  
 with open('CarParkPos', 'wb') as f:  
 pickle.dump(posList, f)  
  
  
while True:  
 img = cv2.imread('carpark.png')  
 for pos in posList:  
 cv2.rectangle(img, pos, (pos[0] + width, pos[1] + height), (255, 0, 255), 2)  
  
 cv2.imshow("Image", img)  
 cv2.setMouseCallback("Image", mouseClick)  
 cv2.waitKey(1)

* **Main Module 2 Code**

import cv2  
import pickle  
import cvzone  
import numpy as np  
  
# Video feed  
cap = cv2.VideoCapture('carpark.mp4')  
  
with open('CarParkPos', 'rb') as f:  
 posList = pickle.load(f)  
  
width, height = 107, 48  
  
  
def checkParkingSpace(imgPro):  
 spaceCounter = 0  
  
 for pos in posList:  
 x, y = pos  
  
 imgCrop = imgPro[y:y + height, x:x + width]  
 # cv2.imshow(str(x \* y), imgCrop)  
 count = cv2.countNonZero(imgCrop)  
  
 if count < 900:  
 color = (0, 255, 0)  
 thickness = 5  
 spaceCounter += 1  
 else:  
 color = (0, 0, 255)  
 thickness = 2  
  
 cv2.rectangle(img, pos, (pos[0] + width, pos[1] + height), color, thickness)  
 cvzone.putTextRect(img, str(count), (x, y + height - 3), scale=1,  
 thickness=2, offset=0, colorR=color)  
  
 cvzone.putTextRect(img, f'Free: {spaceCounter}/{len(posList)}', (100, 50), scale=3,  
 thickness=5, offset=20, colorR=(0, 200, 0))  
  
  
while True:  
  
 if cap.get(cv2.CAP\_PROP\_POS\_FRAMES) == cap.get(cv2.CAP\_PROP\_FRAME\_COUNT):  
 cap.set(cv2.CAP\_PROP\_POS\_FRAMES, 0)  
 success, img = cap.read()  
 imgGray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  
 imgBlur = cv2.GaussianBlur(imgGray, (3, 3), 1)  
 imgThreshold = cv2.adaptiveThreshold(imgBlur, 255, cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C,  
 cv2.THRESH\_BINARY\_INV, 25, 16)  
 imgMedian = cv2.medianBlur(imgThreshold, 5)  
 kernel = np.ones((3, 3), np.uint8)  
 imgDilate = cv2.dilate(imgMedian, kernel, iterations=1)  
  
 checkParkingSpace(imgDilate)  
 cv2.imshow("Image", img)  
 # cv2.imshow("ImageBlur", imgBlur)  
 # cv2.imshow("ImageThres", imgMedian)  
 cv2.waitKey(10)