به نام خدا

**مبانی بینایی کامپیوتر**

**دکتر محمدی**

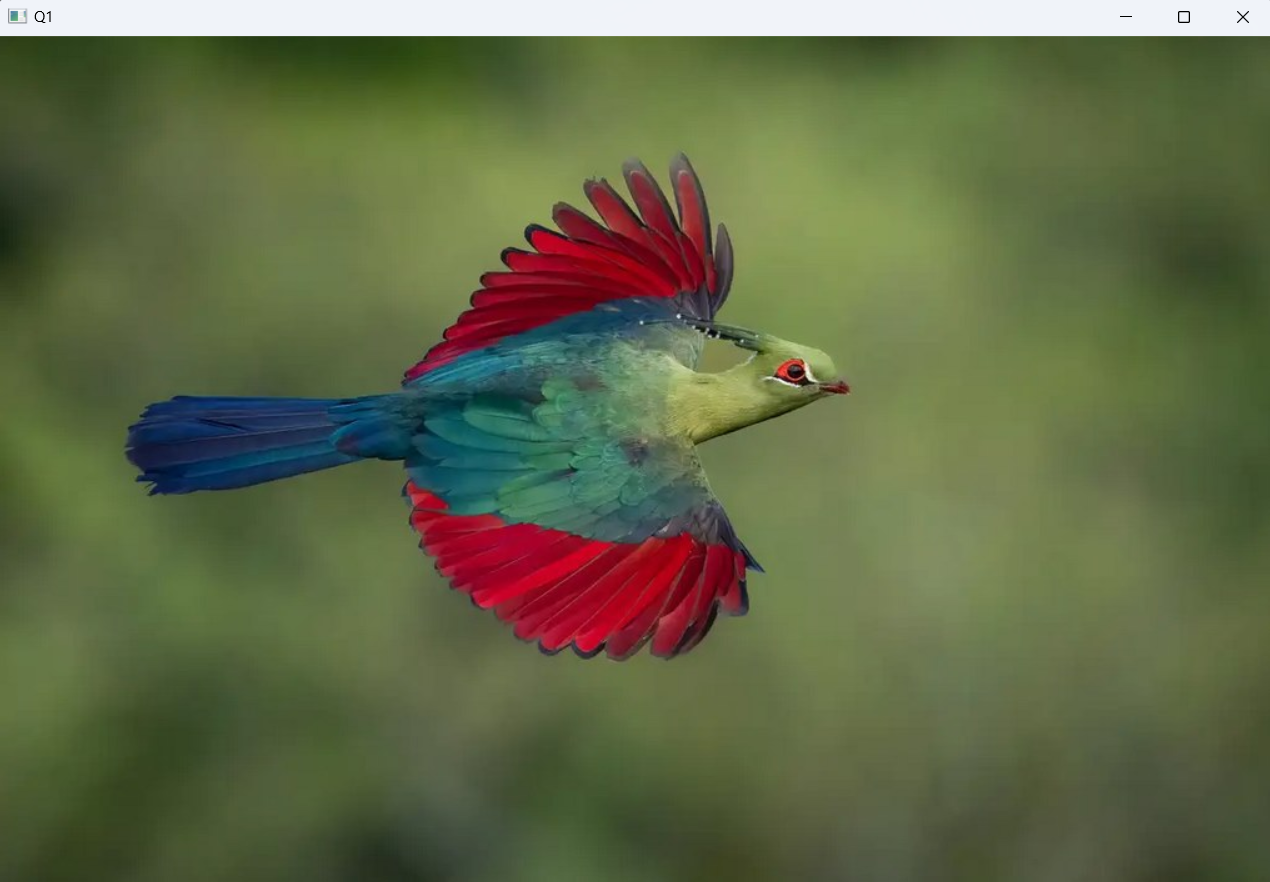
**تمرین صفر**

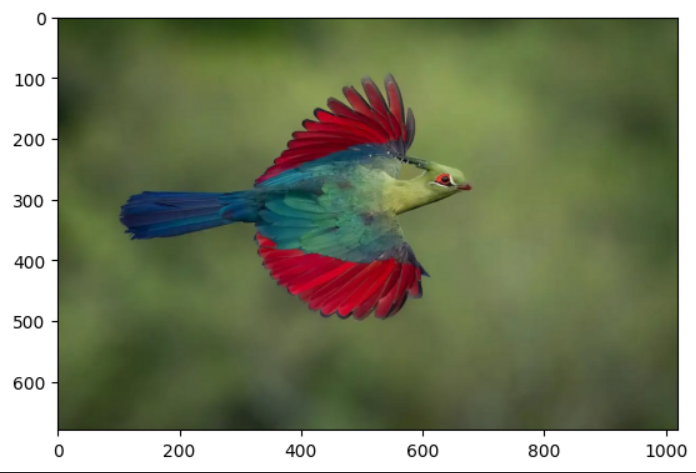
علی عطاریان – ۹۹۵۲۱۴۵۱

**سوال ۱) الف)**

در هر دو کتابخانه cv2 و matplotlib از دستور imread برای خواندن تصویر و از دستور imshow برای نمایش تصویر استفاده می‌شود.

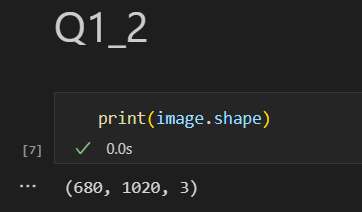
خروجی cv2:



خروجی matplotlib:

تفاوت این دو آن است که cv2 کانالهای رنگ را به فرمت bgr و matplotlib به فرمت rgb ذخیره می‌کند پس اگر تصویر دریافت شده از matplotlib را با دستور cv2.imshow نمایش بدهیم، خواهیم داشت:

این مشکل را با دستور cv2.cvtColor و ویژگی cv2.COLOR\_RGB2BGR برطرف می‌سازیم.

**سوال ۱) ب)**

خروجی نشان می‌دهد که یک آرایه سه بعدی داریم که بعد اول آن ۶۸۰ درایه، بعد دوم ۱۰۲۰ درایه و بعد سوم آن ۳ درایه دارد. این بدین معناست که یک تصویر ۶۸۰\*۱۰۲۰ پیکسلی سه کاناله داریم.

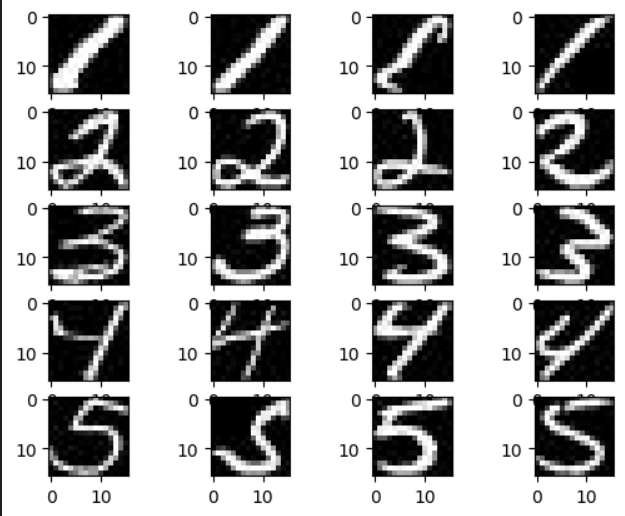
**سوال ۱) ج)**

با توجه به اینکه بعد از ساختن پوشه موردنظر، اگر دوباره دستور os.makedirs را اجرا کنیم ارور دریافت می‌کنیم پس از try استفاده می‌کنیم. تصاویر را با استفاده از cv2.cvtColor و آرگومان COLOR\_BGR2GRAY سیاه و سفید می‌کنیم. تصاویر را با دستور cv2.imwrite ذخیره می‌کنیم. با استفاده از دستورات open,write,close و تبدیل ابعاد آرایه به رشته حروف، فایل متنی موردنظر را ذخیره می‌کنیم.

**سوال ۲)**

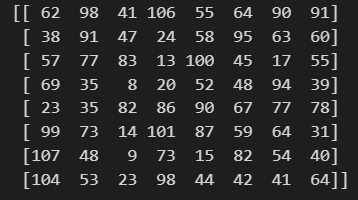
تابع resize را با استفاده از cv2.resize پیاده‌سازی می‌کنیم. در تابع crop هرکدام از ۵ تصویر را به ۱۱۰۰ تصویر مجزا تبدیل می‌کنیم. در واقع تصویر را با step هایی به اندازه cropsize طی و جدا می‌کنیم. با استفاده از np.sum مطمئن می‌شویم تصویر جدا شده خالی نیست.

تصاویر را می‌خوانیم و با ویژگی cv2.IMREAD\_GRAYSCALE مطمئن می‌شویم shape تصویر صحیح است. سپس تابع کراپ کردن را فراخوانی می‌کنیم.

با np.random.randint(1100) یک اینتجر رندوم از ۰ تا ۱۱۰۰ تولید می‌کنیم، همچنین از یک پلات ۴ در ۵ با دستور fig.add\_subplot(5,4,k) برای نمایش دادن خروجی استفاده می‌کنیم:

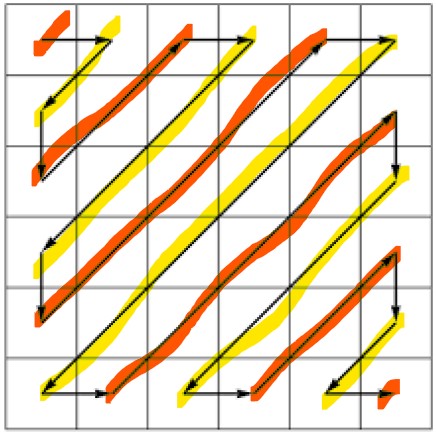
**سوال ۳) الف)**

در پیاده سازی تابع create\_matrix(n) از

np.random.randint(n,n+100, size=(n, n)) استفاده کردیم که آرایه‌ای به شکل n\*n بازمی‌گرداند که درایه‌های آن با اینتجرهای تصادفی در بازه [n,n+100) پر شده‌اند.

پس از مسطح کردن ماتریکس با دستور matrix.flatten() تعداد هر عدد را در ارقام درایه‌های ماتریس می‌شماریم.

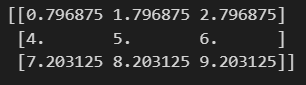


**سوال ۳) ب)**

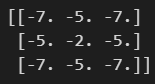
در این تصویر خط‌های مورب نارنجی و زرد را روی ماتریس می‌بینیم. روی هر خط مورب i+j یکسان است و می‌بینیم خانه‌ای که i+j بیشتری دارد دیرتر پیمایش شده. بین خانه‌هایی که i+j برابر دارند، در خطوط نارنجی خانه‌هایی که در پیمایش عادی دیرتر دیده می‌شوند، در پیمایش زیگزاگی زودتر پیمایش می‌شود و در خطوط زرد برعکس. خطوط زرد خانه‌هایی را شامل می‌شوند که باقیمانده i+j آنها بر ۲ یک می‌باشد و خطوط نارنجی باقیمانده i+j آنها بر ۲ صفر می‌باشد. این منطق در کد پیاده‌سازی شده است. هر [] در solution نشان دهنده خانه‌هایی با i+j یکسان می‌باشد که به ترتیب می‌باشند. اگر 0i+j%2= آنگاه عضو جدید به اول لیست می‌رود و در غیراینصورت به آخر لیست می‌رود. در نهایت درایه‌های این آرایه دو بعدی به ترتیب چاپ می‌شوند:

**سوال ۴) الف)**

با استفاده از np.transpose ماتریس را ترانهاده می‌کنیم و با دستور np.dot ضرب ماتریس‌ها را انجام می‌دهیم و از np.linalg.inv برای ماتریس وارون استفاده می‌کنیم. خروجی:

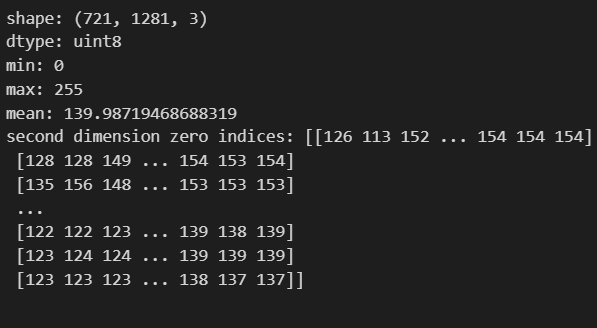


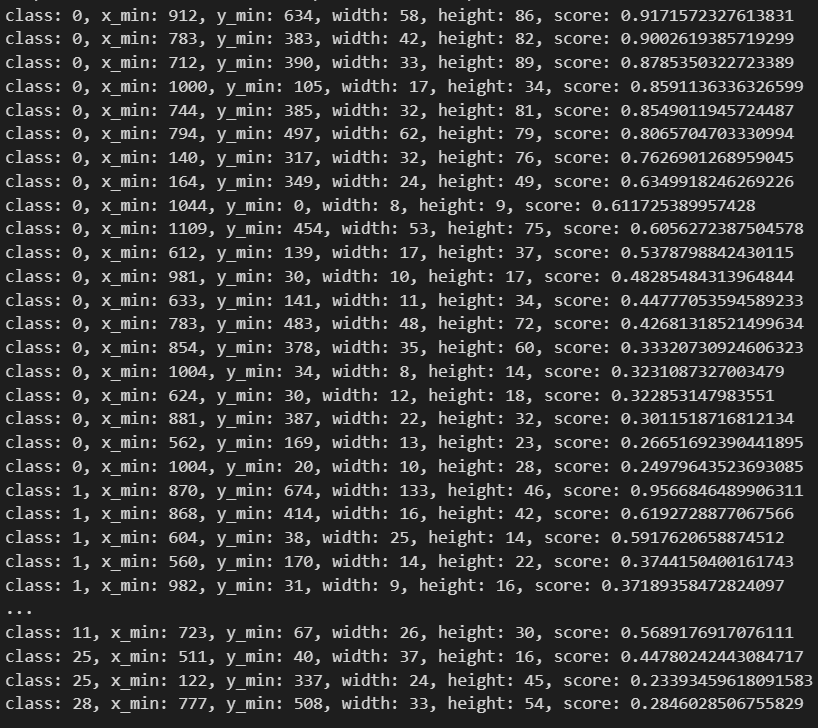
**سوال ۴) ب)**

با استفاده از دو فور تو در تو عمل لغزش را انجام داده و در هر iteration یک آرایه ۳\*۳ با جایگاه مناسب از ماتریس B جدا کرده تا ضرب و جمع موردنظر را انجام دهیم و حاصل را در نهایت در ماتریس جواب قرار دهیم.

**سوال ۵)**

ابعاد تصویر را np.shape، نوع داده را image.dtype، مینیموم را np.min، ماکسیموم را np.max و میانگین را np.mean نشان می‌دهد. همچنین با [:,:,0] تمام درایه‌های مربوط به کانال رنگ آبی نمایش داده می‌شود.

آرایه detentions ۴۹ آیتم دارد که هر کدام دو بعدی می‌باشند. درایه اول بُعد اول تا درایه چهارم بعد اول به ترتیب x\_min,y\_min,width,height را نشان می‌دهند. و بُعد دوم نیز میزان اطمینان از تشخیص خود (امتیاز) را نشان می‌دهد. همچنین classes آرایه‌ای می‌باشد که classid مربوط به هرکدام از ۴۹ شئ را در خود دارد.



ابزاری برای مستطیل کشیدن روی تصویر cv2.rectangle می‌باشد. آرگومان‌هایی که به این تابع داده‌ایم به ترتیب بدین معنا می‌باشند: تصویر موردنظر، ابتدای مستطیل (چپ بالا)، انتهای مستطیل (راست پایین)، رنگ و ضخامت. در نهایت تصویر را همانند سوالات قبلی ذخیره می‌کنیم و نمایش می‌دهیم.

