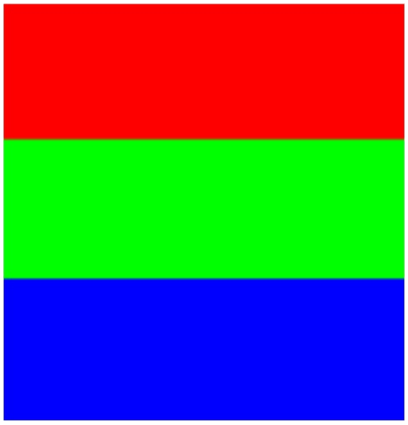
کار بر روی فضاهای رنگی:

همانطور که گفتیم در یک عکس height بیانگر تعداد سطرها و width بیانگر تعداد ستون ها است. اگر به تصویر زیر دقت کنید:



احتمالا می توانید حدس بزنید که این تصویر ماتریسی است که بخش بالایی آن کاملا قرمز است [[0,0,255]]، بخش میانی ان کاملا سبز است [[0,255,0]] و در نهایت آخرین بخش آن کاملا آبی است [[255,0,0]] . در این تصویر هیچ ترکیب رنگی اتفاق نیفتاده است و هر کانال کاملا به صورت مجزا نمایش داده می شود. با استفاده از دستور زیر می توان هر کانال را کاملا از هم جدا نمود:

```
b,g,r = cv2.split(img)
b
```

با انجام این دستور می توانیم خروجی را به شکل زیر برای هر کدام از کانال ها به صورت مجرا داشته باشیم:

```
array([[  0,   0,   0, ...,   0,   0,   0],
       [  0,   0,   0, ...,   0,   0,   0],
       [  0,   0,   0, ...,   0,   0,   0],
       ...,
       [255, 255, 255, ..., 255, 255, 255],
       [255, 255, 255, ..., 255, 255, 255],
       [255, 255, 255, ..., 255, 255, 255]], dtype=uint8)
```

در واقع می دانیم که عکس ما به صورت تعدادی سطر و ستون و همچنین 3 کانال رنگی بوده که با این روش توانسته ایم هر کدام از آنها را به صورت مجزا به دست بیاوریم.

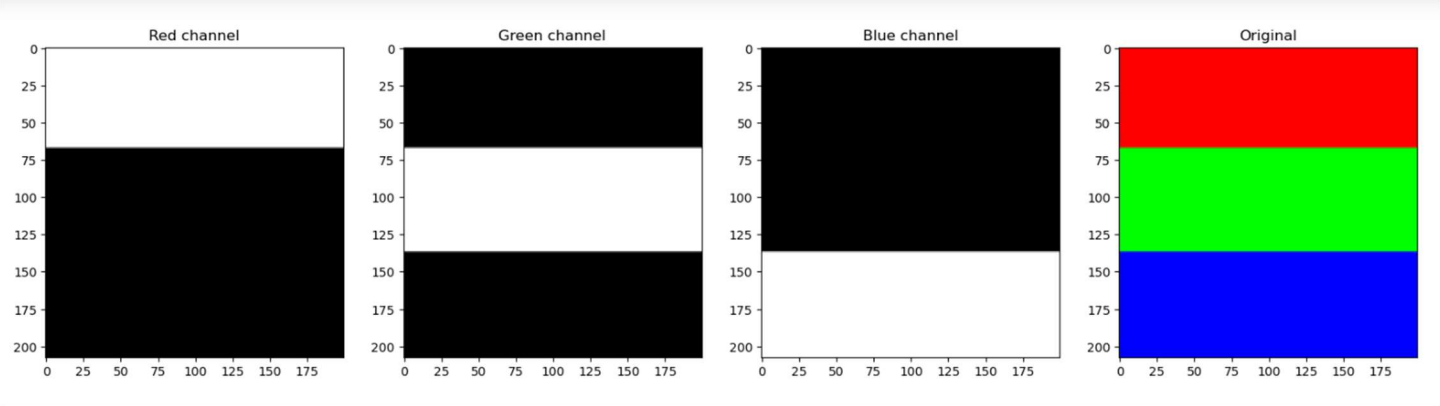
اکنون می توانیم با استفاده از کتابخانه matplotlib چند عکس را در خروجی درون یک سطر نمایش دهیم. برای این کار می توانیم از روش شی گرایی در این کتابخانه استفاده کنیم.

```
# show image with matplotlib
plt.figure(figsize=[20,5]) # width * height
plt.subplot(141);plt.imshow(r, cmap= 'gray');plt.title('Red channel')
plt.subplot(142);plt.imshow(g,cmap= 'gray');plt.title('Green channel')
```

```
plt.subplot(143);plt.imshow(b,cmap= 'gray');plt.title('Blue channel')

# show the original image
plt.subplot(144);plt.imshow(img[...,::-1]);plt.title('Original')
```

همانطور که پیشتر بیان در کانال های سطح خاکستری صفر به معنی عدم و جود پیکسل و رنگ سیاه است و 255 به معنی و جود یک پیکسل و رنگ سفید است. اگر به خروجی کد بالا در نوت بوک خود دقت کنید ما هر کانال را با سطح خاکستری نمایش داده ایم. و چون می دانیم در عکس فوق هر کانال رنگی دارای شدت 255 می باشد بنابراین انتظار خواهیم داشت که این کانال ها در سطح خاکستری به رنگ سفید مطلق نمایش داده شوند.



حال که با نحوه ترکیب این کانال ها آشنا شدیم می توانیم همین کار را با سایر تصاویر نیز انجام دهیم. پیشنهاد می شود برای درک بهتر این موضوع عملیات جداسازی تصاویر را بر روی عکس های واقعی انجام داده و نتایج را با هم مقایسه نمایید.

ادغام کانال های رنگی:

بعد از جداسازی کانال های رنگی ما می توانیم همان تصاویر را دوباره و این بار با استفاده از دستور merge به شکل زیر با مه ادغام نماییم.

```
cv2.merge([b,g,r])
```

نکته مهمی که در زمان ادغام کانال ها باید به ان توجه کرد این است که بعد تمامی کانل ها باید دقیقا شبیه هم باشند و ما نمی توانیم کانال هایی با ابعاد متفاوت را با هم ادغام نماییم.

همچنین ما می توانیم در زمان ادغام سایر کانال هار را حذف کرده و عکس رنگی را تنها با یک کانال رنگی نمایش دهیم. یکی از راه های انجام این عملیات استفاده از ماتریس `zeros_like` می باشد. همچنین در هنگام انجام این عملیات ذکر این نکته

بسیار ضروری است که هنگامی که ما می خواهیم یک ماتریس را به عنوان عکس ایجاد نماییم نوع و جنس ان حتما باید از نوع `uint8` باشد.

پس از عملیات جداسازی کانال های تصویر ما می توانیم با اضافه کردن مقادیر دلخواه میزان هر کانال را تغییر دهیم.

آشنایی بیشتر با `uint8`:

در دنیای برنامه‌نویسی، `uint8` یک نوع داده است که به طور خاص برای ذخیره اعداد صحیح بدون علامت 8 بیتی استفاده می‌شود. این نوع داده در زبان‌های مختلف برنامه‌نویسی مانندC++ ، جاوا و پایتون وجود دارد و کاربردهای گوناگونی در توسعه نرم‌افزار و سخت‌افزار دارد.

ویژگی‌های uint8:

- **بدون علامت** uint8 **:**اعداد منفی را شامل نمی‌شود و فقط اعداد صحیح مثبت و صفر را در خود جای می‌دهد.
- **8بیتی :**این نوع داده از 8 بیت حافظه برای ذخیره عدد استفاده می‌کند. هر بیت می‌تواند مقدار 0 یا 1 داشته باشد.
- **محدوده مقادیر :**با توجه به 8 بیتی بودن uint8 و بدون علامت بودن آن، این نوع داده می‌تواند اعدادی از 0 تا 255 (2 به توان 8 منهای 1) را در خود ذخیره کند.

کاربردهای uint8:

- **ذخیره داده‌های کوچک :**از uint8 می‌توان برای ذخیره داده‌هایی که مقدار کمی دارند، مانند سن افراد، تعداد عناصر یا شاخص‌ها در آرایه‌ها استفاده کرد.
- **پردازش تصویر :**در پردازش تصویر، هر پیکسل معمولاً با استفاده از سه مقدار uint8 برای رنگ‌های قرمز، سبز و آبی (RGB)نمایش داده می‌شود.
- **ارتباطات :**در پروتکل‌های ارتباطی، `uint8`برای ارسال داده‌های کنترلی یا داده‌های کوچک مانند کد وضعیت استفاده می‌شود.
- **برنامه‌نویسی میکروکنترلرها :**در برنامه‌نویسی میکروکنترلرها، به دلیل محدودیت حافظه، استفاده از uint8 برای صرفه‌جویی در حافظه و افزایش سرعت پردازش رایج است.

مزایای استفاده از uint8:

- **صرفه‌جویی در حافظه :**با توجه به 8 بیتی بودنuint8 ، این نوع داده نسبت به انواع داده‌های بزرگتر مانند int یا long intفضای کمتری در حافظه اشغال می‌کند.
- **سرعت پردازش :**پردازش اعداد 8 بیتی معمولاً سریع‌تر از پردازش اعداد بزرگتر است.
- **مناسب برای داده‌های کوچک** uint8 **:**برای ذخیره داده‌هایی که مقدار کمی دارند، بسیار مناسب است.

در پایتون، نوع داده `uint8`به طور مستقیم وجود ندارد. با این حال، می‌توان با استفاده از کتابخانه NumPy این نوع داده را شبیه‌سازی کرد.

پردازش تصویر با NumPy و uint8 در پایتون

همانطور که اشاره شد، در پردازش تصویر، تصاویر به صورت مجموعه‌ای از پیکسل‌ها نمایش داده می‌شوند. هر پیکسل، کوچکترین واحد تصویر است و رنگ و شدت آن را مشخص می‌کند. برای نمایش رنگ هر پیکسل، معمولاً از مدل رنگی RGB استفاده می‌شود که شامل سه رنگ اصلی قرمز(Red) ، سبز (Green) و آبی (Blue) است.

در پایتون، برای کار با تصاویر و انجام عملیات پردازش تصویر، از کتابخانه‌های مختلفی مانندOpenCV ، Pillowو-scikit imageاستفاده می‌شود. این کتابخانه‌ها از آرایه‌های NumPy با نوع داده `uint8`برای نمایش و ذخیره تصاویر استفاده می‌کنند.

نحوه نمایش تصویر با آرایه NumPy وuint8

هر تصویر دیجیتال می‌تواند به عنوان یک آرایه سه بعدی NumPy نمایش داده شود. ابعاد این آرایه به ترتیب شامل ارتفاع، عرض و تعداد کانال‌های رنگی تصویر است. در مدل رنگیRGB ، تعداد کانال‌ها برابر با 3 است که هر کانال به ترتیب رنگ‌های قرمز، سبز و آبی را نشان می‌دهد.

برای مثال، یک تصویر با ابعاد 200 *300پیکسل و فرمت RGB به صورت یک آرایه NumPy با ابعاد 3*200*300نمایش داده می‌شود. هر عنصر این آرایه، مقدار رنگ مربوط به یک پیکسل در تصویر را نشان می‌دهد. از آنجا که هر رنگ (قرمز، سبز یا آبی) با یک بایت (8 بیت) نمایش داده می‌شود، نوع داده این آرایه `uint8`است.

چرا مقادیر رنگ در مدل RGB معمولاً بین 0 تا 255 هستند و چرا uint8 برای نمایش آنها مناسب است؟

در مدل رنگیRGB ، هر رنگ با سه مقدار عددی نمایش داده می‌شود که به ترتیب نشان دهنده میزان رنگ قرمز، سبز و آبی در آن رنگ هستند. این مقادیر معمولاً بین 0 تا 255 هستند و دلیل آن به نحوه نمایش و ذخیره رنگ‌ها در سیستم‌های دیجیتال برمی‌گردد.

چرا 0 تا 255؟

- **8بیت :**در سیستم‌های کامپیوتری، داده‌ها به صورت بیت‌ها ذخیره می‌شوند. هر بیت می‌تواند مقدار 0 یا 1 داشته باشد. برای نمایش هر یک از رنگ‌های قرمز، سبز و آبی، معمولاً از 8 بیت استفاده می‌شود. 8 بیت می‌تواند 2 به توان 8 (2^8) حالت مختلف را نمایش دهد که برابر با 256 حالت است. به همین دلیل، مقادیر رنگ‌ها بین 0 تا 255 (شامل 0 و 255) در نظر گرفته می‌شوند.
- **استاندارد :**استفاده از محدوده 0 تا 255 برای رنگ‌ها به یک استاندارد تبدیل شده است و در بسیاری از نرم‌افزارها و دستگاه‌های دیجیتال از آن استفاده می‌شود.

چرا؟uint8

- **uint8:**نوع داده `uint8`در زبان‌های برنامه‌نویسی مانند C++ و پایتون با استفاده از کتابخانه NumPy برای ذخیره اعداد صحیح بدون علامت 8 بیتی استفاده می‌شود. این نوع داده دقیقاً برای نمایش مقادیر رنگ در مدل RGB مناسب است، زیرا می‌تواند اعدادی بین 0 تا 255 را در خود جای دهد.
- **بازدهی :**استفاده از `uint8`به جای انواع داده‌های بزرگتر مانند `int`یا `float`باعث صرفه‌جویی در حافظه و افزایش سرعت پردازش می‌شود. این امر به ویژه در پردازش تصویر که با حجم زیادی از داده‌ها سر و کار داریم، اهمیت زیادی دارد.

مثال

در نظر بگیرید که می‌خواهیم رنگ قرمز خالص را نمایش دهیم. در مدلRGB ، رنگ قرمز خالص به صورت (255، 0، 0) نمایش داده می‌شود. این بدان معناست که مقدار رنگ قرمز برابر با 255 و مقادیر رنگ‌های سبز و آبی برابر با 0 هستند. هر یک از این مقادیر می‌تواند به راحتی در یک متغیر از نوع `uint8`ذخیره شود.

سرازیر شدن حافظه (Memory Overflow) در uint8

همانطور که می‌دانیم، uint8 یک نوع داده 8 بیتی بدون علامت است که می‌تواند اعدادی بین 0 تا 255 را در خود ذخیره کند. اما اگر بخواهیم عددی بزرگتر از 255 یا کوچکتر از 0 را در یک متغیر uint8 قرار دهیم، چه اتفاقی می‌افتد؟ در این حالت، با پدیده‌ای به نام "سرازیر شدن حافظه" یا "سرریز شدن (Overflow) "مواجه می‌شویم.

سرازیر شدن حافظه در uint8

وقتی یک عدد خارج از محدوده 0 تا 255 را به یک متغیر uint8 اختصاص می‌دهیم، مقدار ذخیره شده در متغیر، باقیمانده تقسیم آن عدد بر 256 خواهد بود. به عبارت دیگر، عدد مورد نظر "به دور 256 می‌پیچد" و مقدار باقیمانده پس از تقسیم بر 256 در متغیر ذخیره می‌شود. به این عمل، "modulo operation"یا "عملیات پیمانه‌ای" نیز گفته می‌شود.

چرا سرازیر شدن حافظه رخ می‌دهد؟

این رفتار به نحوه نمایش اعداد در حافظه کامپیوتر مربوط می‌شود. همانطور که گفته شد، uint8 از 8 بیت برای ذخیره اعداد استفاده می‌کند. وقتی عددی بزرگتر از 255 را در این 8 بیت قرار می‌دهیم، بیت‌های اضافی حذف می‌شوند و فقط 8 بیت پایین‌تر باقی می‌مانند. این امر باعث می‌شود که عدد به صورت پیمانه‌ای کاهش یابد.

کاربردها و عواقب سرازیر شدن حافظه

سرازیر شدن حافظه می‌تواند در برخی موارد مفید باشد. برای مثال، در برخی از الگوریتم‌های رمزنگاری یا محاسبات ریاضی، از این ویژگی برای ایجاد چرخه‌های اعداد استفاده می‌شود.

با این حال، در بسیاری از موارد، سرازیر شدن حافظه می‌تواند منجر به خطا در برنامه شود. برای مثال، اگر در محاسبات مربوط به پردازش تصویر یا کنترل سخت‌افزار، از uint8 استفاده کنیم و به طور ناخواسته با اعدادی خارج از محدوده 0 تا 255 مواجه شویم، ممکن است نتایج نادرست یا غیرمنتظره‌ای حاصل شود.

نحوه جلوگیری از سرازیر شدن حافظه

برای جلوگیری از سرازیر شدن حافظه، باید به محدوده اعداد قابل نمایش توسط هر نوع داده توجه داشته باشیم و در صورت لزوم از انواع داده‌های بزرگتر استفاده کنیم. همچنین، می‌توانیم از تکنیک‌های برنامه‌نویسی مانند بررسی محدوده‌ها و مدیریت خطا برای جلوگیری از بروز این مشکل استفاده کنیم.

- مدرس محمد کهکشانی (مدرس رسمی دانشگاه هاروارد)