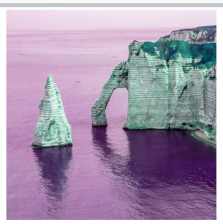
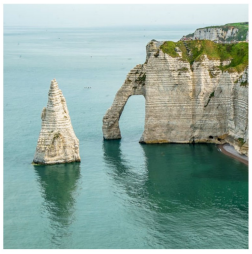
# TP 5- Proposition de solutions

**Solution 1** 1. Les premières instructions affichent un carré blanc ayant 100 pixels de côté.
Les instructions suivantes affichent un carré rose : le vert est enlevé, le rouge et le bleu sont dans les mêmes proportions.
2. La fonction `transfo` permute les couleurs de base : le rouge devient vert, le vert devient bleu et le bleu devient rouge.

```
>>> transfo('Etretat.png')
```



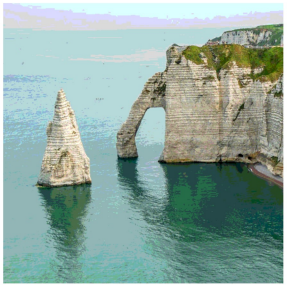
**Solution 2** Niveaux de gris




```
def gris(f_in,f_out):
    f=plt.imread(f_in)
    l,c,d=np.shape(f)
    g=np.zeros((l,c,3))
    for i in range(l):
        for j in range(c):
            for k in range(3):
                g[i,j,k]=0.3*f[i,j,0]
                 +0.59*f[i,j,1]+0.11*f[i,j,2]
    plt.imshow(g)
    plt.show()
    plt.imsave(f_out,g)
```

**Solution 3** Posterisation d'un image : réduction du nombre de nuances

```
def posterise(f_in,f_out,n):
    f=plt.imread(f_in)
    l,c,d=np.shape(f)
    g=np.zeros((l,c,3))
    T=lambda x:round(x*(n-1))/(n-1)
    for i in range(l):
        for j in range(c):
            for k in range(3):
                g[i,j,k]=T(f[i,j,k])
    plt.imshow(g)
    plt.show()
    plt.imsave(f_out,g)
```

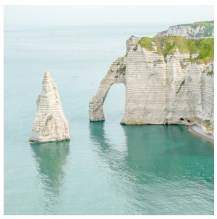



**Solution 4** Éclaircir une image
Pour éclaircir une image, cela revient à diminuer la quantité de couleur, c'est-à-dire augmenter la valeur du nombre entre 0 et 1.
Un simple décalage (avec un maximum pour éviter les quantités négatives) génère une perte de contraste.
Une idée est de considérer une fonction continue de $[0,1]$ dans $[0,1]$, croissante telle que :

$$\begin{cases} h(0)=0 \\ h(1)=1 \\ \forall x\in[0,1], \quad h(x)\geq x \end{cases}$$

Ici, nous choisissons la racine carrée.

```
def eclaircir(f_in,f_out):
    h=lambda x:np.sqrt(x)
    f=plt.imread(f_in)
    t=np.shape(f)
    g=np.zeros((t[0],t[1],3))
    for i in range(t[0]):
        for j in range(t[1]):
            for k in range(3):
                g[i,j,k]=h(f[i,j,k])
    plt.imshow(g)
    plt.show()
    plt.imsave(f_out,g)
```

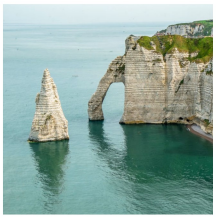



On peut aussi travailler avec un fonction vectorisée :

```
def eclaircir2(f_in,f_out):
    h=lambda x:np.sqrt(x)
    f=plt.imread(f_in)
    g=h(f)
    plt.imshow(g[:,:,:3])
    plt.imsave(f_out,g[:,:,:3])
```

**Solution 5** Flouter une image

```
def flouter(f_in,f_out,d):
    f=plt.imread(f_in)
    t=np.shape(f)
    g=1*f
    for i in range(d,t[0]-d):
        for j in range(d,t[1]-d):
            s=np.zeros(t[2])
            for k in range(i-d,i+d+1):
                for l in range(j-d,j+d+1):
                    s=s+f[k,l]
            g[i,j,:]=s/(2*d+1)**2
    plt.imshow(g)
    plt.show()
    plt.imsave(f_out,g)
```



Autre version :

```
def flouter(f_in,f_out,d):
    f=plt.imread(f_in)
    t=np.shape(f)
    g=1*f
    for i in range(d,t[0]-d):
        for j in range(d,t[1]-d):
            for k in range(3):
                g[i,j,k]=np.sum(f[i-d:i+d+1,j-d:j+d+1,k])/(2*d+1)**2
    plt.imsave(f_out,g)
```

**Solution 6** Redimensionner une image

La solution proposée consiste à définir un pixel de l'image transformée par le pixel associé de l'image de départ. On utilise `round( )` pour trouver l'indice correspondant.

**Redimensionner**

```
def redim(ligne,f_in,f_out):
    f=plt.imread(f_in)
    t=np.shape(f)
    c=int(t[1]*ligne/t[0])
    g=np.zeros((ligne,c,t[2]))
    for I in range(ligne):
        for J in range(c):
            i=round(I/(ligne-1)*(t[0]-1))
            j=round(J/(c-1)*(t[1]-1))
            g[I,J]=f[i,j]
    plt.imshow(g)
    plt.show()
    plt.imsave(f_out,g)
```