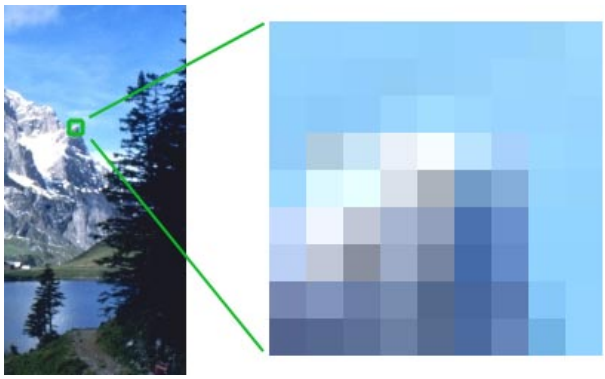
Bitmap conversion

A *bit map picture* is exactly what the name makes one suspect: a sequence of bits (0 or 1) that together represent a digital photo. The picture consists of a matrix (rectangle grid) of individual picture elements (or *pixels*) that each have their own colour. Lets us look at a typical bit map picture to explain this principle:



On the left you can see a photo and on the right you can see a 250% enlargement of one of the mountains. As you can see in the enlargement, a bit map picture consists of hundreds of rows and columns of small elements that each have their own colour. One of those elements is called a *pixel* (which is the short form of *picture element*). The human eye is not capable of seeing an individual pixel, which is the reason why we cannot see a picture with flowing gradations. The number of pixels that is necessary to get a realistic looking picture, depends largely on the way the picture is going to be used.

The colour of a pixel in a bit map picture can be presented in different ways. For this assignment, we distinguish two categories:

- **black-and-white pictures**: the colour of a pixel is a greyscale that is represented as an integer between 0 (black) and 255 (white) or as a floating point number between 0.0 (black) and 1.0 (white).
- **colour picture**: the colour of a pixel is represented by a list of three numbers that represent a red, a green and a blue component. This is called the *RGB-representation* of the colour. By mixing different amounts of these components, a colour is obtained (think about mixing coloured paint). Every component is represented either by an integer between 0 and 255, or by a floating point number between 0.0 and 1.0. The extra values respectively represent 0% and 100% colour for this component.

Assignment

We represent a bit map as a list of lists, where the inner lists always represent the colours of a sequence of pixels from a row of the picture. Every colour is represented by an integer or a floating point (black-and-whit pictures) or a list of three integers or floats (colour pictures). In this assignment, we ask you to do image manipulations, by converting every colour of the bit map to another colour. You work as follows:

- Write a function color2gray that converts the RGB representation (r, g, b) of a colour (with r, g and b integers or floats) to a number $x \in [0, 1]$ that represents a greyscale, base on the following formula $$ x = 0.299r + 0.587g + 0.114b $$ If the colour

components are represented by integers, these first have to be divided by 255 before applying the formula. A list of three integers should be given to the function. The function should print the value of x as a result. With this function, a colour image can be converted to a black-and-white picture.

- Write a function invert, that converts a colour to its invert colour as follows:
  - $x \rightarrow 255 - x$ if the colour is represented by $x \in \mathbb{N}$
  - $x \rightarrow 1 - x$ if the colour is represented by $x \in \mathbb{R}$
  - $[r, g, b] \rightarrow [255 -r, 255 - g, 255 -b]$ if $r, g, b \in \mathbb{N}$
  - $[r, g, b] \rightarrow [1 -r, 1 - g, 1 -b]$ if $r, g, b \in \mathbb{R}$

  As you can see, you should give either a value x or a list $[r, g, b]$ to this function as an argument, and the funcion should print the corresponding inverse value as a result.
- Write a function convertBitmap in which all colours of a bit map can be converted in another colour. To the compulsory parameter bitmap a list of lists should be given that represents a bit map as described above. The function should print a new list of lists that represents the bit map that is obtained by converting the values of the bit map given. The standard conversion is done by squaring all colours of the given bit map:
  - $x \rightarrow (x / 255)^2$ if the colour is represented by $x \in \mathbb{N}$
  - $x \rightarrow x^2$ if the colour is represented by $x \in \mathbb{R}$
  - $[r, g, b] \rightarrow [(r / 255)^2, g / 255)^2, b / 255)^2]$ if $r, g, b \in \mathbb{N}$
  - $[r, g, b] \rightarrow [r^2, g^2, b^2]$ if $r, g, b \in \mathbb{R}$

  The function convertBitmap has a second, optional, parameter converter, to which a function can be given that converts a given colour (that is given to the function as a parameter) to another colour (that is printed by the conversion function). If a conversion function is given to the parameter converter, this function is applied to all pixels of the given picture, to determine the new picture.

Preparation

To determine whether a given object has a certain type of specifications, you can of course use the built-in function type(o), that prints the type of specification of the object o. However, it is better to use the built-in function isinstance(o, t). This function prints a Boolean value that indicates whether the object o belongs to type t, or not.

```
>>> type(3) == int
True
>>> isinstance(3.14, int)
False
>>> isinstance(3.14, float)
True
>>> isinstance([1, 2, 3], list)
True
```

Example

```
>>> color2gray([238, 172, 98])
0.7188156862745098
>>> color2gray([0.7, 0.2, 0.6])
0.3951

>>> invert(168)
87
>>> invert(0.4)
```

```
0.6
>>> invert([238, 172, 98])
[17, 83, 157]
>>> invert([0.7, 0.2, 0.6])
[0.3, 0.8, 0.4]

>>> matrix = [[234, 124, 28], [36, 45, 179]]
>>> convertBitmap(bitmap=matrix, converter=invert)
[[21, 131, 227], [219, 210, 76]]

>>> matrix = [[[0.2, 0.4, 0.6], [0.5, 0.9, 0.0]], [[0.5, 0.7, 0.1], [0.4, 0.3, 1.0]]]
>>> convertBitmap(matrix)
[[[0.04, 0.16, 0.36], [0.25, 0.81, 0.0]], [[0.25, 0.49, 0.01], [0.16, 0.09, 1.0]]]
>>> convertBitmap(bitmap=matrix, converter=color2gray)
[[0.363, 0.6778], [0.5718, 0.4097]]
>>> convertBitmap(bitmap=matrix, converter=invert)
[[[0.8, 0.6, 0.4], [0.5, 0.1, 1.0]], [[0.5, 0.3, 0.9], [0.6, 0.7, 0.0]]]
```

Een *bitmap-afbeelding* is precies wat zijn naam reeds doet vermoeden: een reeks bits (0 of 1) die samen een digitale foto voorstellen. De afbeelding bestaat uit een matrix (rechthoekig rooster) van individuele beeldpunten (of *pixels*) die elk hun eigen kleur hebben. Laat ons een kijken naar een typische bitmap-afbeelding om het principe uit te leggen:

Links zie je een foto, en rechts zie je een 250% uitvergroting van de top van één van de bergen. Zoals je in deze uitvergroting kunt zien bestaat een bitmap-afbeeling uit honderden rijen en kolommen van kleine elementen die elk een eigen kleur hebben. Eén zo'n element wordt een *pixel* genoemd (afkorting voor *picture element*). Het menselijk oog is niet in staat om elke individuele pixel te zien, waardoor we een afbeelding met vloeiende gradaties waarnemen. Het aantal pixels dat nodig is om een realistisch uitziende afbeelding te krijgen, hangt in sterke mate af van de manier waarop de afbeelding zal gebruikt worden.

De kleur van een pixel uit een bitmap-afbeelding kan op verschillende manieren voorgesteld worden. Voor deze opgave onderscheiden we de volgende twee categoriën:

- **zwart-wit afbeeldingen**: de kleur van een pixel is een grijswaarde die wordt voorgesteld als een integer tussen 0 (zwart) en 255 (wit) of als een floating point getal tussen 0.0 (zwart) en 1.0 (wit).
- **kleurafbeeldingen**: de kleur van een pixel wordt voorgesteld door een lijst van drie getallen die een rood-, groen- en blauwcomponent voorstellen. Dit wordt de *RGB-voorstelling* van de kleur genoemd. Door verschillende hoeveelheden van deze componenten te vermengen bekomt men een kleur (denk aan het vermengen van gekleurde verf). Hierbij wordt elke component ofwel voorgesteld door een integer tussen 0 en 255, of door een floating point getal tussen 0.0 en 1.0. De extra waarden stellen in beide

gevallen 0% kleur voor deze component en 100% kleur voor deze component voor.

Opgave

We stellen een bitmap voor als een lijst van lijsten, waarbij de binnenste lijsten telkens de kleuren van een reeks pixels uit een rij van de foto voorstellen. Elke kleur wordt voorgesteld door een integer of floating point (zwart-wit afbeeldingen) of een lijst van drie integers of floats (kleurafbeeldingen). In deze opgave wordt gevraagd om beeldmanipulaties uit te voeren, door elke kleur van een bitmap om te zetten in een nieuwe kleur. Hiervoor ga je als volgt te werk:

- Schrijf een functie kleur2grijs die de RGB-voorstelling (r, g, b) van een kleur (met r, g en b integers of floats) omzet naar een getal $x \in [0, 1]$ die een grijswaarde voorstelt, op basis van de volgende formule $$ x = 0.299r + 0.587g + 0.114b $$ Indien de kleurcomponenten voorgesteld worden door integers, dan moeten deze eerst gedeeld worden door 255 vooraleer de formule toe te passen. Aan de functie moet dus een lijst van drie integers of floats doorgegeven worden. De functie moet de waarde van x als resultaat teruggeven. Met deze functie kan een kleurafbeelding dus omgezet worden naar een zwart-wit afbeelding.
- Schrijf een functie inverteer, die een kleur op de volgende manier omzet naar zijn omgekeerde kleur:
  - $x \rightarrow 255 - x$ als de kleur wordt voorgesteld door $x \in \mathbb{N}$
  - $x \rightarrow 1 - x$ als de kleur wordt voorgesteld door $x \in \mathbb{R}$
  - $[r, g, b] \rightarrow [255 -r, 255 - g, 255 -b]$ als $r, g, b \in \mathbb{N}$
  - $[r, g, b] \rightarrow [1 -r, 1 - g, 1 -b]$ als $r, g, b \in \mathbb{R}$

  Aan deze functie moet dus een waarde x of een lijst $[r, g, b]$ als argument doorgegeven worden, en de functie moet de corresponderende inverse waarde als resultaat teruggeven.
- Schrijf een functie converteerBitmap waarin alle kleuren van een bitmap kunnen omgezet worden in een andere kleur. Aan de verplichte parameter bitmap moet een lijst van lijsten doorgegeven worden die een bitmap voorstelt zoals hierboven beschreven. De functie moet een nieuwe lijst van lijsten als resultaat teruggeven, die de bitmap voorstelt die men bekomt door de waarden van de gegeven bitmap te converteren. De standaard convertie bestaat erin om alle kleuren van de gegeven bitmap te kwadrateren:
  - $x \rightarrow (x / 255)^2$ als de kleur wordt voorgesteld door $x \in \mathbb{N}$
  - $x \rightarrow x^2$ als de kleur wordt voorgesteld door $x \in \mathbb{R}$
  - $[r, g, b] \rightarrow [(r / 255)^2, g / 255)^2, b / 255)^2]$ als $r, g, b \in \mathbb{N}$
  - $[r, g, b] \rightarrow [r^2, g^2, b^2]$ als $r, g, b \in \mathbb{R}$

  De functie converteerBitmap heeft nog een tweede optionele parameter converter, waaraan een functie kan doorgegeven worden die een gegeven kleur (die als parameter aan de convertiefunctie doorgegeven wordt) omzet in een nieuwe kleur (die door de convertiefunctie als waarde teruggegeven wordt). Indien een convertiefunctie wordt doorgegeven aan de parameter converter, dan is het deze functie die wordt toegepast op alle pixels van de gegeven afbeelding, om de nieuwe afbeelding te bepalen.

Voorbereiding

Om na te gaan of een gegeven object een bepaald gegevenstype heeft, kan je natuurlijk gebruik maken van de ingebouwde functie type(o) die het gegevenstype van het object o teruggeeft. Het is echter beter om hiervoor de ingebouwde functie isinstance(o, t) te gebruiken. Deze functie geeft

een Booleaanse waarde terug die aangeeft of het object `o` al dan niet behoort tot het type `t`.

```
>>> type(3) == int
True
>>> isinstance(3.14, int)
False
>>> isinstance(3.14, float)
True
>>> isinstance([1, 2, 3], list)
True
```

Voorbeeld

```
>>> kleur2grijs([238, 172, 98])
0.7188156862745098
>>> kleur2grijs([0.7, 0.2, 0.6])
0.3951

>>> inverteer(168)
87
>>> inverteer(0.4)
0.6
>>> inverteer([238, 172, 98])
[17, 83, 157]
>>> inverteer([0.7, 0.2, 0.6])
[0.3, 0.8, 0.4]

>>> matrix = [[234, 124, 28], [36, 45, 179]]
>>> converteerBitmap(bitmap=matrix, converter=inverteer)
[[21, 131, 227], [219, 210, 76]]

>>> matrix = [[[0.2, 0.4, 0.6], [0.5, 0.9, 0.0]], [[0.5, 0.7, 0.1], [0.4, 0.3, 1.0]]]
>>> converteerBitmap(matrix)
[[[0.04, 0.16, 0.36], [0.25, 0.81, 0.0]], [[0.25, 0.49, 0.01], [0.16, 0.09, 1.0]]]
>>> converteerBitmap(bitmap=matrix, converter=kleur2grijs)
[[0.363, 0.6778], [0.5718, 0.4097]]
>>> converteerBitmap(bitmap=matrix, converter=inverteer)
[[[0.8, 0.6, 0.4], [0.5, 0.1, 1.0]], [[0.5, 0.3, 0.9], [0.6, 0.7, 0.0]]]
```