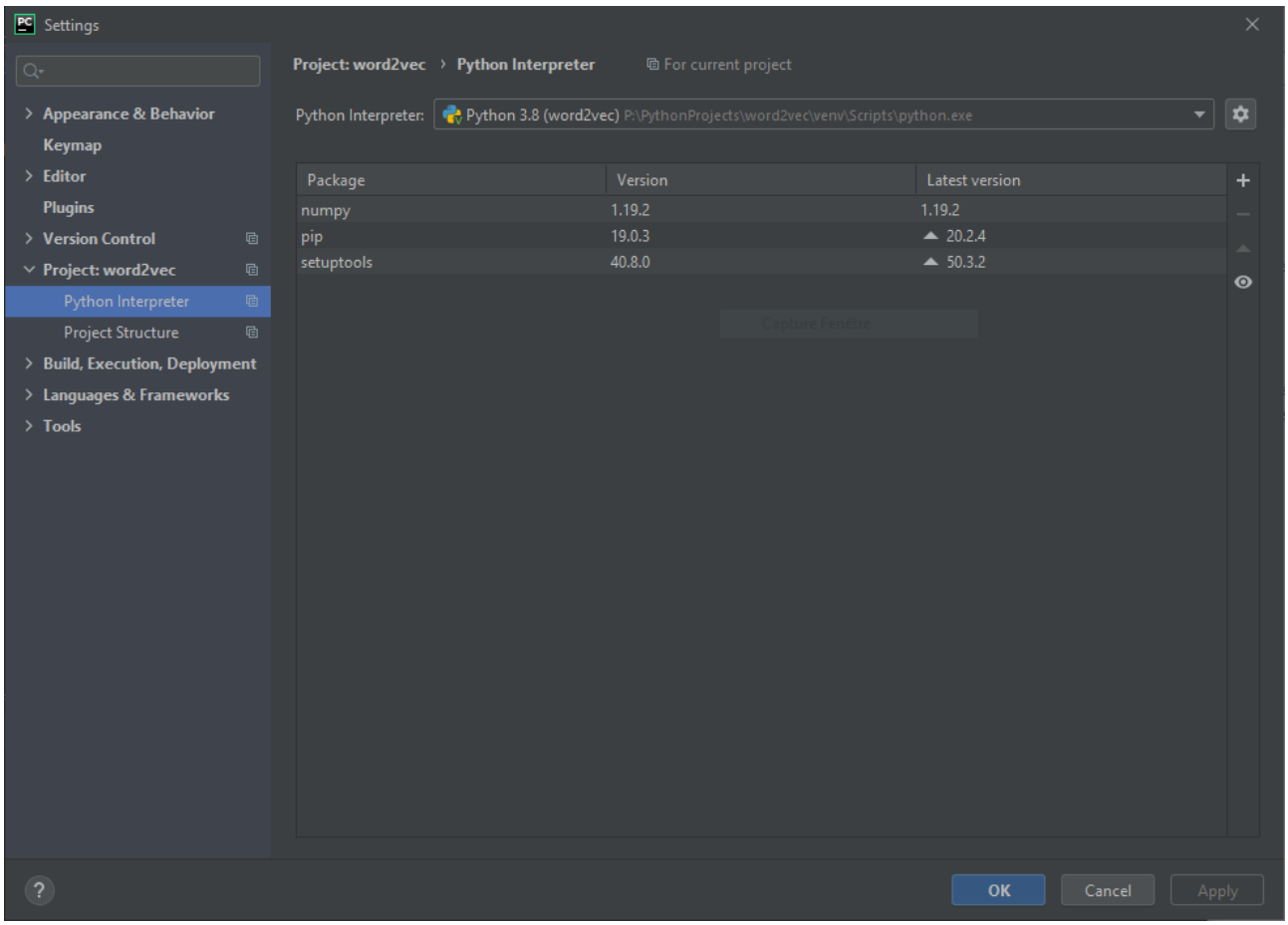
Connaissance de numpy

Numpy est un module central pour Python en particulier pour le calcul scientifique et le traitement des données de grande taille.
Le site web « officiel » de numpy est accessible à cette adresse : https://numpy.org/

Si vous utilisez python, numpy s'il n'est pas déjà présent peut être installé par :
pip3 install python. Attention, il semble qu'il y ait un bug entre la dernière version de numpy et windows, il faut installer une version antérieure :
pip2 install numpy==1.19.3

Si vous utilisez conda, numpy est installable par la commande
conda install python

Si comme moi, vous utilisez un environnement de développement intégré comme PyCharm (https://www.jetbrains.com/fr-fr/pycharm/) et que vous utilisez un environnement virtuel venv, vous devez aller dans le menu settings → Python Interpreter et rajouter numpy dans la liste :



Si vous utilisez jupyter notebook, le principe est le mêmeN

Pour vérifier que numpy fonctionne, la première étape est d'importer le module :
import numpy as np

Par convention, on importe le module numpy et on va préfixer toutes les opérations réalisées avec numpy par np. Si vous avez **un message d'erreur,** n'allez pas plus loin, c'est qu'il y a un problème dans l'importation du module.

Voici quelques éléments sur numpy. Numpy apporte le concept fondamental de tableaux et matrices et la manipulation des matrices et de l'algèbre linéaire.

création et manipulation de tableaux

Les fonctions **zeros** ou **ones** créant respectivement un tableau de 0 ou de 1 (il faut préfixer avec le nom du module numpy):

- le 1er argument est la donnée des dimensions du tableau, sous forme d'un n-uplet listant la taille de chaque dimension, par exemple (10) pour un vecteur de 10 éléments, ou (4,4) pour une matrice carrée de dimension 4x4.

- le 2nd argument donne le type des éléments soit comme nom de type (ex: int), soit encore sous forme de texte, comme 'int' ou 'float'.• Quelques exemples (ils sont affichés comme ils le seraient si vous les tapiez, dans la fenêtre Python console de PyCharm ou directement sur l'interpréteur de Python3 :

```
>>> import numpy as np # np sera le raccourci pour nommer le module numpy
>>> a = np.zeros((2,2), dtype = 'int') # an int array initialized to 0 of shape (2,2)
>>> b = np.ones((2,2), dtype = 'float') # a float array initialized to 1. of shape (2,2)

>>>a
array([[0, 0],
       [0, 0]])

>>>b
array([[1., 1.],
       [1., 1.]])
```

- On peut aussi créer un tableau avec la commande array en passant en paramètre une liste pour un vecteur, ou une liste de listes pour une matrice (notez que le type d'élément peut être déduit du contenu des listes) :

```
>>> tab1d = np.array([0,1,2,3,4], dtype = int)
>>> mat2d = np.array([[0,1,2], [3,4,5], [6,7,8]])
>>> mat2d
array([[0, 1, 2],
[3, 4, 5],
[6, 7, 8]])
```

- Les tableaux numpy supportent des opérations « par élément » :

```
>>> # ajoute 2 à chaque élément et le multiplie par 3
>>> tab1d = 3 * (tab1d + 2)

matd2 + 2
array([[ 2,  3,  4],
       [ 5,  6,  7],
       [ 8,  9, 10]])
```

La forme (dimensions) d'un tableau peut se retrouver par l'attribut shape (sans parenthèses), donc le nombre de dimensions, et chacune d'elles:

```
>>> mat2d.shape
(3, 3)
>>> len(mat2d.shape)
2
>>> mat2d.shape[0] # taille dans la dimension 0
3
```

L'accès aux éléments se fait par des indices ou des tranches :

```
>>> mat2d = np.array([[0,1,2], [3,4,5], [6,7,8]])
>>> mat2d
array([[0, 1, 2],
[3, 4, 5],
[6, 7, 8]])
```

```
>>> mat2d[0,0]
0
>>> mat2d[0,1]
1
>>> mat2d[1,0]
3
>>>  matd2[0]  # accès à la première ligne
array([0, 1, 2])

>>> matd2[0, 1:] # accès à al première ligne à partir de la colonne 1
array([1, 2])
```

Exercice :

Écrire une fonction Python utilisant numpy qui vérifie qu'une matrice est une matrice symétrique

- une fonction bien utile linspace qui permet d'obtenir un tableau d'une valeur donnée à une valeur finale avec un nombre donné d'éléments

```
>>>np.linspace(0,1,9)
array([0.   , 0.125, 0.25 , 0.375, 0.5  , 0.625, 0.75 , 0.875, 1.   ])
```

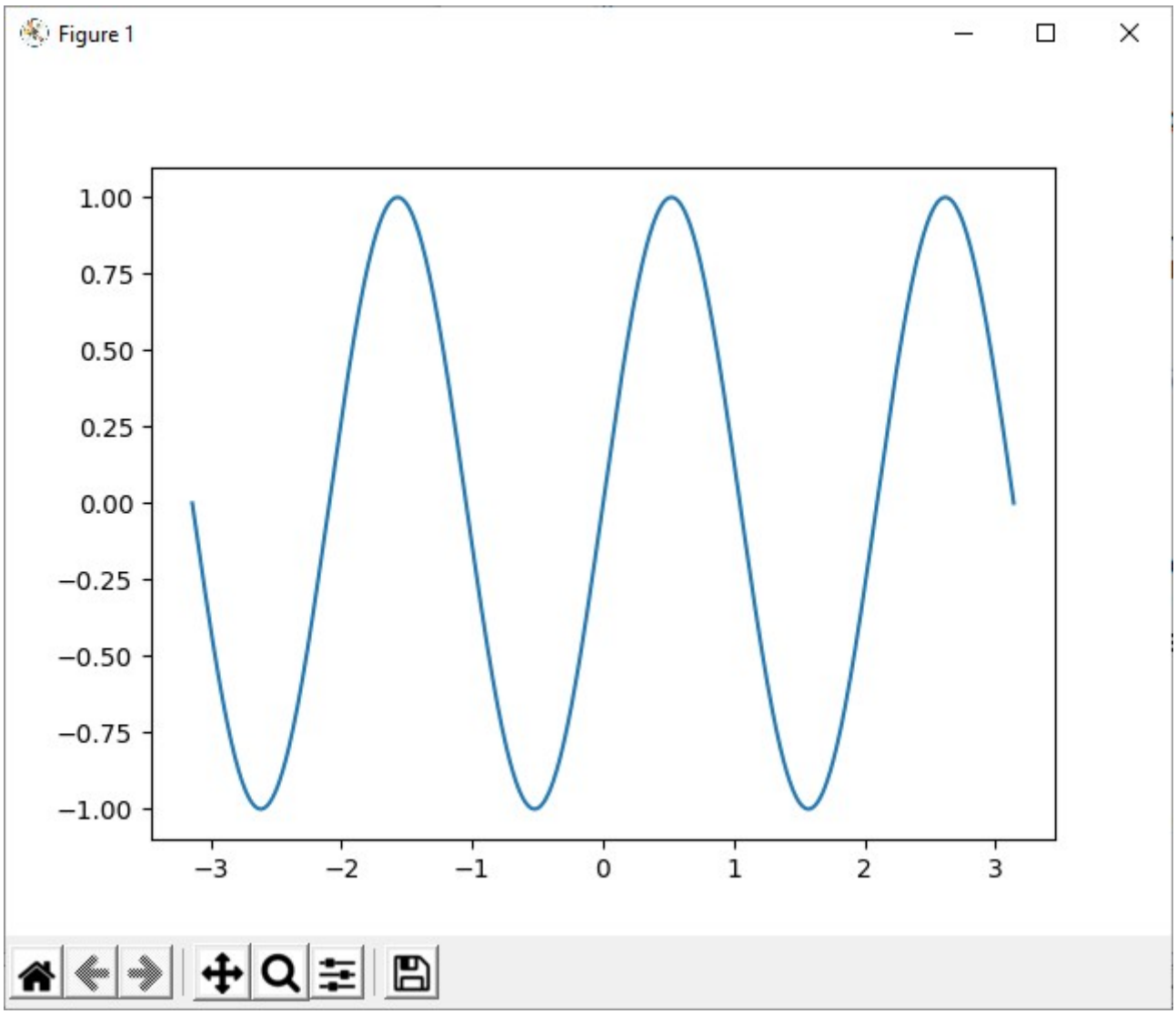
Exercice :

Utiliser les commandes suivantes du module matplotlib (que nous verrons plus tard) qui permet d'afficher des valeurs à partir de deux vecteurs 1d (un en abscisse et un en ordonné) pour afficher la fonction sinus entre $-\pi$ et $+\pi$.

```
>>>import matplotlib.pyplot as plt
...
>>>plt.plot(x,y)
>>> plt.show() f # inutile facultatif en mode interactif
```

Je vous propose dans le cadre de cet apprentissage de numpy de travailler sur la manipulation d'images.
Pour cela, nous avons un module très pratique qui s'appelle pillow (https://pillow.readthedocs.io/en/stable/) très pratique pour manipuler des images.

Par exemple, les quelques lignes suivantes permettent d'afficher une image et de lui faire subir une rotation :




```
>>> from PIL import Image
>>>image = Image.open("c:\\Temp\\falloxbow-1058032.jpg")
>>>image.show()
>>>image90 = image.rotate(90)
>>>image90.show()
```

Note importante et pratique :

Si vous utilisez les jupyter notebook, vous pouvez afficher une image avec la commande **![title](img/fichier.jpg)** en supposant bien sûr que l'image se trouve dans le répertoire img du répertoire courant de votre notebook. Ce n'est pas du code python, c'est du markdown

On transforme ensuite notre image en un tableau de numpy par la commande « magique » array :

```
>>>image_np = np.array(image)
```

On vérifie le type et le format de cette variable :

```
>>>type(image_np)
<class 'numpy.ndarray'>
>>>print(image_np.dtype)
uint8
>>>print(image_np.shape)
(668, 1599, 3)
```

L'image corresponde donc à une matrice à 3 dimensions, les deux premières dimensions correspondant à la hauteur et la largeur de l'image, la 3ème dimension correspondant à 3 valeur (R,G,B) qui sont les composantes rouge, verte et bleu.

---> A vous de tester tout ça sur votre image

Lire ce tutoriel https://www.pythoninformer.com/python-libraries/numpy/index-and-slice/

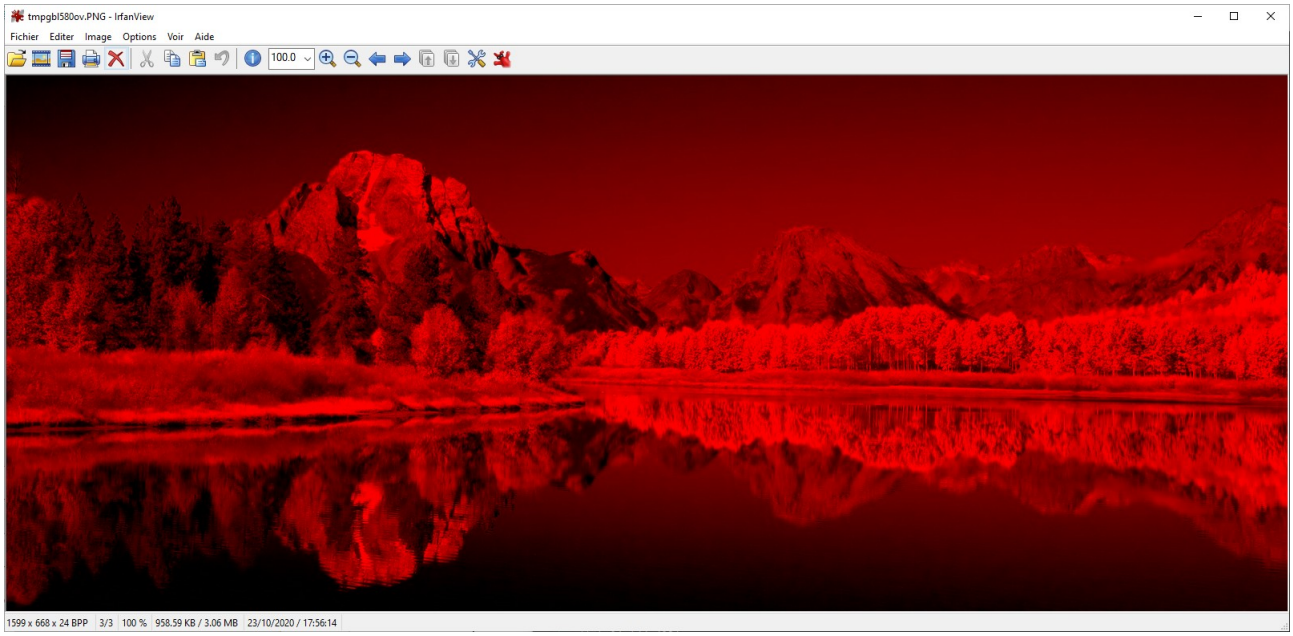
Pour sauvegarder une image, il faut réaliser le processus inverse, transformer la matrice en un fichier image et le sauvegarder :

```
>>>image_save = Image.fromarray(image_np)
>>>image_save('img/mon_fichier.webp')
```

Exercice :

A quoi correspond
```
>>>print(image_np[100,100])
[ 52 132 253]
```
Exercice (manipulation de tranches de numpy) :

Transformez l'image originale, en ne conservant que la composante rouge pour obtenir l'image suivante :



Pour ce faire, vous pourrez utiliser les commandes suivantes :
méthode copy pour copier et garder l'image originale image_np
Passez les composantes verte et bleue à 0 pour ne garder que la composante rouge.
Transformez votre « array » numpy en image avec la méthode `Image.fromarray()`

Exercice (concaténation de tableaux) :

Faîtes de même pour réaliser 3 tableaux correspondant à des images 1 rouge, une bleue et une verte et utilisez la commande numpy concatenate pour concaténer les 3 tableaux en un seul puis l'afficher pour réaliser l'image globale suivante :

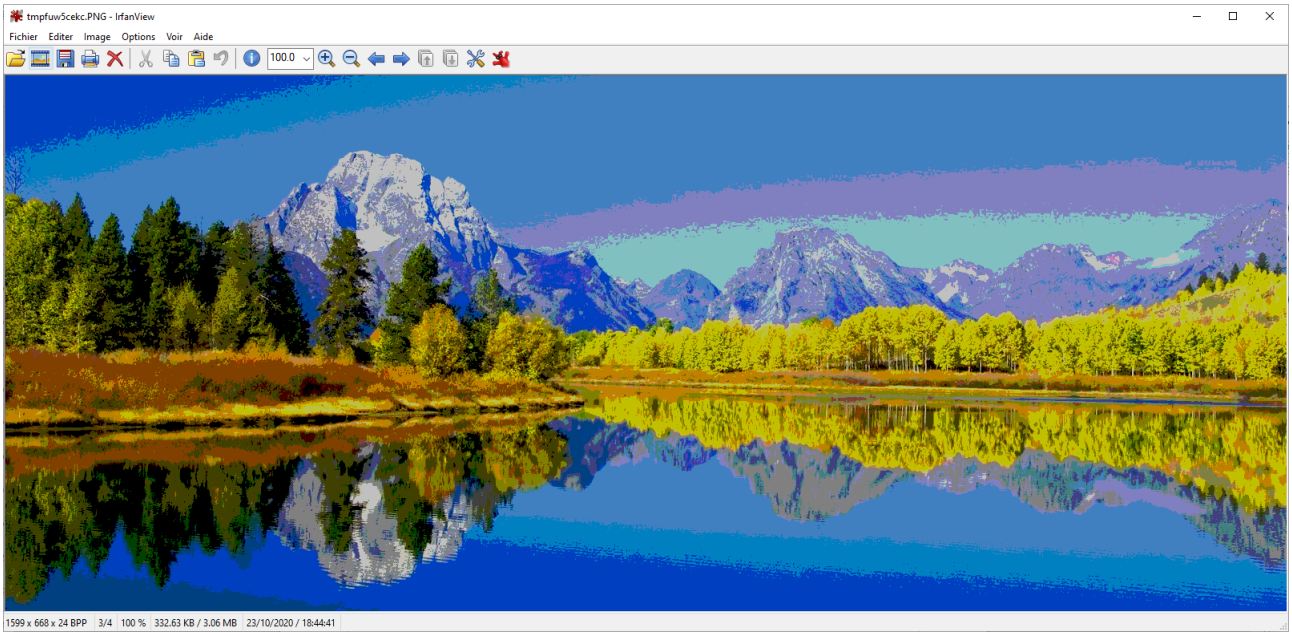
Pour la concaténation, vous pouvez jeter un coup d'œil à cette page web
https://cmdlinetips.com/2018/04/how-to-concatenate-arrays-in-numpy/

Exercice : opération sur la totalité de la matrice

Quand on travaille avec des tableaux numpy les opérations courantes avec des scalaires travaillent sur la totalité du tableau. Nous allons utiliser cette caractéristique afin d'inverser l'image

Le processus est à peu près semblable lorsqu'on souhait diminuer le nombre de couleurs. Il s'agit de grouper les couleurs par groupe par exemple donner aux 10 premières couleurs la valeur 0, les couleurs de 10 à 20 la valeur 10, les couleurs de 20 à 30, la couleur20 etc.

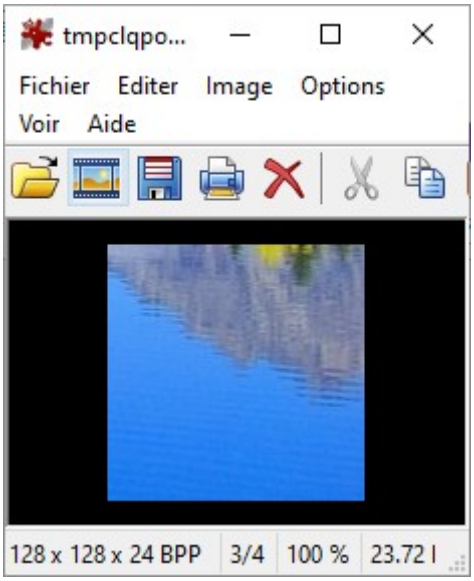
Utilisez la division entière pour réaliser cet exercice pour obtenir le résultat ci-dessous :



Exercice : extraction de tableaux

Le plus simple : en utilisant l'attribut shape, extraire une image de taille 256x256 au centre de

l'image


tmpclqpo...
Fichier Editer Image Options
Voir Aide
128 x 128 x 24 BPP
3/4
100 %
23.72 I