Introduction

Rappel : Ne pas oublier d'exécuter l'import de Numpy ci-dessous!

Informations sur les types de données:

- Documentation Numpy : https://numpy.org/devdocs/user/basics.types.html
- Documentation Numpy : https://numpy.org/doc/stable/reference/generated/numpy.dtype.kind.html
- W3Schools : https://www.w3schools.com/python/numpy/numpy_data_types.asp

Afficher le type de données contenu dans un ndarray

Nous allons voir les types de données que vous pouvez utiliser pour déclarer des ndarray avec Numpy. Numpy trouvant ses fondations dans le langage de programmation C, les types de données que ce module manipule sont similaires à ce langage.

```
matrice = np.array([[1,2], [3,4]])
print(matrice.dtype)
```

```
dtype('int32')
```

Le type des éléments que contient le vecteur précédent est int32. C'est-à-dire des nombres entiers prenant chacun 32 bits d'espace en mémoire.

Changer le type de données

déclarer un ndarray de type uint

Pour changer **le type de données** d'un ndarray, on peut affecter une valeur au paramètre **dtype** de la méthode array provenant du module numpy. Cette valeur doit être une **chaîne de caractères** (type str en Python) ou un objet de type np.type_de_données.
La liste des valeurs possibles (les types de données) est disponible plus haut dans ce notebook (lien vers la documentation).

```
matrice = np.array([[1,2], [3,4]], dtype="uint")
print(matrice.dtype)
```

```
dtype('uint32')
```

Dans l'exemple ci-dessus, nous avons déclaré un **dtype** égal à "**uint32**". "uint32" est un type autorisant les **nombres entiers positifs** prenant **32 bits** en mémoire. Par exemple, si on essaie d"affecter une valeur négative à un de ses éléments, cet élément se verra affecter une valeur délirante.

```
matrice = np.array([[-1,2], [3,4]], dtype=np.uint)
print(matrice)
```

```
[[4294967295          2]
 [         3          4]]
```

Dans l'exemple ci-dessus, on déclare la valeur du premier élément à -1, et le **dtype** est égal à **uint**. Au final, cet élément a pour valeur un nombre immense : 4294967295
Remarque : on a déclaré le type uint en affectant à **dtype** l'objet **np.uint**.
Cela me permet de mettre en avant un message important : **Faîtes très attention aux types de données que vous manipulez!**

Déclarer un ndarray de nombres flottants

Cette fois, déclarons un ndarray de type "**float32**". Cela signifie que l'on désire manipuler des nombres flottants (des nombres à virgule).

```
matrice = np.array([[-1,2], [3,4]], dtype="float32")
print(matrice.dtype)
print(matrice)
```

```
float32
[[-1.  2.]
 [ 3.  4.]]
```

Vous voyez qu'un point a été ajouté à chaque élément.

Déclarer un ndarray de nombres complexes

Un nombre complexe est un nombre qui possède une partie réelle et imaginaire. Numpy est capable de les manipuler.

```
matrice = np.array([[-1,2], [3,4]], dtype=np.complex64)
print(matrice)
```

```
Out[20]:
array([[-1.+0.j,  2.+0.j],
       [ 3.+0.j,  4.+0.j]], dtype=complex64)
```

Ici, le j a été ajouté à chaque élément et représente la partie imaginaire des nombres complexes de la matrice.

Déclarer un ndarray de booléens

```
matrice = np.array([[-1,2], [3,4]], dtype=np.bool_)
print(matrice)
```

```
array([[ True,  True],
       [ True,  True]])
```

Le dtype np.bool est déprécié depuis la version 1.20, il faut à présent utiliser le **dtype = np.bool_**.
Pour ce type, tout ce qui est **différent de zéro sera égal à True**, et les **éléments valant 0 seront égaux à False**.
On va passer le premier élément à 0 pour bien voir le False.

```
matrice = np.array([[0,2], [3,4]], dtype=np.bool_)
print(matrice)
```

```
array([[False,  True],
       [ True,  True]])
```

Déclarer un ndarray de chaines de caractères

```
matrice = np.array([[0,2], [3,4]], dtype=np.str_)
print(matrice)
```

```
array([['0', '2'],
       ['3', '4']], dtype='<U1')
```

Le type np.str est déprécié depuis Numpy 1.20, il faut utiliser le **dtype = np.str_**. On constate que tous les éléments ont été mis entre apostrophes (quotes), ce qui signifie que chaque élément est maintenant une chaîne de caractères.
On peut voir dans notre méthode array() que le dtype est égal à **"<U1"**. Cela signifie que la plus longue chaîne de caractères de notre matrice contient 1 élément. Si nous avions une chaîne de caractères constituée de **3 éléments, alors le dtype serait égal à U3**.
On peut le démontrer avec l'exemple ci-dessous.

```
matrice = np.array([[404,2], [3,4]], dtype=np.str_)
print(matrice)
```

```
array([['404', '2'],
       ['3', '4']], dtype='<U3')
```