

Prise en main de Numpy

- Tableaux
- Attributs
- Types
- Fonction Array

Programmation Python Partie 2 : Prise en main de Numpy

Prise en main de Numpy

Tableaux

Attributs

Types

Fonction Array

Tableaux NumPy (la suite)

```
In [21]: v = np.array([1, 3, 2, 4])  
         print(v)
```

```
[1 3 2 4]
```

```
In [22]: M = np.array([[1, 3], [2, 4]])  
         print(M)
```

```
[[1 3]  
 [2 4]]
```

Prise en main de Numpy

Tableaux

Attributs

Types

Fonction Array

L'attribut `dtype`

- Un *ndarray* peut contenir des nombres entiers, flottants, complexes etc.
- On obtient le type des données via l'attribut `dtype`
- `dtype` est l'abréviation de *data type*

```
In [23]: print(M)  
          print(M.dtype)
```

```
[[1 3]  
 [2 4]]  
int64
```

Prise en main de Numpy

Tableaux

Attributs

Types

Fonction Array

Plus d'attributs

```
In [24]: M.itemsize # nombre d'octets par élément
```

```
Out[24]: 8
```

```
In [25]: M.nbytes # nombre d'octets
```

```
Out[25]: 32
```

```
In [26]: M.nbytes / M.size # égal à itemsize
```

```
Out[26]: 8
```

Prise en main de Numpy

Tableaux

Attributs

Types

Fonction Array

Assignment élément par élément

```
In [27]: print(M)
```

```
[[1 3]
 [2 4]]
```

```
In [28]: M[0,0] = -1
print(M)
```

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

Prise en main de Numpy

Tableaux

Attributs

Types

Fonction Array

Les types doivent être respectés lors d'assignments

```
In [29]: M[0,0] = "hello"
```

```
-----  
-----  
ValueError                                Traceback (most recent call last)  
<ipython-input-29-a09d72434238> in <module>()  
----> 1 M[0,0] = "hello"  
  
ValueError: invalid literal for long() with base 10: 'hello'
```

Attention !

```
In [30]: a = np.array([0, 0, 0])
          print(a.dtype)
          a[0] = 3.2 # perte de précision
          print(a)
```

```
int64
[3 0 0]
```

On peut définir le type de manière explicite en utilisant le mot clé `dtype` en argument:

```
In [31]: a = np.array([0, 0, 0], dtype=np.float64)
          a[0] = 3.2
          print(a)
```

```
[ 3.2  0.  0. ]
```

- Autres options possibles de `dtype`: `int`, `float`, `complex`, `bool`, `object`, etc.
- On peut aussi spécifier la précision en bits: `int32`, `int16`, `float128`, `complex128`, etc.

Prise en main de Numpy

Tableaux

Attributs

Types

Fonction Array

Changement de type

```
In [32]: M = np.array([[ -1, 2], [0, 4]])  
         print(M.dtype)
```

int64

```
In [33]: M2 = M.astype(float)  # conversion en float  
         print(M2)  
         print(M2.dtype)
```

[[-1. 2.]
 [0. 4.]]
float64

```
In [34]: M3 = M.astype(bool)  # conversion en bool  
         print(M3)  
         print(M3.dtype)
```

[[True True]
 [False True]]
bool

Prise en main de Numpy

Tableaux

Attributs

Types

Fonction Array

Utilisation de fonction de génération d'*arrays*

arange "array range"

```
In [35]: x = np.array(range(0, 10, 2)) # à partir d'une liste  
print(x)
```

```
[0 2 4 6 8]
```

```
In [36]: x = np.arange(0, 10, 2) # OK : plus efficace  
print(x)
```

```
[0 2 4 6 8]
```

```
In [37]: x = np.arange(-1, 1, 0.5) # avec flottants  
print(x)
```

```
[-1. -0.5  0.  0.5]
```

Prise en main de Numpy

Tableaux

Attributs

Types

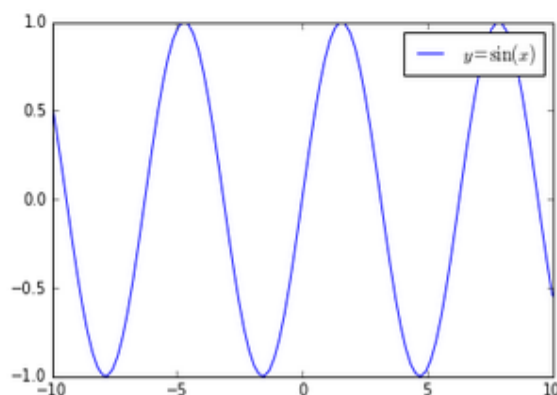
Fonction Array

linspace

```
In [38]: # avec linspace, le début et la fin SONT inclus  
np.linspace(0, 10, 6)
```

```
Out[38]: array([ 0.,  2.,  4.,  6.,  8., 10.] )
```

```
In [39]: x = np.linspace(-10, 10, 100)  
y = np.sin(x)  
plt.plot(x, y, label='$y = \sin(x)$')  
plt.legend()  
plt.show()
```



Prise en main de Numpy

Tableaux

Attributs

Types

Fonction Array