

# X - Python

## Commentaire

#	Insertion de commentaire	# Cette ligne ne sera pas lue par Python
---	--------------------------	--

## I - Structures élémentaires

### I.1 - Types élémentaires

#### Nombres

+	Addition	3.4 + 2 renvoie 5.4
-	Soustraction	3.4 - 2 renvoie 1.4
*	Multiplication	3.4 * 2 renvoie 6.8
/	Division	3.4/2 renvoie 1.7
**	Puissance	3.4**2 renvoie (presque) 11.56

#### Booléens

True, False	Valeurs booléennes vrai, faux	
==	Égal	4 == 2*2 renvoie True
>	Strictement supérieur	4 > 2 renvoie True
<	Strictement inférieur	4 < 2 renvoie False
>=	Supérieur ou égal	4 >= 2 * 2 renvoie True
<=	Inférieur ou égal	4 <= 2 * 2 renvoie True

#### Connecteurs logiques

and	Et logique	(3 == 0) and (4 == 2*2) renvoie False
or	Ou logique	(3 == 0) or (4 == 2*2) renvoie True
not	Non logique	not (3 == 0) renvoie False

## I.2 - Structures de contrôle

Affectation		
=	Affectation  Appel du contenu Écrasement du contenu de	<code>x = 3</code> stocke la valeur 3 dans la variable nommée <code>x</code> . <code>2 * x + 3</code> renvoie 9 <code>x = 25 + 3 * 12</code>
Instruction conditionnelle		
<pre>if c1:     i1 elif c2:     i2 else:     i3</pre>	<code>c1, c2</code> sont des booléens <code>i1, i2, i3</code> sont des instructions Attention aux indentations Attention aux deux-points : <code>elif</code> (sinon mais si) et <code>else</code> (sinon) sont optionnels	<code>x = 20</code> <code>if x &lt; 20:</code> <code>print("Riri")</code> <code>elif x &lt; 50:</code> <code>print("Fifi")</code> <code>else:</code> <code>print("Loulou")</code> Affiche Fifi
Boucle itérative		
<pre>for element in liste:     i1</pre>	Nombre prédéterminé d'itérations <code>i1</code> est une suite d'instructions Attention aux indentations Attention aux deux-points : Liste peut être une liste <code>[3, 12, 1, 4]</code> un intervalle d'entiers <code>range(a, b)</code> un intervalle de réels <code>np.arange(a, b, pas)</code> <code>np.linspace(a, b, nombre)</code>	<code>for i in [3, 12, 1, 4]:</code> <code>print(i)</code> Affiche 3 12 1 4 <code>for i in range(3, 7):</code> <code>print(i)</code> Affiche 3 4 5 6
Boucle conditionnelle		
<pre>while c:     i1</pre>	<code>c</code> est un booléen <code>i1</code> est une suite d'instructions Attention aux indentations Attention aux deux-points : Attention à modifier la condition à chaque passage pour qu'elle devienne fausse	<code>i = 3</code> <code>while i &lt; 48:</code> <code>i = 2 * i</code> <code>print(i)</code> Affiche 6, 12, 24, 48

## Fonctions

<pre>def f(x, y):     i1     return z</pre>	<p><b>x, y</b> sont les paramètres formels  <b>i1</b> est une suite d'instructions  <b>z</b> est la valeur renvoyée          Attention aux deux-points :</p>	<pre>def f(x):     y = x**2 + 1     return 3 * y</pre> <p>Affiche f(3) renvoie 30</p>
---	--	---

## I.3 - Modules

## Importer des modules

<pre>from numpy import * log(2)</pre>	<p>Importe toutes les fonctions de numpy          Appel sans préciser la provenance</p>
<pre>import numpy as np np.log(2)</pre>	<p>Charge le module <b>numpy</b>.          Appel en précisant le module d'appartenance.</p>

## I.4 - Numpy - Calculs numériques

Module pour effectuer des calculs numériques :

```
import numpy as np
```

## Constantes

<b>np.e</b>	Constante e	Vaut environ 2.718
<b>np.pi</b>	Constante $\pi$	Vaut environ 3.14

## Fonctions

<b>np.exp</b>	Exponentielle	<b>np.exp(1)</b> renvoie environ 2.718
<b>np.log</b>	Logarithme népérien	<b>np.log(1)</b> renvoie 0
<b>np.sqrt</b>	Racine carré	<b>np.sqrt(4)</b> renvoie 2.0
<b>np.abs</b>	Valeur absolue	<b>np.abs(-3)</b> renvoie 3
<b>np.floor</b>	Partie entière	<b>np.floor(3.14)</b> renvoie 3.0

## Création de tableaux / matrices

<b>np.array</b>	Crée un tableau à partir de la liste des éléments	<b>np.array([[1, 2, 3], [4, 5, 6]])</b> définit $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$
<b>np.zeros((n, p))</b>	Crée une matrice à <b>n</b> lignes et <b>p</b> colonnes ne contenant que des zéros	<b>np.zeros((2, 3))</b> définit $\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$

<code>np.ones((n, p))</code>	Crée une matrice à <b>n</b> lignes et <b>p</b> colonnes ne contenant que des 1	<code>np.ones((2, 3))</code> définit $\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$
<code>np.eye(n)</code>	Crée la matrice identité d'ordre <b>n</b>	<code>np.eye(3)</code> définit $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$
<code>np.arange(a, b, pas)</code>	Crée un vecteur ligne d'éléments de <b>a</b> (inclus) à <b>b</b> (exclus) espacés de <b>pas</b>	<code>np.arange(1.2, 2, 0.2)</code> définit (1.2 1.4 1.6 1.8)
<code>np.linspace(a, b, nbre)</code>	Crée un vecteur ligne d'éléments régulièrement espacés de <b>a</b> (inclus) à <b>b</b> (inclus) contenant <b>nbre</b> éléments	<code>np.linspace(1.2, 2, 5)</code> définit (1.2 1.4 1.6 1.8 2.)
<b>Manipulation de matrices</b>		
<code>t[i][j]</code>	Accède à la ligne <b>i</b> colonnes <b>j</b> de <b>t</b> Numérotation à partir de 0	<code>t = np.array([[1, 2, 3], [4, 5, 6]])</code> <code>t[1][2]</code> renvoie 6
<code>np.shape</code>	Renvoie le nombre de lignes et le nombre de colonnes	<code>t = np.array([[1, 2, 3], [4, 5, 6]])</code> <code>a, b = np.shape(t)</code> <code>a</code> contient 2, <code>b</code> contient 3
<code>np.reshape</code>	Aplatit puis redimensionne un tableau	<code>a = np.array([[1, 2, 3], [4, 5, 6]])</code> <code>np.reshape(a, (3, 2))</code> renvoie $\begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}$
<b>Opérations sur les matrices</b>		
		<code>a = np.array([[1, 2], [3, 4]])</code> <code>b = np.array([[ -1, 1], [0, 1]])</code>
+	Addition élément par élément	<code>a + b</code> renvoie $\begin{pmatrix} 0 & 3 \\ 3 & 5 \end{pmatrix}$
-	Soustraction élément par élément	<code>a - b</code> renvoie $\begin{pmatrix} 2 & 1 \\ 3 & 3 \end{pmatrix}$
*	Multiplication d'une matrice par un réel	<code>2 * a</code> renvoie $\begin{pmatrix} 2 & 4 \\ 6 & 8 \end{pmatrix}$
<code>np.dot</code>	Produit matriciel	<code>np.dot(a, b)</code> renvoie $\begin{pmatrix} -1 & 3 \\ -3 & 7 \end{pmatrix}$
Les fonctions <code>np.exp</code> , <code>np.sqrt</code> ,... s'effectuent élément par élément		

## Stastitiques

<code>*, /, **</code>	<b>ATTENTION!</b> Opérations élément par élément	<code>b**a</code> renvoie $\begin{pmatrix} -1 & 1 \\ 0 & 1 \end{pmatrix}$
		<code>t = np.array([[1, 2, 3], [4, 5, 6]])</code>
<code>np.sum</code>	Somme des éléments d'un tableau	<code>np.sum(t)</code> renvoie 21 <code>np.sum(t, 0)</code> renvoie $\begin{pmatrix} 5 & 7 & 9 \end{pmatrix}$ <code>np.sum(t, 1)</code> renvoie $\begin{pmatrix} 6 \\ 15 \end{pmatrix}$
<code>np.min</code>	Minimum des éléments d'un tableau	<code>np.min(t)</code> renvoie 1 <code>np.min(t, 0)</code> renvoie $\begin{pmatrix} 1 & 2 & 3 \end{pmatrix}$ <code>np.min(t, 1)</code> renvoie $\begin{pmatrix} 1 \\ 4 \end{pmatrix}$
<code>np.max</code>	Maximum des éléments d'un tableau	<code>np.max(t)</code> renvoie 6 <code>np.max(t, 0)</code> renvoie $\begin{pmatrix} 4 & 5 & 6 \end{pmatrix}$ <code>np.max(t, 1)</code> renvoie $\begin{pmatrix} 3 \\ 6 \end{pmatrix}$
<code>np.mean</code>	Moyenne des éléments d'un tableau	<code>np.mean(t)</code> renvoie 3.5 <code>np.mean(t, 0)</code> renvoie $\begin{pmatrix} 2.5 & 3.5 & 4.5 \end{pmatrix}$ <code>np.mean(t, 1)</code> renvoie $\begin{pmatrix} 2 \\ 5 \end{pmatrix}$
<code>np.median</code>	Médiane des éléments d'un tableau	<code>np.median(t)</code> renvoie 3.5 <code>np.median(t, 0)</code> renvoie $\begin{pmatrix} 2.5 & 3.5 & 4.5 \end{pmatrix}$ <code>np.median(t, 1)</code> renvoie $\begin{pmatrix} 2 \\ 5 \end{pmatrix}$
<code>np.var</code>	Variance des éléments d'un tableau	<code>np.var(t)</code> renvoie 2.916... <code>np.var(t, 0)</code> renvoie $\begin{pmatrix} 2.25 & 2.25 & 2.25 \end{pmatrix}$ <code>np.var(t, 1)</code> renvoie $\begin{pmatrix} 0.6... \\ 0.6... \end{pmatrix}$
<code>np.std</code>	Écart-type des éléments d'un tableau	<code>np.std(t)</code> renvoie 1.707... <code>np.std(t, 0)</code> renvoie $\begin{pmatrix} 1.5 & 1.5 & 1.5 \end{pmatrix}$ <code>np.std(t, 1)</code> renvoie $\begin{pmatrix} 0.816... \\ 0.816... \end{pmatrix}$
<code>np.cumsum</code>	Somme cumulée des éléments Aplatit le tableau si nécessaire	<code>np.cumsum(t)</code> renvoie $\begin{pmatrix} 1 & 3 & 6 & 10 & 15 & 21 \end{pmatrix}$ <code>np.cumsum(t, 0)</code> renvoie $\begin{pmatrix} 1 & 2 & 3 \\ 5 & 7 & 9 \end{pmatrix}$ <code>np.cumsum(t, 1)</code> renvoie $\begin{pmatrix} 1 & 3 & 6 \\ 4 & 9 & 15 \end{pmatrix}$

## I.5 - Pyplot - Graphiques

Module pour effectuer des rendus graphiques :

```
import matplotlib.pyplot as plt
```

<b>Tracé</b>	
<code>plt.plot(X, Y)</code>	X : liste des abscisses Y : liste des ordonnées Crée le graphique contenant le tracé de la suite de points
<code>plt.show()</code>	Montre le graphique
<b>Compléments</b>	
<code>plt.xlim(xmin, xmax)</code>	xmin : abscisse minimale xmax : abscisse maximale
<code>plt.ylim(ymin, ymax)</code>	ymin : ordonnée minimale ymax : ordonnée maximale
<code>plt.axis([xmin, xmax, ymin, ymax])</code>	Fixe les abscisses / ordonnées minimales / maximales
<code>plt.grid(True)</code>	Affiche le quadrillage
<code>plt.grid(False)</code>	Masque le quadrillage
<code>plt.legend()</code>	Affiche la légende.
<b>Graphiques particuliers</b>	
<code>plt.hist(x)</code>	Crée un histogramme avec les valeurs de x Choix des critères automatique ou à préciser avec une option
<code>plt.bar(x, hauteur)</code> height liste des hauteurs des barres	x liste des abscisses des barres
<code>plt.boxplot</code>	Boîtes à moustaches

## I.6 - Random - Pseudo-alea

Module pour utiliser des nombres pseudo-aléatoires :

```
import numpy.random as rd
```

<b>Loi uniforme</b>	
<code>rd.rand(n, p)</code>	Renvoie un tableau à n lignes et p colonnes Chaque élément est la réalisation d'une variable aléatoire de loi uniforme sur $[0, 1]$

## I.7 - Pandas - Panel data - Gestion des données

Module pour manipuler des données :

```
import pandas as pd
```

Statistiques	
pd.mean	Moyenne des éléments du tableau par catégorie
pd.std	Écart-type des éléments du tableau par catégorie