

Démystifier les langages de haut niveau

Outils Numériques / Semestre 5
Institut d'Optique / B1_1

C'est quoi cette syntaxe ???



```
import numpy
```

- Que représentent ces différentes syntaxes ?

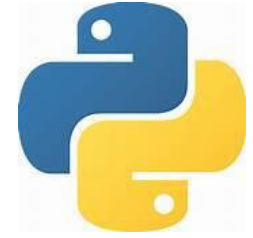
```
v = numpy.array([1, 2, 3])
```

```
a = v.max()
```

Hein!?

```
print( v.shape )
```

Distributions / Environnements



Distribution : ensemble de logiciels et de librairies
incluant des environnements et des interpréteurs

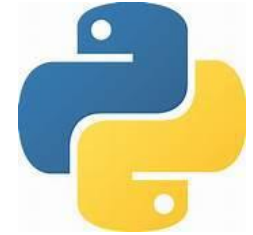


Bibliothèques : ensemble de modules supplémentaires
incluant des classes, des fonctions...

Environnement (IDE) : ensemble d'outils pour l'édition et l'interprétation des commandes / programmes
incluant des interpréteurs et des éditeurs de texte



Distributions / Environnements



- Installation de bibliothèques / packages

Dans un shell/prompt

```
> pip install numpy
```

Dans un shell/prompt (Anaconda)

```
> conda install numpy
```



Package : ***pip install SupOpNumTools***

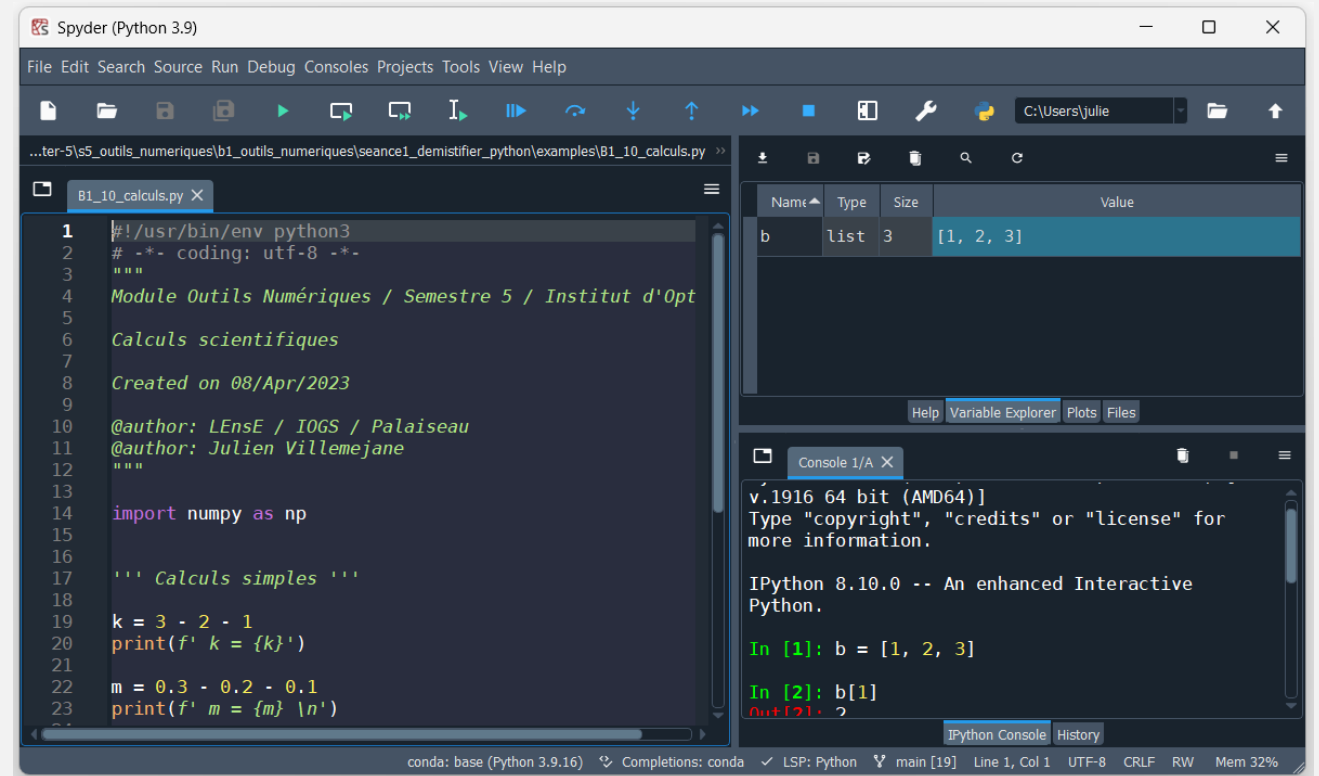
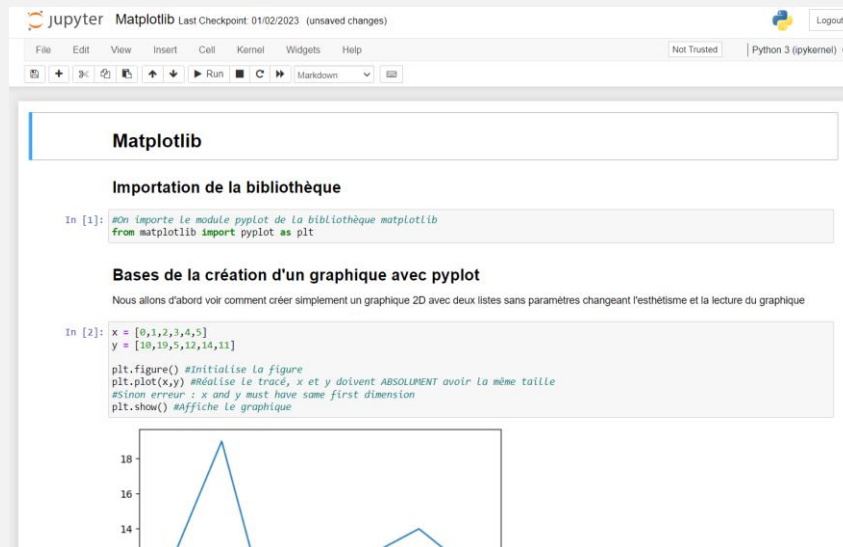
Dépôt : <https://github.com/IOGS-Digital-Methods/SupOpNumTools>

Doc : <https://iogs-digital-methods.github.io/SupOpNumTools/>

Coder en Python



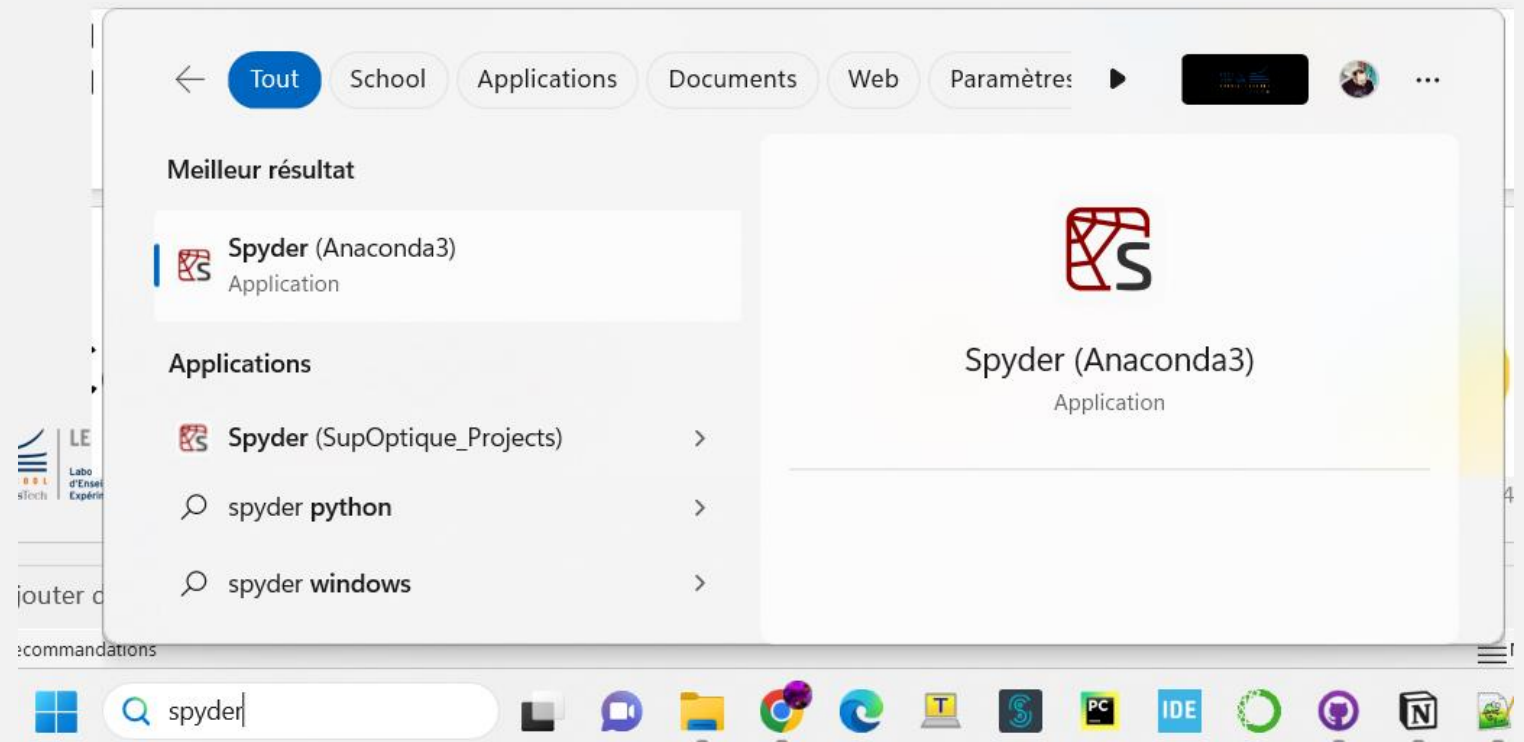
- Jupyter ou Spyder ?



Coder en Python



- Lancer Spyder

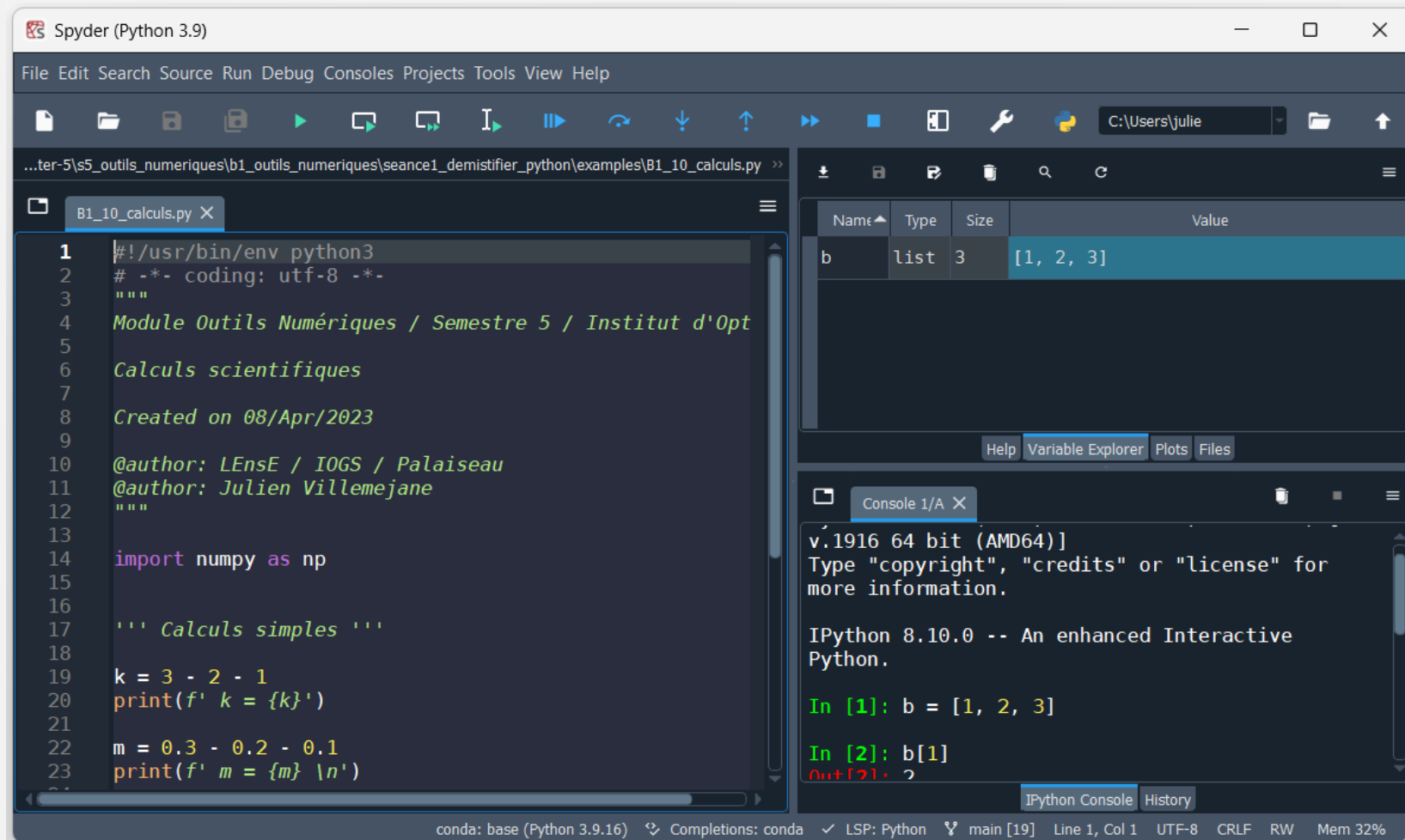


Coder en Python



Outils

Editeur
de texte



Variables

Console

Trucs et Astuces



- Sections

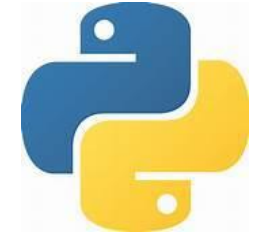
#%%

```
36
37 #%% Frequency Response / Bode
38 w = np.logspace(1, 6, 101)
39 mag, phase, w = ct.bode_plot(sysRC.getTF(), w, plot=True)
40 mag_db = 20*np.log(mag)
41 phase_deg = phase * 180 / np.pi
42 f = w/(2*np.pi)
```

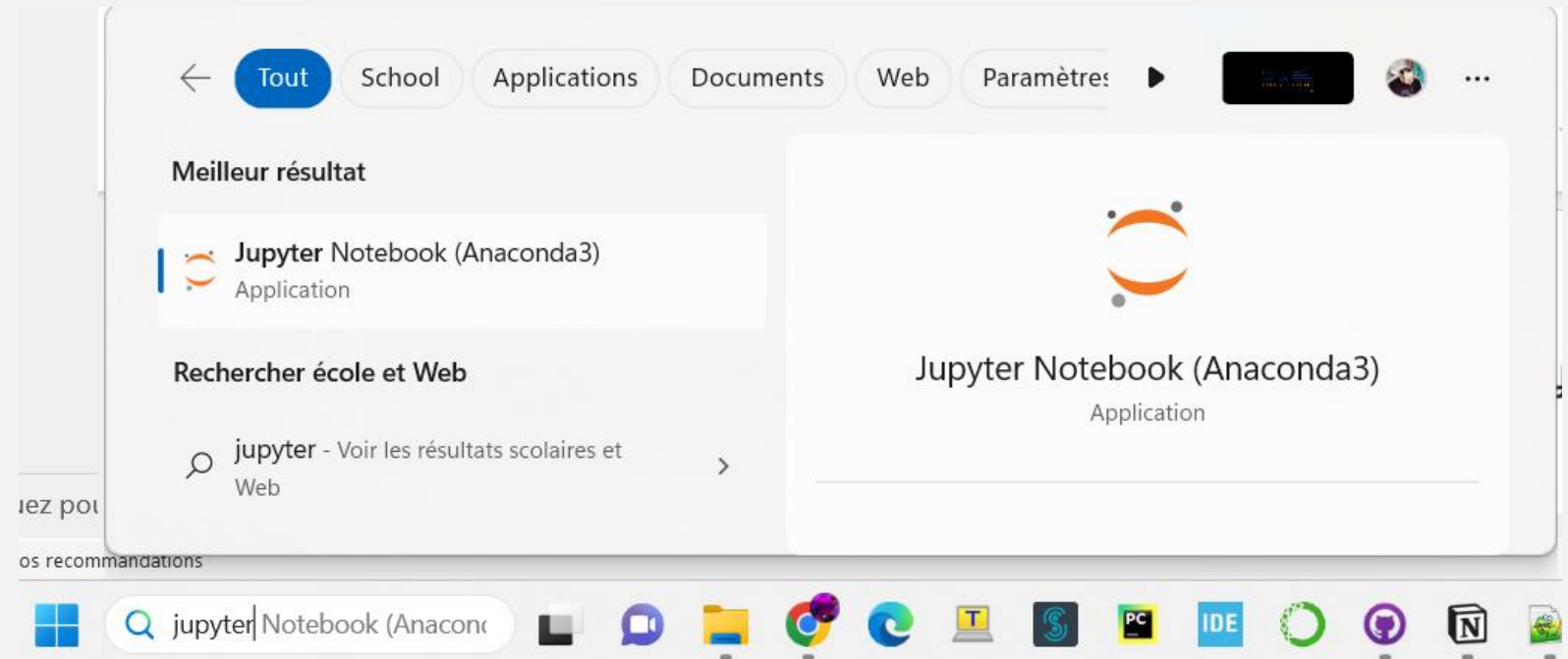
*Exécutables indépendamment
(... ou presque)*



Coder en Python



- Lancer Jupyter



Coder en Python



Outils

Editeur de
textes
pré-formatés
(Markdown)

jupyter Matplotlib Last Checkpoint: 01/02/2023 (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Matplotlib

Importation de la bibliothèque


```
In [1]: #On importe le module pyplot de la bibliothèque matplotlib
from matplotlib import pyplot as plt
```

Bases de la création d'un graphique avec pyplot

Nous allons d'abord voir comment créer simplement un graphique 2D avec deux listes sans paramètres changeant l'esthétique et la lecture du graphique

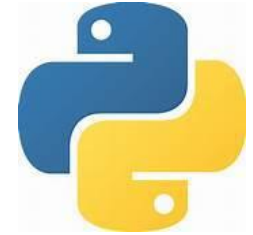
```
In [2]: x = [0,1,2,3,4,5]
y = [10,19,5,12,14,11]

plt.figure() #Initialise la figure
plt.plot(x,y) #Réalise le tracé, x et y doivent ABSOLUMENT avoir la même taille
#Sinon erreur : x and y must have same first dimension
plt.show() #Affiche le graphique
```



Serveur
Web local

Quelques rappels sous Python



- Variables

```
a = 2 + 3  
print( a )  
print( 'a =', a )    ou    print( f'a = {a}' )
```

```
5  
a = 5
```

- Listes

```
b = [1, 2, 3]  
print( b )
```

```
[1, 2, 3]
```

```
print( b[1] )
```

```
2
```

```
a = b + b
```

```
??
```

Doit-on faire confiance aux ordinateurs ?

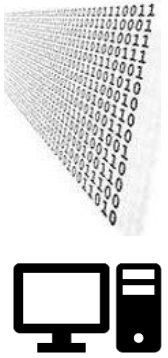
- Testez les deux calculs suivants sous Python

> 3 – 2 – 1

> 0.3 – 0.2 – 0.1



Codage des informations en machine



- Représentation binaire

Deux niveaux de tension possible uniquement en machine

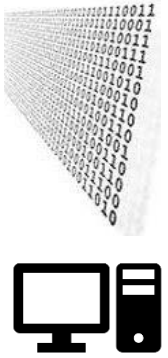
Meilleure robustesse pour la transmission de données sur de longues distances

Chaque **donnée binaire** est appelée **BIT** (***B**inary **digiT***)

Un **mot binaire** est composé de plusieurs chiffres binaires

Pour un mot binaire de n bits, il est possible d'obtenir 2^n combinaisons

Codage des informations en machine



- Différentes sortes de données à coder

Des données numériques

Entiers naturels

Entiers relatifs

Réels

Des données non-numériques

Caractères alphanumériques

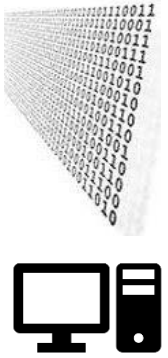
	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	'	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	\$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	BS	CAN	(8	H	X	h	x
1001	HT	EM)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[k	{
1100	FF	FS	,	<	L	\	l	
1101	CR	GS	-	=	M]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

ASCII
(7 bits)

UNICODE
UTF-8/16/32

A	0000 0000 0100 0001
S	0000 0000 0101 0011
C	0000 0000 0100 0011
I	0000 0000 0100 1001
I	0000 0000 0100 1001
	0000 0000 0010 0000
天	0101 1001 0010 1001
地	0101 0111 0011 0000
	0000 0000 0010 0000
س	0000 0110 0011 0011
ل	0000 0110 0100 0100
ا	0000 0110 0100 0111
م	0000 0110 0100 0101
	0000 0000 0010 0000
α	0000 0011 1011 0001
₹	0010 0010 0111 0000
γ	0000 0011 1011 0011

Codage des informations en machine



- Différentes sortes de données à coder

Des données numériques

Entiers naturels

Entiers relatifs

Réels

Des données non-numériques

Caractères alphanumériques

	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	'	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	\$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	BS	CAN	(8	H	X	h	x
1001	HT	EM)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[k	{
1100	FF	FS	,	<	L	\	l	
1101	CR	GS	-	=	M]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

ASCII
(7 bits)

UNICODE
UTF-8/16/32

A	0000 0000 0100 0001
S	0000 0000 0101 0011
C	0000 0000 0100 0011
I	0000 0000 0100 1001
I	0000 0000 0100 1001
	0000 0000 0010 0000
天	0101 1001 0010 1001
地	0101 0111 0011 0000
	0000 0000 0010 0000
س	0000 0110 0011 0011
ج	0000 0110 0100 0100
ا	0000 0110 0010 0111
م	0000 0110 0100 0101
	0000 0000 0010 0000
α	0000 0011 1011 0001
£	0010 0010 0111 0000
γ	0000 0011 1011 0011



$1 \neq '1'$

Codage des informations en machine



- Nombres entiers

Nombre fini de valeurs
sur un intervalle donné

Sur N bits : 2^N combinaisons

2^N entiers naturels de 0 à 2^N-1

$$0b\ 1011 = 1.2^3 + 0.2^2 + 1.2^1 + 1.2^0$$

2^N entiers relatifs de -2^{N-1} à $2^{N-1}-1$

0b **1**011

signe

Codage des informations en machine



- Nombres entiers

Nombre fini de valeurs
sur un intervalle donné

Sur N bits : 2^N combinaisons

2^N entiers naturels de 0 à $2^N - 1$

$$0b\ 1011 = 1.2^3 + 0.2^2 + 1.2^1 + 1.2^0$$

2^N entiers relatifs de -2^{N-1} à $2^{N-1} - 1$

0b **1**011

signe



$-0 \neq 0$

Complément à 2

Codage des informations en machine



- Nombres entiers

Nombre fini de valeurs
sur un intervalle donné

Sur N bits : 2^N combinaisons

2^N entiers naturels de 0 à $2^N - 1$

$$0b\ 1011 = 1.2^3 + 0.2^2 + 1.2^1 + 1.2^0$$

2^N entiers relatifs de -2^{N-1} à $2^{N-1} - 1$

0b **1**011

signe



$-0 \neq 0$

Complément à 2

- Nombres réels

Infinité de valeurs
sur un intervalle donné

Normalisation des informations

IEEE 754, datant de 1985

Simple précision : 32 bits

Double précision : 64 bits

Codage des informations en machine



- Norme IEEE754 / Simple



$$\text{valeur} = s \times 2^e \times m$$

Cas normalisé

Possibilité de coder l'infini

Plus petite valeur codifiée
 $1,175\,494\,35 \times 10^{-38}$

- Nombres réels

**Infinité de valeurs
sur un intervalle donné**

Normalisation des informations

IEEE 754, datant de 1985

Simple précision : 32 bits

Double précision : 64 bits

Codage des informations en machine



- Exemple en C++

```
int main(void){  
    int a = 3, b = 2;  
    float k = 2.5;  
    int c = a / b;  
    cout << "c = " << c << endl;  
    float d = a / b;  
    cout << "d = " << d << endl;  
    return 0;  
}
```

```
c = 1  
d = 1
```

Process returned 0 (0x0) execution time : 0.053 s
Press any key to continue.

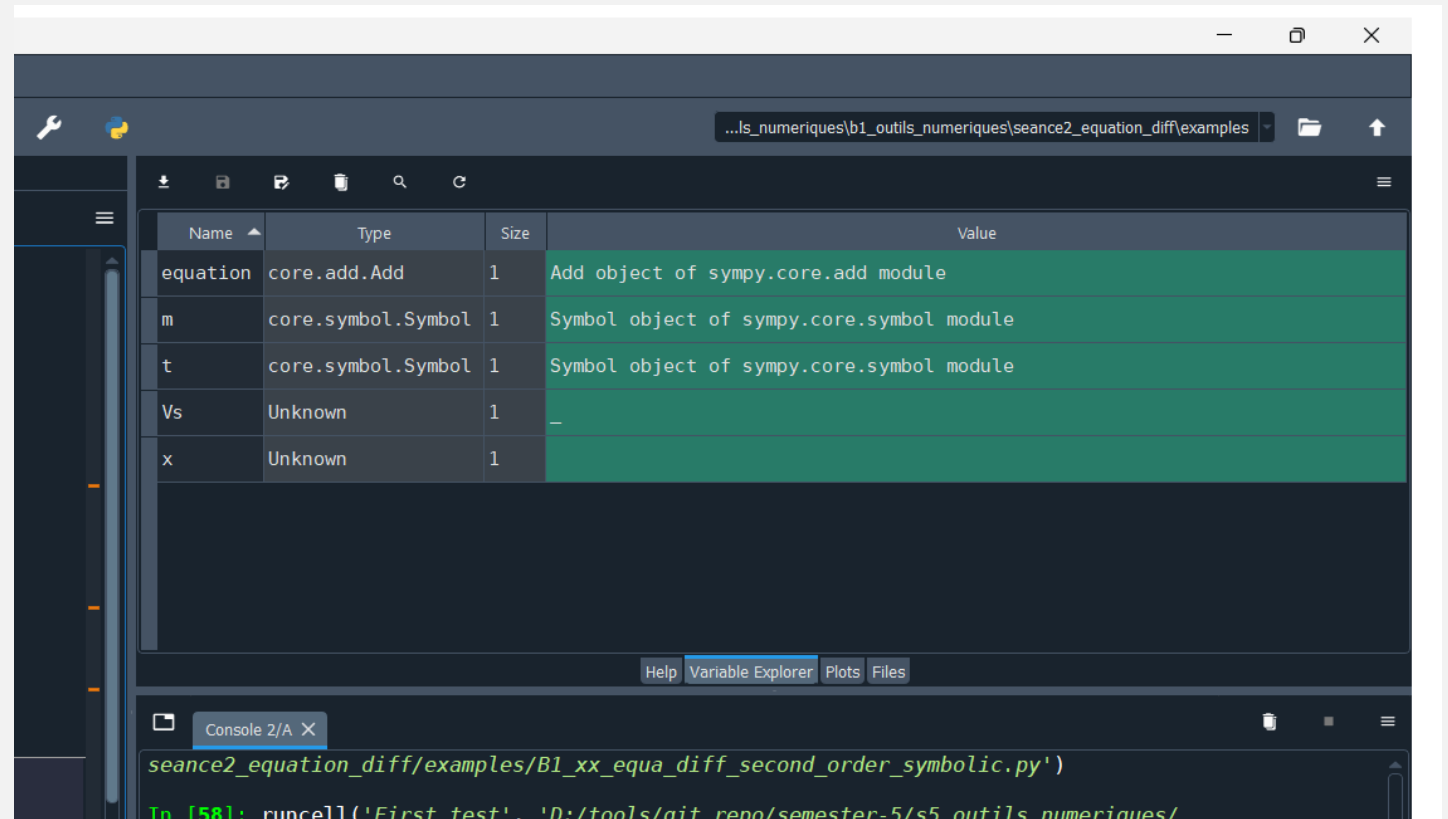
```
float e = (float) a / b;  
cout << "e = " << e << endl;
```

```
e = 1.5
```

Trucs et Astuces



- *Variable explorer*



Trucs et Astuces

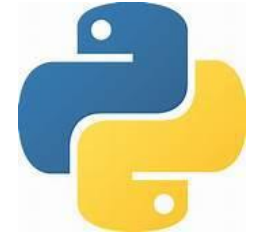


- Connaître le type de données

```
k = 1  
print( f'k = {k}' )  
print( type( k ) )
```

```
k = 1  
<class 'int'>
```


Quelques rappels sous Python



- Utilisation de bibliothèques

```
import numpy  
ma = numpy.array( [1, 2, 3] )
```

```
import numpy as np  
ma = np.array( [1, 2, 3] )
```

```
from matplotlib import pyplot  
pyplot.figure()
```

```
from matplotlib import pyplot as plt  
plt.figure()
```



Quelques rappels sous Python



- Utilisation des vecteurs / matrices

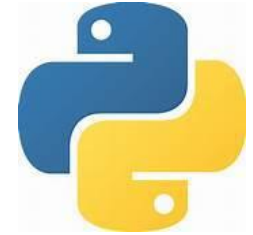
```
import numpy as np
x = np.array( [1,2,3] )
y = np.sin(x)
print( y )
```

???

```
mb = np.array( [[1,2,3] , [4,5,6]] )
mc = np.array( [[1,2,3] , [4,5,6]] )
mm = mb + mc
print( mm )
```

```
[[ 2  4  6]
 [ 8 10 12]]
```

Quelques rappels sous Python



- Quelques vecteurs particuliers

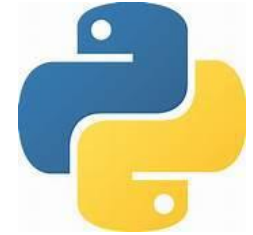
```
import numpy as np
vz = np.zeros( 10 )
print( f'shape of vz : {vz.shape}' )
print( f'values of vz : {vz}' )
```

???

```
mo = np.ones( (10,3) )
print( f'shape of mo : {mo.shape}' )
print( f'values of mo : {mo}' )
```

???

Quelques rappels sous Python



- Quelques vecteurs particuliers

```
import numpy as np
vlin = np.linspace( -1, 3, 21 )
print( f'shape of vlin : {vlin.shape}' )
print( f'values of vlin : {vlin}' )
```

???

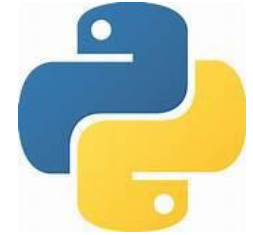
```
vlog = np.logspace( 1, 5, 11 )
print( f'shape of vlog : {vlog.shape}' )
print( f'values of vlog : {vlog}' )
```

???

```
vara = np.arange( 5, step=0.5 )
print( f'shape of vara : {vara.shape}' )
print( f'values of vara : {vara}' )
```

???

Quelques rappels sous Python



- Travailler avec des vecteurs

```
import numpy as np  
mb = np.array( [[1,2,3] , [4,5,6]] )
```

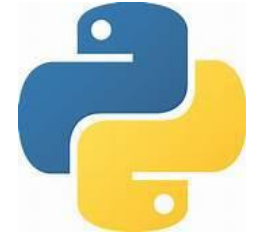
```
total = np.sum(mb)  
total_c = np.sum(mb, axis=0)  
total_r = np.sum(mb, axis=1)
```

Que contiennent les variables ***total***,
total_c et ***total_r*** ?

```
moy = np.mean(mb)  
moy_c = np.mean(mb, axis=0)  
moy_r = np.mean(mb, axis=1)
```

Que contiennent les variables ***moy***,
moy_c et ***moy_r*** ?

Quelques rappels sous Python



- Travailler avec des vecteurs

```
import numpy as np  
vect = np.arange( 100 )  
vect_p = vect[ 10 : 30 ]  
vect_s = vect[ 50 : ]
```

Que contiennent les variables ***vect***,
vect_p et ***vect_s*** ?

```
c = vect[(vect > 2) & (vect < 11)]  
tf = (vect > 2) & (vect < 11)
```

Que contiennent les variables ***c*** et ***tf*** ?

```
mb = np.array( [[1,2,3] , [4,5,6]] )  
mc = mb[ : , 1:3 ]
```

Que contient la variable ***mc*** ?

Trucs et Astuces



- Affichage des figures

Tools / Preferences

OU

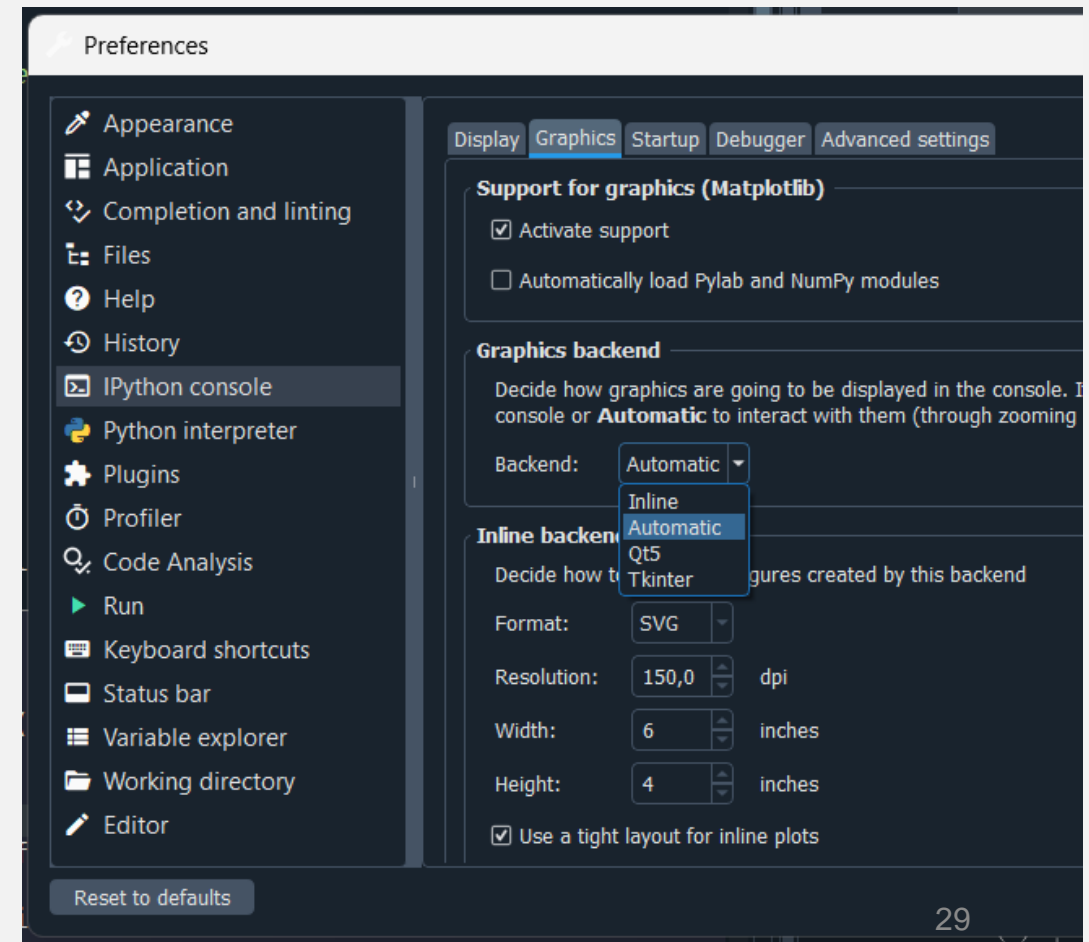
Outils / Préférences

IPython console

Graphics

Activate Support

Backend : **Automatic**



Puis-je utiliser de la même manière...

- des listes (*list*)

```
b = [1, 2, 3]  
a = b + b  
print( a )
```

???

- des matrices (*np.array*)

```
vb = np.array( [1,2,3] )  
va = vb + vb  
print( va )
```

???

Quelques rappels sous Python



- Nombres complexes

```
import numpy as np  
mk = np.array([1j, 2, 3], dtype=complex)  
print( mk )
```

```
[ 0+1j  2+0j  3+0j]
```

```
nk = 1j + 3  
print( nk )  
print( type( nk ) )
```

```
(1j + 3)  
<class 'complex'>
```

Résoudre des problèmes linéaires

- Equation polynomiale

$$-6.x^2 - 2.x + 4 = 0$$

```
import numpy.polynomial.polynomial as  
nppol
```

```
X = nppol.polyroots( [ 4, -2, -6] )  
print( X )
```

Résoudre des problèmes linéaires

- Système d'équations

$$\begin{cases} a_1 \cdot x + b_1 \cdot y = c_1 \\ a_2 \cdot x + b_2 \cdot y = c_2 \end{cases}$$

- Représentation matricielle

$$A = \begin{pmatrix} a_1 & b_1 \\ a_2 & b_2 \end{pmatrix} \quad b = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}$$

$$X = \begin{pmatrix} x \\ y \end{pmatrix}$$

Si le système possède une solution,
alors la **matrice A est inversible**
et le résultat peut s'obtenir par :

$$X = A^{-1} \cdot b$$

Résoudre des problèmes linéaires



- Système d'équations

$$\begin{cases} a_1 \cdot x + b_1 \cdot y = c_1 \\ a_2 \cdot x + b_2 \cdot y = c_2 \end{cases}$$

$$A = \begin{pmatrix} a_1 & b_1 \\ a_2 & b_2 \end{pmatrix} \quad b = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}$$

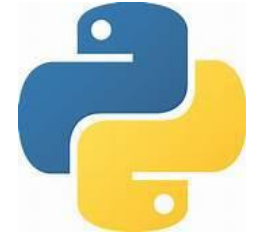
- Résolution numérique

```
from scipy import linalg  
X = linalg.solve( A , b )
```

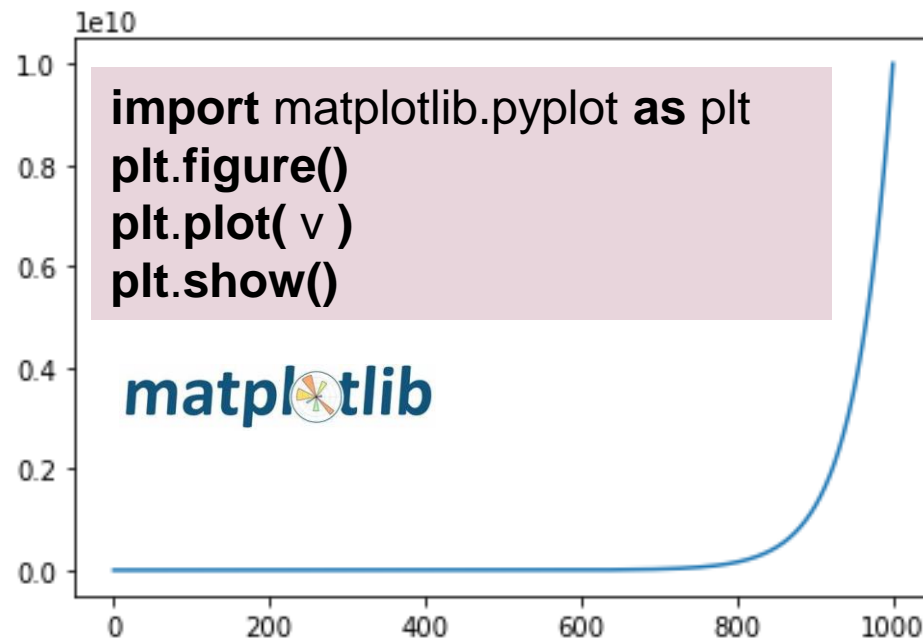
Si le système possède une solution,
alors la **matrice A est inversible**
et le résultat peut s'obtenir par :

$$X = A^{-1} \cdot b$$

Quelques rappels sous Python

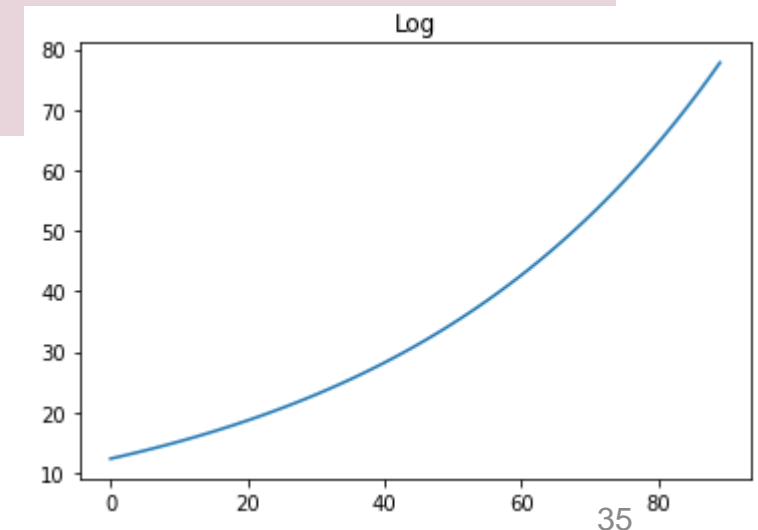


- Vecteurs (suite)

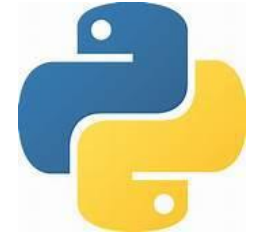


```
import numpy as np
v = np.logspace( 1, 10, 1001 )
```

```
v2 = v[ 10 : 100 ]
plt.figure()
plt.plot( v2 )
plt.show()
```



Quelques rappels sous Python

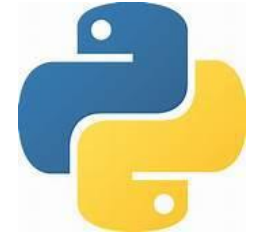


- Gestion des erreurs

```
try:  
    g = int( input('Saisir un entier : ') )  
    a = 5/g  
    print( f'g = {g} et a = {a}' )
```

```
except ValueError :  
    print('Vous n\'avez pas saisi un entier !')  
except ZeroDivisionError :  
    print('Division par 0 !!')  
except :  
    print('Erreur !!!')
```


Arguments au lancement



- TO DO

***args et **kwargs**

Bibliographie

Document rédigé par Julien VILLEMEJANE
LEnSE / Institut d'Optique / France

<http://lense.institutoptique.fr/>

Création : Avril 2023

- ***Python pour le calcul symbolique*** – *WikiBooks*
https://fr.wikibooks.org/wiki/Python_pour_le_calcul_scientifique