Programmation en Python

Table des matières

Introduction

La distribution Anaconda

Présentation de Jupyter

Outils Logiciels pour l'enseignement supérieur

Les librairies scientifiques

Programmation de base :

- Déclaration de variables
- Les opérateurs
- Les boucles
- Les fonctions
- Les courbes
- Réutilisation des scripts
- Les nombres aléatoires
- Les nombres complexes
- Les matrices
- Algèbre linéaire

Programmation avancée :

Introduction

Python est un langage de programmation interprété, c'est-à-dire que les instructions que vous lui envoyez sont "transcrites" en langage machine au fur et à mesure de leur lecture. D'autres langages (comme le C / C++) sont appelés "langages compilés" car, avant de pouvoir les exécuter, un logiciel spécialisé se charge de transformer le code du programme en langage machine. On appelle cette étape la "compilation". À chaque modification du code, il faut rappeler une étape de compilation.

Les avantages d'un langage interprété sont la simplicité (on ne passe pas par une étape de compilation avant d'exécuter son programme) et la portabilité (un langage tel que Python est censé fonctionner aussi bien sous Windows que sous Linux ou Mac OS, et on ne devrait avoir à effectuer aucun changement dans le code pour le passer d'un système à l'autre). Cela ne veut pas dire que les langages compilés ne sont pas portables, loin de là ! Mais on doit utiliser des compilateurs différents et, d'un système à l'autre, certaines instructions ne sont pas compatibles, voire se comportent différemment.

Introduction

En contrepartie, un langage compilé se révélera bien plus rapide qu'un langage interprété (la traduction à la volée de votre programme ralentit l'exécution), bien que cette différence tende à se faire de moins en moins sentir au fil des améliorations. De plus, il faudra installer Python sur le système d'exploitation que vous utilisez pour que l'ordinateur puisse comprendre votre code.

La distribution Anaconda

Présentation de la distribution Anaconda

- Distribution Python 3 dédié aux calculs scientifiques.
- Environnement de Développement Intégré (Spyder)
- Outils de création d'interface (Qt Designer,...)
- Large panel de fonctionnalités scientifiques

Figure 1: Distribution Scientifique

Figure 2: Environnement Spyder

La distribution Anaconda - Installation

Procédure d'installation

- ☹ Manuelle : installation de Python, puis pip, puis jupyter . . .
- ☺ Automatique : installation de la distribution Anaconda, version Python 3.9
  - url : https://www.anaconda.com/products/distributionDownloads

Anaconda Installers

Windows

Python 3.9

64-Bit Graphical Installer (510 MB)

32-Bit Graphical Installer (404 MB)

MacOS

Python 3.9

64-Bit Graphical Installer (515 MB)

64-Bit Command Line Installer (508 MB)

Linux

Python 3.9

64-Bit (x86) Installer (581 MB)

64-Bit (Power8 and Power9) Installer (255 MB)

64-Bit (AWS Graviton2 / ARM64) Installer (488 M)

64-bit (Linux on IBM Z & LinuxONE) Installer (242 M)

La distribution Anaconda - Installation

Procédure d'installation

- ☹ Manuelle : installation de Python, puis pip, puis jupyter . . .
- ☺ Automatique : installation de la distribution Anaconda, version Python 3.9
  - url : https://www.anaconda.com/products/distributionDownloads

CHARLES CARNUS
LYCÉE TECHNIQUE PRIVÉ D'ENSEIGNEMENT SUPÉRIEUR

Page d'accueil Anaconda

La distribution Anaconda - Ouverture de Jupyter

Ouvrir Jupyter Notebook en cliquant sur "Launch"

2ème méthodes

- Dans un terminal, lancer la ligne de commande :

```
$ jupyter notebook
```

Présentation de Jupyter

Jupyter Notebook

Figure 3: Interface d'un notebook jupyter

Présentation de Jupyter

Interface des notebooks

- Tout se passe dans le navigateur web (Firefox, Edge, Safari, etc)
- Un notebook est composé de "cellules"
- Une cellule peut contenir du
  - code python,
  - du code markdown (edition du texte).

Figure 4: Les différents types de cellules.

Présentation de Jupyter

Le Markdown

- En bref
  - Langage simple plus concis que le Latex ou le HTML.
  - Possibilités de conversions multiples via l'utilitaire pandoc.
- Liste des commandes : https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet

Exemple:

```
# Ceci est un titre

## Ceci est un sous titre

Ce qui suit est une liste classique

* ceci est un item
* ceci est egalement un item

Nous pouvons egalement faire des
listes numerotees

1. ceci est un item numerote
2. idem
```

Ceci est un titre

Ceci est un sous titre

Ce qui suit est une liste classique

- ceci est un item
- ceci est également un item

Nous pouvons également faire des listes numérotées

1. ceci est un item numéroté
2. idem

CHARLES CARNUS
LYCÉE TECHNIQUE PRIVÉ D'ENSEIGNEMENT SUPÉRIEUR

Les commandes Python spécifiques à Jupyter

- Gestion des figures avec Matplotlib

```
%matplotlib inline        #affichage des courbes dans le navigateur
```

ou

```
%matplotlib qt            #affichage des courbes dans une fenetre qt
```

- Lecture de fichiers sonores

```
import IPython     #import de la librairie IPython
IPython.display.Audio("mon_fichier.wav")
```

- Gestion des interactions

```
from ipywidgets import interact #import de la librairie

def f(x):      # creation d'une fonction python
    print(3*x+4)

interact(f, x=10) #creation de l'interaction
```

Présentation de Jupyter

Exemple

```
%matplotlib inline
from ipywidgets import interact
import numpy as np
import matplotlib.pyplot as plt

Fe=1000
f=2
t=np.arange(Fe)/Fe
theta=2*np.pi*f*t

def f_x(a1,a2,a3,a4):
    a=[a1,a2,a3,a4]
    x=np.zeros(Fe)

    for n in range(4):
        x=x+a[n]*np.sin((n+1)*theta)

    plt.plot(t,x)

interact(f_x,a1=0.,a2=0.,a3=0.,a4=0.)
```

Outils Logiciels pour l'enseignement supérieur

Cahier des charges

- Fonctions mathématiques poussées (fonctions trigonométriques, complexes, ...etc)
- Outils de visualisation 2D, 3D,
- Outils de calcul formel,
- ...

Quelques outils possibles

Matlab, Scilab, . . . Python !

Le Langage Python

En bref

- Créateur: Guido van Rossum.
- Versions:
  - Python 1 (1990)
  - ...
  - **Python 3.11.2 (24 octobre 2022)**
- Caractéristiques du langage:
  - Multiplateforme.
  - Langage de script.
  - Programmation procédurale, objet,...

Python par rapport aux autres langages

Classement TIOBE (Mars 2023)

Mar 2023	Mar 2022	Change	Programming Language	Ratings	Change
1	1		Python	14.83%	+0.57%
2	2		C	14.73%	+1.67%
3	3		Java	13.56%	+2.37%
4	4		C++	13.29%	+4.64%
5	5		C#	7.17%	+1.25%
6	6		Visual Basic	4.75%	-1.01%
7	7		JavaScript	2.17%	+0.09%
8	**10**	^	**SQL**	**1.95%**	**+0.11%**
9	8	v	PHP	1.61%	-0.30%
10	13	^	Go	1.24%	+0.26%

Python par rapport aux autres langages

Classement TIOBE (Mars 2023)

Mar 2023	Mar 2022	Change	Programming Language	Ratings	Change
11	9	⌄	Assembly language	1.11%	-0.79%
12	15	⌃	MATLAB	1.08%	+0.28%
13	12	⌄	Delphi/Object Pascal	1.06%	-0.06%
14	23	⌃⌃	Scratch	1.00%	+0.47%
15	17	⌃	Classic Visual Basic	0.98%	+0.38%
16	11	⌄⌄	R	0.93%	-0.44%
17	**30**	⌃⌃	**Fortran**	**0.79%**	**+0.40%**
18	16	⌄	Ruby	0.76%	+0.10%
19	26	⌃⌃	Rust	0.73%	+0.22%
20	14	⌄⌄	Swift	0.71%	-0.20%

Python Vs C

Exemple

Langage C

```c
#include <stdio.h>

int main( void )
{
    long long int k,S=0;
    for (k=0;k<10000000;k++)
    {
        S=S+k;
    }
    printf ("%lli\n" ,S);
    return 0;
}
```

Python

```python
S=0
for k in range(10000000):
    S=S+k
print (S)
```

Figure 5: Exemple: Somme d'une suite arithmétique ($S = 0 + 1 + \cdots + N$)

- Python est un langage plus concis et plus haut niveau que le langage C. Il permet de coder des applications plus rapidement.
- C reste un langage nettement plus rapide à l'execution.

Python Vs Matlab Vs Scilab (Déc 2021)

Figure 6: Matlab

Figure 7: Scilab

Figure 8: Python Spyder

Langage	Matlab	Scilab	Python
Développeur	Mathworks	Scilab	PSF
Version 1	1984	1994	1990
Dern. Version	R2022b	6.1.1	3.11.2
Prix (€)	> 3000	0	0
TIOBE	12	> 100	1
Github Repo	60 575	895	1 474 838
Indeed	1 725	81	12 786
APEC	375	13	3 241

Table 1: Comparaison Matlab, Scilab, Python

Figure 9: Popularité des mots clés (Google)

Python Vs C Vs C++ (Déc 2021)

Langage	C	C++	Python
TIOBE	2	4	1
Github Repo	170 523	156 442	1 474 838
Indeed	3 125	2 518	12 786
APEC	2 319	1 493	3 241

Table 2: Comparaison C, C++, Python

Figure 10: Popularité des mots clés (Google)

Les bases du langage Python

CHARLES CARNUS
LYCÉE TECHNIQUE PRIVÉ D'ENSEIGNEMENT SUPÉRIEUR

Un programme type

```
import webbrowser                    #importation des libraires
import os
from csv import reader
from numpy.random import uniform

file_out="csv/notes.html"            #affectation
f = open("csv/liste_S1.csv", 'rt')
html="<html>\n<body>\n<h1>Resultats du S1: BTS SN</h1>\n<table>\n"
csv_data = reader(f,delimiter=';')

for row in csv_data:                 #structure de controle for
    note=uniform(0,20)
    html=html+ "<tr><td>%s</td><td>%.2f</td>\n" %(row[1],note)
    if round(note, 2) >=10:          #structure de controle if
        html=html+"<td>Valide<td></tr>\n"
    else:
        html=html+"<td>Echec<td></tr>\n"

f.close()
html=html+"</table>\n</body>\n</html>"
file = open("csv/notes.html", "w")
file.write(html)
file.close()
webbrowser.open_new_tab("file://"+os.getcwd()+"/"+file_out)
```

Les librairies scientifiques

C'est un regroupement de fonctions prédéfinies, qui une fois importées permettent d'accéder à de nouvelles fonctions.
Ces librairies (modules ou bibliothèques) sont importées au début du programme via l'instruction `import`.
On peut citer :

- `numpy` : permet de faire du calcul scientifique
- `scipy` : permet de faire du traitement du signal ...
- `matplotlib` : permet de faire des graphiques
- `statistics` : permet de calculer des valeurs statistiques basiques (moyenne, médiane, variance, ...)
- ...

Les librairies scientifiques

- `import module` : il faudra alors faire précéder les fonctions de ce module du préfixe module. Par exemple `math.sqrt(2)` va renvoyer la racine carrée de 2 avec le module math si on a écrit en début de programme : `import math`
- `from module import fonction` : la fonction `"fonction"` peut alors s'utiliser directement (recommandé). Par exemple `sqrt(2)` va renvoyer la racine carrée de 2 avec le module `math` si on a écrit en début de programme : `from math import sqrt`
- `from module import *` : toutes les fonctions du module `"module"` sont alors importées (à éviter). Par exemple `sqrt(2)` va renvoyer la racine carrée de 2 avec le module `math` si on a écrit en début de programme : `from math import *`

Les librairies scientifiques

- Instruction inconnue

```
cos(pi)
```

```
NameError                                 Traceback (
    most recent call last)
<ipython-input-4-a971a9dfc2c9> in <module>
----> 1 cos(pi)

NameError: name 'cos' is not defined
```

- Solution 1

```
from math import *
cos(pi)
```

```
-1.0
```

- Solution 2

```
import numpy as np
np.cos(np.pi)
```

```
-1.0
```

Les librairies scientifiques

CHARLES CARNUS
LYCÉE TECHNIQUE PRIVÉ D'ENSEIGNEMENT SUPÉRIEUR

Numpy (from numpy import *)

Librairie permettant la manipulation et les opérations sur les tableaux, vecteurs, et matrices, etc.

Code Listing 1: Sans Numpy

```
from math import *

angles=[0,pi/6,pi/4,pi/3,pi/2]

mes_cosinus=[]

for k in range(len(angles)):
    valeur=cos(angles[k])
    mes_cosinus.append(valeur)

print(mes_cosinus)
```

Code Listing 2: Avec Numpy

```
from numpy import *

angles=[0,pi/6,pi/4,pi/3,pi/2]

#creation d'un tableau numpy
tableau=array(angles)

mes_cosinus=cos(tableau)

print(mes_cosinus)
```

Figure 11: Exemple: Calcul des cosinus de plusieurs angles

Les librairies scientifiques

CHARLES CARNUS
LYCÉE TECHNIQUE PRIVÉ D'ENSEIGNEMENT SUPÉRIEUR

Numpy (from numpy import *)

- Création de tableaux

```
x1=array([10,2.1,4])     #tableau contenant 10, 2.1 et 4.
x2=zeros(10)             #tableau contenant 10 zeros
x3=ones(10)              #tableau contenant 10 fois le nombre 1.
x4=5*ones(10)            #tableau contenant 10 fois le nombre 5.
x5=arange(100)           #tableau contenant les valeurs 0, 1, ...,99
x6=arange(3,10,2)        #tableau contenant les valeurs 3, 5, 7, 9
```

- Accès aux éléments

```
x1[0]                    #acces au premier element de x1 c-à-d 10
x1[-1]                   #acces au dernier element de x1 c-à-d 4
x5[:10]                  #valeurs 0,1, ..., 9
x5[10:]                  #valeurs 10,11, ..., 99
x5[::2]                  #valeurs 0,  2,  4,  ... ,  98
```

- Fonctions

```
len(x1)                  #longueur du tableau x1 c-à-d 3
sort(x1)                 #tableau contenant les valeurs 2.1,4 et 10
```

Les librairies scientifiques

Matplotlib (from matplotlib.pyplot import *)

Librairie permettant l'affichage et l'export de courbes (2D, 3D).

```
from numpy import *
from matplotlib.pyplot import *

f0=2
#creation de la base temps
t=arange(1000)/1000

#creation de la sinusoide
x=cos(2*pi*f0*t)

#affichage
plot(t,x)
xlabel('temps(s)')
ylabel('sinusoide')
show()
```

Figure 12: Exemple: Affichage de la sinusoide $x(t) = \cos(2\pi f_0 t)$ avec $f_0 = 2$ Hz et $t = \frac{n}{1000}$ $(n \in \mathbb{N})$

Les librairies scientifiques

Matplotlib (from matplotlib.pyplot import *)

Figure 13: Gallerie Matplotlib

Les librairies scientifiques

Scipy (from scipy import *)

Librairie contenant un ensemble complet de fonctions mathématiques.

```
from numpy import *
from scipy.stats import *
from matplotlib.pyplot import *

N=10000
noise=norm.rvs(scale=0.05,size=N)

x=arange(-0.2,0.2,0.01)
pdf=norm.pdf(x,scale=0.05)

hist(noise,20, normed=True)
plot(x,pdf)
xlabel('x')
ylabel('pdf(x)')
show()
```

Figure 14: Exemple: Distribution Gaussienne

Les librairies scientifiques

Scipy (from scipy import *)

```
from numpy import *
from scipy.signal import *
from scipy.io.wavfile import *
from matplotlib.pyplot import *

Fe=44100
t=arange(0,10,1/Fe)
x=chirp(t,0,10,Fe/2)
write('audio/chirp.wav',Fe,x)

plot(t,x)
xlabel("temps(s)")

figure()
f, t, Sxx=spectrogram(x,Fe)
pcolormesh(t, f, Sxx)
xlabel("temp(s)")
ylabel("frequence (Hz)")
show()
```

Figure 15: Exemple: Création d'une sinusoïde modulée en fréquence

Les librairies scientifiques

Scipy (from scipy import *)

```
from numpy import *
from scipy.io.wavfile import *
from matplotlib.pyplot import *

filein='./audio/sample.wav'
fileout='./audio/sample_out.wav'
Fe,signal=read(filein)

#substract left / right channels
x_mono=signal[:,0]-signal[:,1]
write(fileout,Fe,x_mono)

f, ax = subplots(3)
ax[0].plot(signal[:,0])
ax[1].plot(signal[:,1])
ax[2].plot(x_mono)
xlabel("samples")
show()
```

Figure 16: Exemple: Suppression de la composante sonore placée au centre de l'image spatiale

Les librairies scientifiques

Scipy (from scipy import *)

```
from scipy.io import *
from matplotlib.pyplot import *
from skimage import color
from skimage import filters

M1=imread('img/lenna.png')
M2=color.rgb2gray(M1)
val = filters.threshold_otsu(M2)
M3 = M2 > val
M4=filters.gaussian_filter(M2, sigma=5)

f, ax= subplots(2, 2)
ax[0,0].imshow(M1)
ax[0,1].imshow(M2,cmap=cm.gray)
ax[1,0].imshow(M3,cmap=cm.gray)
ax[1,1].imshow(M4,cmap=cm.gray)
show()
```

Figure 17: Exemple: Traitement d'images

...Pour aller plus loin

Conception d'Interface Utilisateur (PyQt, TkInter, Kivy,...)

Figure 18: Exemple: Interface réalisée avec PyQt5

...Pour aller plus loin

CHARLES CARNUS
LYCÉE TECHNIQUE PRIVÉ D'ENSEIGNEMENT SUPÉRIEUR

Développement de sites Web

- Exemples de sites utilisant le framework Python/Django
  - Instagram,
  - Spotify,
  - Prezi,
  - ...
- Exemples de sites utilisant d'autres solutions basées sur Python
  - Youtube,
  - Dropbox,
  - ...
  - et le logiciel qui gère les notes dans les IUT !

Programmation de base : Déclaration de variables

CHARLES CARNUS
LYCÉE TECHNIQUE PRIVÉ D'ENSEIGNEMENT SUPÉRIEUR

Une variable, dans le domaine informatique, est un conteneur (espace mémoire) qui sert à stocker une valeur.
Par exemple, si dans le langage courant on dit z est égal à 10, on utilise la variable dont le nom est z pour lui fixer la valeur 10.
En informatique, cette opération est appelée "affectation" ("assignation"), et consiste à stocker une valeur en mémoire vive de l'ordinateur (RAM).
Les variables Python sont automatiquement créées au moment où on leur assigne une valeur.
Pour choisir le nom d'une variable, il faut respecter certaines règles :

- le nom doit commencer par une lettre ou par un underscore (non recommandé sauf cas très particuliers)
- le nom d'une variable ne doit contenir que des caractères alphanumériques courants (pas d'espace dans le nom d'une variable, ni de caractères spéciaux comme les caractères accentués ou tout autre signe comme le signe –)
- on ne peut pas utiliser certains mots qui possèdent déjà une signification spéciale dans Python (mots réservés).

Programmation de base : Déclaration de variables

CHARLES CARNUS
LYCÉE TECHNIQUE PRIVÉ D'ENSEIGNEMENT SUPÉRIEUR

```
a_1 = 'Bonjour'
b = "Salut"
phrase = "Lycée Charles Carnus"
z = 10
x5 = 3.14
A, B = 5.41, 12
C = D = 3
E = 5 + 16
D = A + E
liste1 = []                    #Liste vide
liste2 = [1, 4, 9]             #Liste de 3 éléments
liste3 = [[1, 2], [3, 4]] #Liste
t = 1, 5, 'Bonjour'            #Tuple
np.arange(3)              #array([0, 1, 2])
np.arange(3,6)            #array([3, 4, 5])
np.arange(3,7,2)          #array([3, 5])
np.array([1, 2, 3])             #array([1, 2, 3])
np.array([[1, 2], [3, 4]])  #array([[1, 2], [3, 4]])
```

Les commentaires

```
# Ceci est un commentaire qui sera ignoré à l'exécution

"""
Ceci est un commentaire
sur plusieurs lignes
"""
```

Programmation de base : Déclaration de variables

```
Z = np.linspace(0,1,11,endpoint=True)[1:]
print(Z)
```

```
[0.1  0.2  0.3  0.4  0.5  0.6  0.7  0.8  0.9  1. ]
```

```
Z = np.linspace(0,1,11,endpoint=True)
print(Z)
```

```
[0.   0.1  0.2  0.3  0.4  0.5  0.6  0.7  0.8  0.9  1. ]
```

```
Z = np.linspace(0,1,11,endpoint=False)[1:]
print(Z)
```

```
[0.0909 0.1818 0.2727 0.3636 0.4545 0.5455 0.6364 0.7273 0.8182 0.9091]
```

```
Z = np.linspace(0,1,11,endpoint=False)
print(Z)
```

```
[0. 0.0909 0.1818 0.2727 0.3636 0.4545 0.5455 0.6364 0.7273 0.8182 0.9091]
```

Programmation de base : Déclaration de variables

CHARLES CARNUS
LYCÉE TECHNIQUE PRIVÉ D'ENSEIGNEMENT SUPÉRIEUR

Si une variable n'est pas "définie" (si aucune valeur ne lui a été affectée), son utilisation produit une erreur :

```
>>> n
```

```
NameError                                 Traceback (most recent call last)
<ipython-input-1-ab0680a89434> in <module>
----> 1 n
NameError: name 'n' is not defined
```

En mode interactif, la dernière expression affichée est affectée à la variable _. Ainsi, lorsqu'on utilise Python comme calculatrice, cela nous permet de continuer des calculs facilement, par exemple :

```
>>> taxe = 12.5 / 100
>>> prix = 100.50
>>> prix * taxe
12.5625
>>> prix + _
113.0625
>>> round(_, 2)
113.06
```

Cette variable doit être considérée comme une variable en lecture seule par l'utilisateur.

Programmation de base : Chaîne de caractères

CHARLES CARNUS
LYCÉE TECHNIQUE PRIVÉ D'ENSEIGNEMENT SUPÉRIEUR

```
>>> word = 'Python'
>>> word[0]  # caractère en position 0
'P'
>>> word[5]  # caractère en position 5
'n'
```

Les indices peuvent également être négatifs, on compte alors en partant de la droite. Par exemple :

```
>>> word[-1]
'n'
>>> word[-2]
'o'
>>> word[-6]
'P'
```

```
>>> word[0:2]  # caractères de la position 0 (incluse) à 2 (exclue)
'Py'
>>> word[2:5]  # caractères de la position 2 (incluse) à 5 (exclue)
'tho'
```

Programmation de base : Chaîne de caractères

Les valeurs par défaut des indices de tranches ont une utilité ; le premier indice vaut zéro par défaut (c.-à-d. lorsqu'il est omis), le deuxième correspond par défaut à la taille de la chaîne de caractères.

```
>>> word[:2]    # caractères du début jusqu'à la position 2 (exclue)
'Py'
>>> word[4:]    # caractères de la position 4 (incluse) jusqu'à la
                                    fin
'on'
>>> word[-2:]   # caractères de la seconde position en partant de la
                                    fin (incluse) jusqu'à la fin
'on'
```

On note que le début est toujours inclus et la fin toujours exclue.
Les chaînes de caractères, en Python, ne peuvent pas être modifiées. On dit qu'elles sont immuables. Affecter une nouvelle valeur à un indice dans une chaîne produit une erreur.
La fonction native `len()` renvoie la longueur d'une chaîne :

```
>>> s = 'LycéeCharlesCarnus'
>>> len(s)
18
```

Programmation de base : Chaîne de caractères

CHARLES CARNUS
LYCÉE TECHNIQUE PRIVÉ D'ENSEIGNEMENT SUPÉRIEUR

```
>>> 'BTS SN'
```

```
'BTS SN'
```

```
>>> "BTS SN"
```

```
"BTS SN"
```

```
>>> 'L'école'
```

```
  File "<ipython-input-3-9e621d1e5ca0>", line 1
    'L'école'
          ^
SyntaxError: invalid syntax
```

```
>>> 'L\'école'
```

```
"L'école"
```

```
>>> "L'école"
```

```
"L'école"
```

```
>>> "L\'école"
```

```
"L'école"
```

```
s = 'Ligne1.\nLigne2.'
# \n correspond à
# nouvelle ligne
s
```

```
'Ligne1.\nLigne2.'
```

```
print(s)
```

```
Ligne1.
Ligne2.
```

```
print('BTS SN')
```

```
BTS SN
```

```
print("BTS SN")
```

```
BTS SN
```

Programmation de base : Chaîne de caractères

CHARLES CARNUS
LYCÉE TECHNIQUE PRIVÉ D'ENSEIGNEMENT SUPÉRIEUR

```
>>> print('C:\home\name')
# \n nouvelle ligne
```

```
C:\home
ame
```

```
>>> print(r'C:home\name')
# notez r avant simple quote
```

```
C:\home\name
```

```
>>> 'Un' + 'Deux' + 'Trois'
```

```
'UnDeuxTrois'
```

```
>>> 3 * 'Un' + 'Deux'
```

```
'UnUnUnDeux'
```

```
>>> print("""\
Utilisation: machin [OPTIONS]
     -h              Afficher le message
     -H hostname     Hôte auquel se connecter
""")
```

```
Utilisation: machin [OPTIONS]
     -h              Afficher le message
     -H hostname     Hôte auquel se connecter
```

Programmation de base : `print()`

CHARLES CARNUS
LYCÉE TECHNIQUE PRIVÉ D'ENSEIGNEMENT SUPÉRIEUR

Syntaxe

```
print(*objects, sep=' ', end='\n', file=sys.stdout, flush=
False)
```

Paramètres

- **objects** objet(s) à l'imprimé. * indique qu'il peut y avoir plus d'un objet
- **sep** les objets sont séparés par sep. Valeur par défaut: ' ' (espace simple)
- **end** end est imprimé à la fin
- **file** doit être un objet avec la méthode write(string). S'il est omis, sys.stdout sera utilisé pour imprimer les objets à l'écran.
- **flush** si True, le flux est vidé de force. Valeur par défaut : False

Exp : `a = 5; print("a =", a, sep='->', end='\n')`

```
a =->5
```

Programmation de base : print()

```
import os
print('Bonjour, ' + os.getlogin() + '! Comment allez-vous ?')
```

```
Bonjour, utilisateur! Comment allez-vous ?
```

```
print('home', 'user', 'documents', sep='/')
```

```
home/user/documents
```

```
a = (4500 + 2575) / 14800
print("Le résultat est", a)
```

```
Le résultat est 0.4780405405405405
```

```
print(f"Le résultat est {a:.2f}")
```

```
Le résultat est 0.48
```

```
print(f"Le résultat est {a:.5f}")
```

```
Le résultat est 0.47804
```

Programmation de base : print()

```
a = 126; b = 59
print(f"Les valeurs sont {a:d} et {b:d}, leur rapport = {a/b:.2f}")
```

```
Les valeurs sont 126 et 59, leur rapport = 2.14
```

```
print(f"[{a:>6d}]") ; print(f"[{b:>6d}]")
```

```
[   126]
[    59]
```

```
print(f"[{a:<6d}]") ; print(f"[{b:<6d}]")
```

```
[126   ]
[59    ]
```

```
print(f"[{a:^6d}]") ; print(f"[{b:^6d}]")
```

```
[ 126  ]
[  59  ]
```

```
print(f"{a:*^6d}") ; print(f"{b:*^6d}")
```

```
*126**
**59**
```

Programmation de base : print()

```
print(f"{a:0>6d}") ; print(f"{b:0>6d}")
```

```
000126
000059
```

```
print(f"Diplôme {'BTS':>6s}") ; print(f"Formation {'SN':>4s}")
```

```
Diplôme    BTS
Formation   SN
```

```
print(f"{1.75:7.3f}"); print(f"{7/6:10.3f}")
```

```
  1.750
     1.167
```

```
print(f"Résultat : {4.1 * 6.7}")
```

```
Résultat : 27.47
```

```
print(f"Le minimum est {min(1, -2, 4)}")
```

```
Le minimum est -2
```

Programmation de base : print()

```
print(f"{1_000_000_000:e}"); print(f"{1000000000:e}")
```

```
1.000000e+09
1.000000e+09
```

```
print(f"{0.000_000_001:e}"); print(f"{0.000000001:e}")
```

```
1.000000e-09
1.000000e-09
```

```
x = 5.8264e3; y = 7.6e-2; print("x =",x,"\ty =",y)
```

```
x = 5826.4      y = 0.076
```

```
D = 12; nom = "Carnus"
print("Le Lycée {} est dans le département {}".format(nom, D))
```

```
Le Lycée Carnus est dans le département 12
```

```
a1 = 45; a2 = 12; a3 = (a1 + a2)/2
print("On a le chiffre {} et {} -> Moy = {:.3f}".format(a1, a2, a3))
```

```
On a le chiffre 45 et 12 -> Moy = 28.500
```

Programmation de base : print()

```
print("La résistance {0} a la valeur {1} Ohms. On a {1} pour {0}." .
                                    format("R_1", 120))
```

```
La résistance R_1 a la valeur 120 Ohms. On a 120 pour R_1.
```

```
A=12; B=35.6; C="abcd"; D=7.8251896
print("a=%d, b=%f, d=%.3f, c=%s et z=%f \"texte\" A=%d " %(A,B,D,C,4
                                    83,A))
```

```
a=12, b=35.600000, d=7.825, c=abcd et z=483.000000 "texte" A=12
```

```
print('-Carnus'*5)
```

```
-Carnus-Carnus-Carnus-Carnus-Carnus
```

```
print("""
Language:
\t1 Python
\t2 C\n\t3 JavaScript
""")
```

```
Language:
        1 Python
        2 C
        3 JavaScript
```

Programmation de base : input()

input() est une fonction intégrée utilisée pour lire une chaîne à partir d'une entrée standard (dans la plupart des cas, le clavier).

Syntaxe

```
x = input(prompt)
```

où prompt est une chaîne (facultative), qui est imprimée sur la sortie standard, avant que la fonction input() ne commence à lire la valeur entrée par l'utilisateur. prompt est utile pour permettre aux utilisateurs de savoir quel type d'entrée l'application attend.
La fonction input() renvoie la chaîne saisie par l'utilisateur dans l'entrée standard.

```
x = input('Saisir une valeur : ')
print('La valeur est : ', x)
type(x)
```

```
Saisir une valeur : 29
La valeur est :  29
str
```

Remarque : La valeur retournée par input() est de type "string"

Programmation de base : input()

CHARLES CARNUS
LYCÉE TECHNIQUE PRIVÉ D'ENSEIGNEMENT SUPÉRIEUR

Lire un nombre à l'aide de input()

Par défaut, la fonction input() renvoie une chaîne de caractères. Si on souhaite lire un nombre, on peut transtyper la chaîne en int, float ou complex, en utilisant respectivement les fonctions int(), float()et complex().

```
x = int(input('Entrer un entier : '))
print('La valeur saisie est : ', x)
y = float(input('Entrer un réel : '))
print('La valeur saisie est : ', y)
```

```
Entrer un entier : 29
La valeur saisie est :  29
Entrer un réel : 3.14
La valeur saisie est :  3.14
```

```
x = input()
print('Entrée saisie est : ', x)
```

```
87
Entrée saisie est :  87
```

Programmation de base : input()

```
nombre = int(input('Entrer un chiffre '))
```

```
Entrer un chiffre Carnus
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-117-c2ba7c51e0e0> in <module>
----> 1 nombre = int(input('Entrer un chiffre '))

ValueError: invalid literal for int() with base 10: 'Carnus'
```

```
try:
    nombre = int(input('Entrer un chiffre '))
except ValueError:
    pass
```

```
Entrer un chiffre Carnus
```

Programmation de base : Les listes

Python connaît différents types de données combinés, utilisés pour regrouper plusieurs valeurs. Le plus souple est la liste, qui peut être écrite comme une suite, placée entre crochets, de valeurs (éléments) séparées par des virgules. Les éléments d'une liste ne sont pas obligatoirement tous du même type, bien qu'à l'usage ce soit souvent le cas.

```
>>> squares = [1, 4, 9, 16, 25]
>>> squares
[1, 4, 9, 16, 25]
>>> squares[0]
1
>>> squares[-1]
25
>>> squares[-3:]
[9, 16, 25]
>>> squares[:]
[1, 4, 9, 16, 25]
>>> squares + [36, 49, 64, 81, 100]
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

Les listes sont muables : il est possible de modifier leur contenu.

```
>>> cubes = [1, 8, 27, 65, 125]
>>> cubes[3] = 64
>>> cubes
[1, 8, 27, 64, 125]
```

Programmation de base : Les opérateurs

CHARLES CARNUS
LYCÉE TECHNIQUE PRIVÉ D'ENSEIGNEMENT SUPÉRIEUR

Opérateurs mathématiques

Symbole	Opération	Types	Exemples
+	Addition	Entier, réel, chaîne de caractères	$3 + 2 \rightarrow 5$ $"a" + "b" \rightarrow "ab"$
–	Soustraction	Entier, réel	$3 - 2 \rightarrow 1$
*	Multiplicaion	Entier, réel, chaîne de caractères	$3 * 2 \rightarrow 5$ $3 * "b" \rightarrow "bbb"$
**	Puissance	Entier, réel	$3 ** 2 \rightarrow 9$
/	Division	Entier, réel	$3 / 2 \rightarrow 1.5$
//	Division entière (Euclidienne)	Entier, réel	$7 // 2 \rightarrow 3$
%	Modulo (reste de la division)	Entier, réel	$7 \% 2 \rightarrow 1$

Programmation de base : Les opérateurs

CHARLES CARNUS
LYCÉE TECHNIQUE PRIVÉ D'ENSEIGNEMENT SUPÉRIEUR

Opérateur	Signification
pow(x,y)	x à la puissance y. Par défaut : $0^0 = 1$
divmod(x,y)	Renvoie le couple $(x//y, x\%y)$
abs(x)	Valeur absolue d'un réel, module d'un complexe
int(x) float (x)	Conversion vers le type `int`, conversion vers le type `float`
round(x) round(x,n)	Arrondit à l'entier le plus proche, valeur approchée à 10^{-n} près
z.real, z.imag, z.conjugate()	Partie réelle, partie imaginaire, conjugué
complex(x,y), x+u*1j	Renvoie le nombre complexe $x + iy$

Programmation de base : Les opérateurs

CHARLES CARNUS
LYCÉE TECHNIQUE PRIVÉ D'ENSEIGNEMENT SUPÉRIEUR

Opérateurs logiques

Opérateur	Opération	Exemples
or	Ou logique	x or y → False x or z → True
and	Et logique	x and y → False x and z → True
not	Non logique	not y → False not z → True

Programmation de base : Les opérateurs

CHARLES CARNUS
LYCÉE TECHNIQUE PRIVÉ D'ENSEIGNEMENT SUPÉRIEUR

Opérateurs de comparaison

Opérateur	Opération
$<$	Strictement inférieur ($<$)
$<=$	Inférieur ou égal ($\leqslant$)
$>$	Strictement supérieur ($>$)
$>=$	Supérieur ou égal ($\geqslant$)
$==$	égal
$!=$	Différent ($\neq$)

Programmation de base : Les opérateurs

```
2+11*5
```

57

```
# Ecrire un commentaire
```

```
print('Bonjour')
```

Bonjour

```
print(5.6/2)
```

2.8

```
# Calcul simple.
print("2+3*5 donne:")
print(2+3*5)
```

2+3*5 donne:
17

Programmation de base : Les opérateurs

CHARLES CARNUS
LYCÉE TECHNIQUE PRIVÉ D'ENSEIGNEMENT SUPÉRIEUR

```
# Affectation
a=5
print(a)
```

5

```
a = = 4
```

False

```
x, y = (1, 2)  # ou
x, y = [1, 2]   # ou
(x, y) = (1, 2)
print(x)
print(y)
```

1
2

```
abc = "Deux mots"
print(abc)
```

Deux mots

```
a = 7
print("double de ", a, " est ", 2*
                    a)
```

double de 7 est 14

Programmation de base : Les boucles

CHARLES CARNUS
LYCÉE TECHNIQUE PRIVÉ D'ENSEIGNEMENT SUPÉRIEUR

Les structures de contrôles

Test if

```
#Test de parite

nb=3
if (nb%2)==0:
    print("Nombre pair")
else:
    print("Nombre impair")
```

Pas de switch

Boucle for

```
#boucle for allant de 0 a 9

for indice in range(10):
    #affichage
    print("indice=%d" % indice)
```

Boucle while

```
#Boucle while allant de 0 a 9

indice=0
while(indice<10):
    print("indice=%d" % indice)
    indice=indice+1
```

Programmation de base : Les boucles

CHARLES CARNUS
LYCÉE TECHNIQUE PRIVÉ D'ENSEIGNEMENT SUPÉRIEUR

Condition *if*, *elif*, *else*

Exemple 1 :

```
if a == 4:
    print('Fin')
```

Exemple 2 :

```
a = 5

if a == 1:
    print(1)
elif a == 2:
    print(2)
else:
    print('Fin')
```

Boucle *for*

Exemple 1 :

```
for i in range(5):
    print(i)
```

Exemple 2 :

```
for word in ('juin', 'juillet', 'août'):

    print("Un mois d'été : %s" % word)
```

Programmation de base : Les boucles

Boucle *while*, *break*, *continue*

Exemple 1 :

```
a = 2
while a < 100:
    a = a**2 + 1
print(a)
```

Exemple 2 :

```
a = 1.5
while a < 100:
    if a%3 == 0:
        break
    a = 2*a + 1
print(a)
```

Exemple 3 :

```
a = [0, 1, 2, 4, 10]
for element in a:
    if element == 0:
        continue
    print(1. / element)
```

Exemple 4 :

```
voyelles = 'aeiouy'
for i in 'carnus':
    if i in voyelles:
        print(i)
```

Exemple 5 :

```
[i**2 for i in range(4)]
```

Programmation de base : Les boucles

CHARLES CARNUS
LYCÉE TECHNIQUE PRIVÉ D'ENSEIGNEMENT SUPÉRIEUR

```
print("Entrer un nombre < 60")
nb = float(input())
if nb < 60:
    print("Ce nombre convient.")
else:
    print("Ce nombre est trop
                        grand.")
```

```
Entrer un nombre < 60
45
Ce nombre convient.
```

```
for x in range(5):
    if 2 <= x <= 3:
        print(x, "est dans
        l''intervalle")
    else:
        print(x, "est en dehors de
        l''intervalle")
```

```
0 est en dehors de l''intervalle
1 est en dehors de l''intervalle
2 est dans l''intervalle
3 est dans l''intervalle
4 est en dehors de l''intervalle
```

```
i = 1
while i <= 3:
    print(i)
    i = i + 1
```

```
1
2
3
```

```
for i in range(2):
    print("BTS")
```

```
BTS
BTS
```

```
for i in range(2, 4):
    print(i**2)
```

```
4
9
```

```
S = 0
for i in range(1, 5):
    S = S + i
print(S)
```

```
10
```

Programmation de base : Les boucles

```
for x in range(4):
    print(x)
else:
    print("Fin de la boucle")
```

```
0
1
2
3
Fin de la boucle
```

```
for x in range(2):
    for y in range(10,13):
        print(x, y)
```

```
0 10
0 11
0 12
1 10
1 11
1 12
```

```
a = [1.1,1.2,1.3,1.4]
for index, elem in enumerate(a):
    print(index,elem)
```

```
0 1.1
1 1.2
2 1.3
3 1.4
```

```
a = [1.1,1.2,1.3,1.4]
for index, elem in enumerate(a):
    print(f"Elément: {elem} a pour
                    indice : {
                    index}")
```

```
Elément: 1.1 a pour indice : 0
Elément: 1.2 a pour indice : 1
Elément: 1.3 a pour indice : 2
Elément: 1.4 a pour indice : 3
```

Programmation de base : Les boucles

CHARLES CARNUS
LYCÉE TECHNIQUE PRIVÉ D'ENSEIGNEMENT SUPÉRIEUR

```
for x in range(2, 10):
    if(x==7):
        break
    print(x)
```

```
2
3
4
5
6
```

```
for x in range(2, 7):
    if(x==4):
        continue
    print(x)
```

```
2
3
5
6
```

```
ma_liste = [nombre for nombre in range(1, 6)]
print(ma_liste)
```

```
[1, 2, 3, 4, 5]
```

```
ma_liste = ["C", "a", "r", "n", "u", "s"]
print([element for element in ma_liste])
```

```
['C', 'a', 'r', 'n', 'u', 's']
```

Programmation de base : Les boucles

CHARLES CARNUS
LYCÉE TECHNIQUE PRIVÉ D'ENSEIGNEMENT SUPÉRIEUR

```
pair = []
for nombre in range(1, 20):
    if nombre%2==0:
        pair.append(nombre)
print(pair)
```

```
[2, 4, 6, 8, 10, 12, 14, 16, 18]
```

```
pair =[]
for nombre in range(1, 20): nombre if nombre%2!=0 else pair.append(
                                    nombre)
print(pair)
```

```
[2, 4, 6, 8, 10, 12, 14, 16, 18]
```

```
for i in range(1, 8): i if i%2==0 else print(i**2)
```

```
1
9
25
49
```

Programmation de base : Les boucles

```
print([number**2 for number in range(1, 8) if number%2!=0])
```

```
[1, 9, 25, 49]
```

```
list1 = [1, 2, 3, 4 ]
list2 = [-1, -2, 3 , 2]
print([i*k for i in list1 for k in list2 if i%2==0 if k>0])
```

```
[6, 4, 12, 8]
```

```
from time import sleep

def progress(percent=0, width=30):
    left = width * percent // 100
    right = width - left
    print('\r[', '#' * left, ' ' * right, ']',
          f' {percent:.0f}%',
          sep='', end='', flush=True)

for i in range(101):
    progress(i)
    sleep(0.1)
```

```
[####################          ] 68%
[##############################] 100%
```

Programmation de base : Les fonctions

CHARLES CARNUS
LYCÉE TECHNIQUE PRIVÉ D'ENSEIGNEMENT SUPÉRIEUR

Exemple 1 :

```
x = 5
def plusCinq(y):
    return x + y

plusCinq(9)
```

Exemple 2 :

```
def triple(x):
    return 3 * x

triple(15)
```

Exemple 3 :

```
def triple1(x=2):
    return 3 * x

a=triple1()
print(a)

b=triple1(5)
print(b)
```

Exemple 4 :

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

def f(x):
    return x**2 + 10*np.sin(x)

x = np.arange(-8, 8, 0.1)
plt.plot(x, f(x))
```

Programmation de base : Les fonctions

```
import numpy as np

def g(x):
    return np.sqrt(x + 1)

x = 2

while x < 4:
    print("g(",x,") =",g(x))
    x = x + 0.5
```

```
g( 2 ) = 1.7320508075688772
g( 2.5 ) = 1.8708286933869707
g( 3.0 ) = 2.0
g( 3.5 ) = 2.1213203435596424
```

```
import numpy as np

def g(x):
    return np.sqrt(x + 1)

x = 2

while x < 4:
    print("g({:0.1f}) =
    {:0.2f}".format(x, g(x)))
    x = x + 0.5
```

```
g(2.0) = 1.73
g(2.5) = 1.87
g(3.0) = 2.00
g(3.5) = 2.12
```

Programmation de base : Les fonctions

CHARLES CARNUS
LYCÉE TECHNIQUE PRIVÉ D'ENSEIGNEMENT SUPÉRIEUR

```
# Créer la fonction switch
def switch(case):
    # Vérifier si l'argument correspond aux cas
    if case == "celeron":
        print ("Ce PC est aussi performant qu'une calculatrice!!!")

    elif case =="core i3":
        print ("Bonne chance avec ça ;)")

    elif case == "core i5":
        print ("Ouais, ça devrait aller.")

    elif case == "core i7":
        print ("Bien.")

    elif case == "core i9":
        print ("Très bien...")

    else:
        print ("Quoi ?")
# Demander l'entrée de l'utilisateur
cpuModel = str.lower(input("Veuillez entrer votre modèle de
                                    processeur (CPU) : "))
# Appel de la fonction
switch(cpuModel)
```

```
Veuillez entrer votre modèle de processeur (CPU) : CORE I9
Très bien...
```

Programmation de base : Les courbes

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

X = np.linspace(-2*np.pi, 2*np.pi, 256, endpoint=True)

C, S = 3*np.cos(X), 4*np.sin(X)

plt.figure()
plt.plot(X, C)
plt.plot(X, S)

plt.title('Courbes : sin(x) et cos(x)')
plt.xlabel('x')
plt.ylabel('Amplitude')

plt.grid()
plt.show()
```

Programmation de base : Réutilisation des scripts

Programme dans Jupyter NoteBook :

```
%run mon_programme.ipynb
```

Programme dans Spider

```
%run mon_programme.py
```

Programmation de base : Les nombres aléatoires

```
a = np.random.random(5)
print(a)
```

```
[0.16189084 0.41386536 0.12957121 0.40515615 0.39975044]
```

```
b = np.random.randint(0, 10, 5)
print(b)
```

```
[5 6 2 8 0]
```

```
c = np.random.uniform(-1.0, 1.0, 5)
print(c)
```

```
[-0.29661204  0.66944828 -0.91315794 -0.05392884  0.596433  ]
```

Un nombre complexe z peut s'écrire sous plusieurs formes.

$$z = a + \mathrm{j}b = M\cos(\theta) + \mathrm{j}M\sin(\theta) = M\Big(\cos(\theta) + \mathrm{j}\sin(\theta)\Big) = Me^{\mathrm{j}\theta}$$

Avec a la partie réelle (notée $\Re e$), b la partie imaginaire (notée $\Im m$), j le nombre imaginaire pur (noté i en mathématiques).
M le module de z et θ l'argument de z.
Le conjugué du nombre complexe z est noté $\overline{z}$, avec $\overline{z} = a - \mathrm{j}b = Me^{-\mathrm{j}\theta}$

Déclaration d'un nombre complexe sous Python

```
import numpy as np
import matplotlib.pyplot as plt
z = 5+3j
r = np.real(z)
i = np.imag(z)
print("Partie réelle =",r,", Partie imaginaire =",i)
m = np.abs(z)
p = np.angle(z)
print("Module de z =",m,", Argument de z =",p)
```

```
Partie réelle = 5.0 , Partie imaginaire = 3.0
Module de z = 5.830951894845301 , Argument de z = 0.5404195002705842
```

Programmation de base : Les matrices

Pour utiliser les matrices (tableaux) sous Python, on doit importer le module numpy.

```
import numpy as np
```

Création d'un vecteur (tableau 1D)

```
a = np.array([5, 9, 16])
```

Le type donné par cette instruction peut être généré par l'instruction :

```
type(a)
```

```
numpy.ndarray
```

Ce type appartient au package numpy alors que [5, 9, 16] est de type list.

```
type([5, 9, 16])
```

```
list
```

Programmation de base : Les matrices

CHARLES CARNUS
LYCÉE TECHNIQUE PRIVÉ D'ENSEIGNEMENT SUPÉRIEUR

Création d'un tableau (matrice)

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

```
array([[1, 2, 3],
       [4, 5, 6]])
```

Nombre d'éléments d'un tableau

```
a = np.array([5, 9, 16])
b = np.array([[1, 2, 3], [4, 5, 6]
                    ])
np.size(a)
```

```
3
```

```
np.size(b)
```

```
6
```

Taille d'un tableau

```
a = np.array([5, 9, 16, 18])
b = np.array([[1, 2, 3], [4, 5, 6]
                    ])
np.shape(a)
```

```
(4,)
```

```
np.shape(b)
```

```
(2, 3)
```

Produit terme à terme

```
a = np.array([[3, 2, 1], [2, 3, 1]
                    ])
b = np.array([[1, 2, 3], [4, 5, 6]
                    ])
a*b
```

```
array([[ 3,  4,  3],
       [ 8, 15,  6]])
```

CHARLES CARNUS
LYCÉE TECHNIQUE PRIVÉ D'ENSEIGNEMENT SUPÉRIEUR

Produit matriciel

```
a = np.array([[3, 2, 1], [2, 3, 1]
                    ])
b = np.array([1, 2, 3])
np.dot(a,b)
```

```
array([10, 11])
```

Transposée d'un tableau

```
a = np.array([[1, 2, 3], [4, 5, 6]
                    ])
a.T
```

```
array([[1, 4],
       [2, 5],
       [3, 6]])
```

Extraction d'une partie d'un tableau

```
a = np.array([11, 12, 13, 14, 15])
a[1:3]
```

```
array([12, 13])
```

```
a[1:]
```

```
array([12, 13, 14, 15])
```

```
a[:3]
```

```
array([11, 12, 13])
```

```
a[:]
```

```
array([11, 12, 13, 14, 15])
```

CHARLES CARNUS
LYCÉE TECHNIQUE PRIVÉ D'ENSEIGNEMENT SUPÉRIEUR

Extraction d'une partie d'un tableau (suite)

```
a = np.array([[1, 2, 3],
              [4, 5, 6]])
a[0,1]
```

```
2
```

```
a[:,1:3]
```

```
array([[2, 3],
       [5, 6]])
```

```
a[:1]
```

```
array([2, 5])
```

```
a[0,:]
```

```
array([1, 2, 3])
```

```
a[:,1:2]
```

```
array([[2],
       [5]])
```

Extraction d'une partie d'un tableau

```
>>> a[0, 3:5]
array([3, 4])

>>> a[4:, 4:]
array([[44, 55],
       [54, 55]])

>>> a[:, 2]
a([2, 12, 22, 32, 42, 52])

>>> a[2::2, ::2]
array([[20, 22, 24],
       [40, 42, 44]])
```

```
>>> a[(0,1,2,3,4), (1,2,3,4,5)]
array([1, 12, 23, 34, 45])

>>> a[3:, [0,2,5]]
array([[30, 32, 35],
       [40, 42, 45],
       [50, 52, 55]])

>>> mask = np.array([1,0,1,0,0,1], dtype=bool)
>>> a[mask, 2]
array([2, 22, 52])
```

Programmation de base : Les matrices

Matrice de 0

```
np.zeros((2,3))
```

```
array([[ 0.,  0.,  0.],
       [ 0.,  0.,  0.]])
```

Matrice de 1

```
np.ones((2,3))
```

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

Matrice identité

```
np.eye(3)
```

```
array([[ 1.,  0.,  0.],
       [ 0.,  1.,  0.],
       [ 0.,  0.,  1.]])
```

CHARLES CARNUS
LYCÉE TECHNIQUE PRIVÉ D'ENSEIGNEMENT SUPÉRIEUR

Construction de matrice à partir d'un vecteur

```
u = np.arange(1, 7)
u
```

```
array([1, 2, 3, 4, 5, 6])
```

```
np.shape(u)
```

```
(6,)
```

```
u.shape = (2, 3)
u
```

```
array([[1, 2, 3],
       [4, 5, 6]])
```

```
np.shape(u)
```

```
(2, 3)
```

Construction de matrice à partir d'un vecteur (2ème méthode)

```
a = np.arange(0,10).reshape(2,5)
print(a)
```

```
[[0 1 2 3 4]
 [5 6 7 8 9]]
```

Déclaration du type de données

```
a = np.array([[1,2],[4,7]],dtype=float)
print(a.dtype)
```

```
float64
```

Déclaration d'une matrice avec des éléments identiques

```
a = np.full(shape=(2,4),fill_value=0.1)
print(a)
```

```
[[0.1 0.1 0.1 0.1]
 [0.1 0.1 0.1 0.1]]
```

Programmation de base : Les matrices

CHARLES CARNUS
LYCÉE TECHNIQUE PRIVÉ D'ENSEIGNEMENT SUPÉRIEUR

Passage d'un type list à une matrice

```
lst= [1.2,3.1,4.5,6.3]
a = np.asarray(lst,dtype=float).\
reshape(2,2)
print(a)
```

```
[[1.2  3.1]
 [4.5  6.3]]
```

Ajout d'une ligne (concaténation de matrices)

```
b = np.array([[4.1,2.6]])
c = np.append(a,b,axis=0)
print(c)
```

```
[[1.2  3.1]
 [4.5  6.3]
 [4.1  2.6]]
```

Ajout d'une colonne à une matrice

```
d = np.array([[7.8],[6.1],[5.4]])
print(np.append(c,d,axis=1))
```

```
[[1.2  3.1  7.8]
 [4.5  6.3  6.1]
 [4.1  2.6  5.4]]
```

Maximum de chaque colonne

```
v=([[1, 2],[5, 4],[3, 6]])
print(np.max(v,axis=0))
```

```
[5  6]
```

Maximum de chaque ligne

```
print(np.max(v,axis=1))
```

```
[2  5  6]
```

CHARLES CARNUS
LYCÉE TECHNIQUE PRIVÉ D'ENSEIGNEMENT SUPÉRIEUR

Déterminant d'une matrice

```
from numpy.linalg import det
a = np.array([[1, 2],
              [3, 4]])
det(a)
```

```
-2.0
```

Inverse d'une matrice

```
from numpy.linalg import inv
a = np.array([[1, 3, 3],
              [1, 4, 3],
              [1, 3, 4]])
inv(a)
```

```
array([[ 7., -3., -3.],
       [-1.,  1.,  0.],
       [-1.,  0.,  1.]])
```

Résolution d'un système d'équations linéaires

```
a = np.array([[3,1], [1,2]])
b = np.array([9,8])
x = np.linalg.solve(a, b)
x
```

```
array([ 2.,  3.])
```

Vérification des résultats :

```
np.allclose(np.dot(a, x), b)
```

```
True
```

Programmation avancée :

Résolution d'équations différentielles

Résolution de l'équation différentielle $\frac{\mathrm{d}y}{\mathrm{d}t} + y = 0$ dans l'intervalle $t \in [0, 5]$, avec la condition initiale : $y(t = 0) = 1$.

```
import numpy as np
from scipy.integrate import odeint
from matplotlib import pyplot as plt

%matplotlib inline

def calc_derivative(ypos, time):
    return -ypos

time_vec = np.linspace(0, 5, 50)
yvec = odeint(calc_derivative, 1, time_vec)

plt.figure(figsize=(4, 3))
plt.plot(time_vec, yvec)
plt.xlabel('t: Temps')
plt.ylabel('y: Amplitude')
plt.tight_layout()
```

Programmation avancée :

Recherche des minimums et des zéros d'une fonction

$f(x) = x^2 + 5\sin(x)$ dans l'intervalle $t \in [-5, 5]$.

```
import numpy as np
from scipy import optimize
import matplotlib.pyplot as plt
%matplotlib inline

# Définition de la fonction
x = np.arange(-5, 5, 0.1)
def f(x):
    return x**2 + 5*np.sin(x)

# Recherche du minimum
grid = (-5, 5, 0.1)
xmin= optimize.brute(f, (grid, ))
print("Coordonnées du minimum sont : %s" % xmin , f(xmin) )

# Calcul des zéros de la fonction
z1 = optimize.root(f, 1) # Valeur initiale est 1
print("La 1ere solution trouvée est : %s" % z1.x)
z2 = optimize.root(f, -2.5) # Valeur initiale est -2.5
print("La 2nd solution trouvée est : %s" % z2.x)
```

Recherche des minimums et des zéros d'une fonction (suite)

```
# Courbes
fig = plt.figure(figsize=(6, 4))
ax = fig.add_subplot(111)

# Courbe de la fonction
ax.plot(x, f(x), 'b-', label="f(x)")

# Le minimum
xmins = np.array([xmin[0]])
ax.plot(xmins, f(xmins), 'go', label="Minimum")

# Les zéros
roots = np.array([z1.x, z2.x])
ax.plot(roots, f(roots), 'kv', label="Zéros")

# Décoration de la figure
ax.legend(loc='best')
ax.set_xlabel('x')
ax.set_ylabel('f(x)')
ax.axhline(0, color='gray')
plt.show()
```

```
Coordonnées du minimum sont :
[-1.11052734] [-3.24639427]
La 1ere solution trouvée est :
[0.]
La 2nd solution trouvée est :
[-2.08593458]
```

Programmation avancée : Les polynômes

Soit le polynôme $p(x) = 3x^2 + 2x - 1$.

```
# Déclaration du polynôme
p = np.poly1d([3, 2, -1])

# Calcul de p(0)
p0=p(0)
print("La valeur de p(0) est : ", p0)

# Calcul des racines de p(x)=0
s=p.roots
print("Les racines de p(x) sont : ", s)

# Calcul de l'ordre de p(x)
o=p.order
print("ordre de p(x) est : ", o)
```

```
La valeur de p(0) est :  -1

Les racines de p(x) sont :
[-1.          0.33333333]

ordre de p(x) est :  2
```

Programmation avancée : Manipulation de fichiers

CHARLES CARNUS
LYCÉE TECHNIQUE PRIVÉ D'ENSEIGNEMENT SUPÉRIEUR

```
# Déclaration du nom de fichier à créer et son extension
NomFichier = 'test.txt'

# Création et ouverture du fichier test.txt en mode write 'w'
# (écriture). Si le fichier test.txt existe déjà, il est écrasé
Fichier = open(NomFichier,'w')

# écriture dans le fichier avec la méthode write()
Fichier.write('BTS SN')

# Fermeture du fichier avec la méthode close()
Fichier.close()
```

Programmation avancée : Manipulation de fichiers

CHARLES CARNUS
LYCÉE TECHNIQUE PRIVÉ D'ENSEIGNEMENT SUPÉRIEUR

```
# Ouverture du fichier test.txt en mode append 'a' (ajout)
Fichier = open('test.txt','a')
Fichier.write('\nUne deuxième ligne.\n')  # '\n' saut de ligne
Fichier.write('abc\tABC\t123\n')          # '\t' tabulation
Fichier.write(str(36.4)+'\n')   # str() transforme un nombre
                                # en chaîne de caractères
Fichier.close()
```

```
BTS SN
Une deuxième ligne.
abc     ABC     123
36.4
```

```
# Ouverture du fichier test.txt en mode read 'r'
#(lecture en mode texte)
Fichier = open('test.txt','r')
chaine = Fichier.read() # Lecture avec la méthode read()
print('Contenu du fichier :\n' + chaine)# Affichage du contenu du
                                        # fichier
Fichier.close()
```

```
Contenu du fichier :
BTS SN
Une deuxième ligne.
abc     ABC     123
36.4
```

Programmation avancée : Manipulation de fichiers

CHARLES CARNUS
LYCÉE TECHNIQUE PRIVÉ D'ENSEIGNEMENT SUPÉRIEUR

```
obFichier = open('Monfichier','a')
obFichier.write('Lycée Charles Carnus\n')
obFichier.write("Rodez\n")
obFichier.close()
```

```
ofi = open('Monfichier', 'r')
t = ofi.read()
print(t)
```

```
Lycée Charles Carnus
Rodez
```

Programmation avancée : Manipulation de fichiers

Soit le fichier "Monfichier" qui contient ce code :

```
# Importation
%matplotlib inline
from scipy import signal
wc=2500
# Calcul
T=signal.TransferFunction([0.8],[1/(wc*wc),0.3/wc,1])
```

On veut supprimer toutes les lignes de commentaires (voir la fonction ci-dessous)

```
def supp_com(source,destination):
    fs = open(source, 'r')
    fd = open(destination, 'w')
    while 1:
        txt = fs.readline()
        if txt =='':
            break
        if txt[0] != '#':
            fd.write(txt)
    fs.close()
    fd.close()
    return
```

On enregistre le fichier modifié dans "Monfichier_f" :

```
supp_com('Monfichier', 'Monfichier_f')
```

On ouvre le fichier "Monfichier_f"

```
ofi = open('Monfichier_f', 'r')
t = ofi.read()
print(t)

-----------------------------------------------

%matplotlib inline
from scipy import signal
wc=2500
T=signal.TransferFunction([0.8],[1/(wc*wc),0.3/wc,1])
```

Programmation avancée : Manipulation de fichiers

Ouverture d'un dossier compressé "Monfichier.zip" :

```
import zipfile
with zipfile.ZipFile ("Monfichier.zip", "r") as fz:
    for info in fz.infolist () :
        print(info.filename, info.date_time, info.file_size)
```

```
Monfichier (2019, 10, 6, 16, 48, 26) 133
```

Ouvertue du fichier "Monfichier"

```
import zipfile
with zipfile.ZipFile ("Monfichier.zip", "r") as fz:
    data = fz.read ("Monfichier")
print(data)
```

```
b'# Importation \n%matplotlib inline \nfrom scipy import signal \nwc=2500
\n# Calcul \nT=signal.TransferFunction([0.8],[1/(wc*wc),0.3/wc,1])'
```

Enregistrement des données dans un fichier CSV

```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(0, 10, 11)
print("Les valeurs de x sont : \n x = ",x)
y = x**2
print("Les valeurs de y sont : \n y = ",y)

plt.figure()
plt.plot(x, y)
plt.xlabel('Abscisses')
plt.ylabel('Ordonnées')
plt.title('Courbe y=f(x)')
plt.grid(True)
plt.show()

data=np.matrix([x,y]).T
print("La matrice de données est : \n",data)
print("Taille de la matrice : ",data.shape)
np.savetxt("test_KB.csv", data,delimiter=",", header="x, y",
  comments="")
```

Résultats

```
Les valeurs de x sont :
 x = [ 0.  1.  2.  3.  4.  5.  6.  7.  8.  9. 10.]
Les valeurs de y sont :
 y = [  0.   1.   4.   9.  16.  25.  36.  49.  64.  81. 100.]
```

```
La matrice de données est :
 [[  0.   0.]
 [  1.   1.]
 [  2.   4.]
 [  3.   9.]
 [  4.  16.]
 [  5.  25.]
 [  6.  36.]
 [  7.  49.]
 [  8.  64.]
 [  9.  81.]
 [ 10. 100.]]
Taille de la matrice :  (11, 2)
```