

# Aide-mémoire, PYTHON

---

## Table des matières

Aide-mémoire, PYTHON.....	1
Liens.....	2
En cas de fermeture de la console .....	2
Généralités.....	2
Aide sur les commandes .....	2
Raccourcis clavier .....	2
Gestion des erreurs – Mode pas à pas .....	2
Les imports.....	3
Affectation, calculs .....	3
Entrées/Sorties.....	3
Les fonctions .....	3
Tests.....	4
Boucles.....	4
Simulation – Tirage aléatoire .....	4
Pour faire des maths.....	4
Listes .....	5
Les compréhensions de listes.....	5
Les tuples. ....	5
Les dictionnaires.....	5
Les ensembles : set.....	6
Graphisme avec matplotlib.....	6
Le module turtle : pour dessiner .....	7
Le module sympy : le calcul formel .....	7
Le module Numpy : manipulation de matrices et de tableaux.....	8
La lecture de fichiers.....	8
La bibliothèque csv pour le travail sur les fichiers csv .....	8
La bibliothèque pandas : manipulation des données.....	8
La bibliothèque networkX : étude des graphes et des réseaux. ....	9
La bibliothèque PIL : traitement d’images. ....	10
Autres bibliothèques .....	10
Annexes :.....	11
PIP.....	11
JUPYTER.....	11
Utiliser jupyter depuis EduPython.....	11

## Liens

<http://www.python-simple.com/python-pandas/dataframes-indexation.php>

<http://apprendre-python.com/page-apprendre-dictionnaire-python>

[http://www.isn-ozanam.fr/index\\_ISN.php](http://www.isn-ozanam.fr/index_ISN.php) MDP : ozanam\_2018-19

<https://openclassrooms.com/fr/courses/235344-apprenez-a-programmer-en-python/>

## En cas de fermeture de la console

Si un élève ferme par mégarde la console, on peut la rouvrir en suivant le chemin :

Affichage – Fenêtre de l'IDE – Console interactive

## Généralités

Les commentaires sont précédés du symbole #	<code># ceci est un commentaire</code>
<b>Indentation</b> : PYTHON fonctionne par blocs écrits avec un décalage. Un bloc est annoncé par le symbole :	<pre>if x &gt; 0 :     y = x else :     y = -x</pre>


## Aide sur les commandes

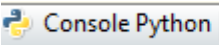
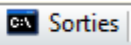
Avoir de l'aide sur une fonction	<code>help(fonction)</code> Exemple : <code>help(sqrt)</code>
Les docstrings permettent d'écrire une aide sur les fonctions. Cette aide s'affiche lorsque l'on tape la fonction.	<pre>def exemple(x) :     "ceci est texte qui s'affichera"     Corps de la fonction</pre>


## Raccourcis clavier

Aller à l'étape suivante dans le mode pas à pas	F7
Reprendre des commandes saisies plus « haut » dans la console	Flèche haut
Compléter une commande	La tabulation
Récupérer une valeur	Le _ (underscore)

## Gestion des erreurs – Mode pas à pas

On peut effectuer le programme étape par étape grâce à la commande  (« aller à l'étape suivante » - raccourci clavier : F7) et visualiser l'état de chaque variable en se plaçant dans l'onglet « variables » (en bas à gauche de l'écran

dans la configuration classique) :   

On sort ensuite du mode pas à pas grâce à la commande  (stopper le débogage).

Lorsqu'un programme rencontre une erreur on peut « revenir en arrière » (Menu Outil **Analyse rétrospective**) afin de voir ce qui se passe, en utilisant la console pour faire des tests, l'onglet variable ou même en laissant le curseur sur le code pour voir la valeur d'une variable.

## Les imports

# -*- coding: utf-8 -*- from math import * from random import * import matplotlib.pyplot as plt import numpy as np	
--	--

## Affectation, calculs

Affecter à la variable $a$ la valeur 5	$a=5$
Affectation multiple	$xA,yA=2.0, 3.0$
Calculer une puissance $a^n$	$a ** n$
Quotient dans la division euclidienne de $a$ par $b$	$a//b$
Reste dans la division euclidienne de $a$ par $b$	$a\%b$

Pour certains calculs, il est nécessaire d'importer un module spécifique. Par exemple, pour importer le module `maths`, on doit taper dans la zone de saisie du programme : `from math import *`

On a alors accès aux instructions suivantes :

Racine carrée de $a$	<code>sqrt(a)</code>
Partie entière de $a$	<code>floor(a)</code>
Arrondir un nombre à $i$ chiffre près	<code>round(x,i)</code>
Arrondir par excès à l'entier	<code>ceil(nombre)</code> (avec le module <code>math</code> )

## Entrées/Sorties

Saisir dans la console	On appelle directement les fonctions dans la console	<code>sin(2.3)</code> <code>mon_milieu(1.0,2.0)</code>
Afficher dans la console	la valeur de la variable $a$	<code>print(a)</code>
	la lettre $a$	<code>print("a")</code>
	un mélange texte et valeur	<code>print("la valeur de a est ", a)</code> ou mieux <code>print("On trouve a: %f et b: %f"%(a, b))</code> avec le formatage : <code>print(f "On trouve {a :.2f} et {b :2f} ")</code> pour un affichage à deux décimales
Définir une fonction $f$ de paramètres $a, b, c \dots$ renvoyant une valeur $y$ .		Exemple : <code>def f(x):</code> <code>    return 3 * x + 1</code>

## Les fonctions

La commande <code>assert</code> permet de tester une condition avant le corps de la fonction	<code>Assert test</code> Exemple ( <code>assert a !=0</code> )
La fonction <code>lambda</code> . Elle évite de définir une fonction par la commande <code>def</code>	<code>f = lambda x : x**2+1</code> Exemple pour $f(x) = x^2 + 1$
Commenter ses fonctions avec <code>docstrings</code> . On accède au teste par la commande <code>help(NomDeLaFonction)</code>	<code>def delta(a,b,c):</code> "Renvoie le discriminant d'un polynôme du second degré"

	<code>return b**2-4*a*c</code>
Variable globale dans une fonction.	Spécifier dans le corps de la fonction : <code>global a</code> (par exemple)
Passage d'arguments. Python offre une souplesse dans le passage d'arguments  Dans l'exemple on peut appeler <code>delta(b=3)</code> ; <code>delta()</code> , etc.	<code>def delta(a=1,b=1,c=1):</code> "Renvoie le discriminant d'un polynôme du second degré" <code>return b**2-4*a*c</code>

## Tests

Si le reste dans la division euclidienne de $a$ par 2 est nul Alors $a$ est pair Sinon $a$ est impair	<code>def est_pair(a):</code> <code>if a%2==0:</code> <code>return True</code> <code>else:</code> <code>return False</code>
Tester l'égalité $a = b$	<code>a == b</code>
Tester $a \neq b$	<code>a != b</code>
Tester $a \leq b$	<code>a &lt;= b</code>
Sinon si (contraction de else et de if)	<code>elif condition :</code>

Le « alors » se traduit par deux points et un décalage des instructions. L'indentation est automatique lorsque l'on tape sur la touche entrée après les deux points.

Le « else » traduisant « sinon » est au même niveau que le « if ».

## Boucles

Tant que $a < 10$	<code>while a &lt; 10 :</code> <code>  bloc d'instructions indenté</code>
Pour $i$ allant de 1 à $n$ ( $i$ prend les valeurs entières de l'intervalle $[1 ; n + 1[$ )	<code>for i in range(1,n+1) :</code> <code>  bloc d'instructions indenté</code>
$i$ prend tour à tour les valeurs entières de 0 à 4 (intervalle $[0 ; 5[$ )	<code>for i in range(5) :</code>
$i$ prend les valeurs entières de l'intervalle $[3 ; 11[$ avec un pas de 2	<code>for i in range(3,11,2) :</code>

## Simulation – Tirage aléatoire

Il faut penser à importer le module random en tapant dans le script : `from random import *`

Renvoyer un entier aléatoire entre $a$ et $b$	<code>randint(a,b)</code>
Renvoyer un nombre aléatoire entre 0 et 1	<code>random()</code>
Renvoyer un nombre décimal aléatoire entre $a$ et $b$	<code>uniform(a,b)</code>

## Pour faire des maths

Il existe de nombreuses bibliothèques pour faire des maths.

La plus utile est math (cmath si on veut travailler avec des complexes)

Il existe des modules pour travailler sur les fractions, les nombres décimaux, etc

Définir un complexe	<code>A=complex(3,4)</code> ou <code>a=3+4*1j</code>
Partie réel, partie imaginaire	<code>a.real</code> <code>a.imag</code>
Importer cmath pour calculer module, argument, etc	<code>From cmath import *</code>
Module fractions	<code>From fractions import Fraction</code>

Définir une fraction	A=Fraction(4,5)
Module decimal	From decimal import Decimal
Définir un nombre décimal	A=decimal('0.1')

## Listes

Les listes sont des objets qui peuvent en contenir d'autres. La liste peut être modifiée.

Créer une liste vide L	L=[]
Créer une liste de nombres L	L=[1,3,7,8]
Ajouter un élément à la fin de la liste L	L.append(élément)
Retirer un élément	L.remove(élément)
Indiquer le nombre d'éléments d'une liste L	len(L)
Trier les éléments	L.sort()
Compter le nombre d'apparitions d'un élément	n=L.count(élément)
Tirer au sort un élément d'une liste	n=choice(L)
Savoir si un élément n est présent	if n in L
Trouver l'index d'un élément	L.index(élément)
Minimum d'une liste	min(liste)
Maximum d'une liste	max(liste)
Retirer un élément à partir de son index	del L[0] (retire le premier élément de la liste l)
Retirer le dernier élément	del L(-1)
Effectuer la somme des éléments d'une liste	sum(L)

Attention, l'indice du premier élément d'une liste est 0. Ainsi, L[n] renverra le (n + 1)<sup>ème</sup> élément de la liste.

## Les compréhensions de listes

On peut définir les éléments d'une liste directement dans la liste.

Créer une liste 10 000 entiers aléatoires entre 1 et 6	L1=[randint(1,6) for i in range(10000)]
Créer la liste des 100 premiers entiers naturels	L2=[i for i in range(100)]
Créer la liste des 10 premiers carrés	L3=[i**2 for i in range(100)]
Créer la liste des carrés des éléments de la liste L2	L3=[i**2 for i in L2]
Créer la liste des valeurs d'une fonction à partir d'une liste avec une condition.	L2=[f(i) for i in L1 if i != 4]

## Les tuples.

Un **tuple** est une **liste** qui ne peut plus être modifiée.

Créer un tuple vide	A=()
Créer un tuple à un élément	A=(1,) ou A=1,

## Les dictionnaires

Un **dictionnaire** en **python** est une sorte de **liste** mais au lieu d'utiliser des **index**, on utilise des **clés**, c'est à dire des valeurs autres que numériques.

Exemple Dic1={"name": "Olivier", "age": 30}={clé:valeur}

Initialisation d'un dictionnaire	Dic1={}
Ajouter une valeur (indiquer une clé et une valeur)	Dic1[clé]=valeur
Obtenir une valeur dans un dictionnaire	Dic1.get("name") renvoi Olivier
Supprimer une entrée	del Dic1[clé]
Lister les clés	Dic1.keys()
Lister les valeurs	Dic1.values()
Lister les clés et les valeurs en même temps	Dic1.items()

## Les ensembles : set

Un ensemble est une collection non ordonnée d'objets uniques et immuables (en profondeur).

Les ensembles peuvent être utiles en maths pour les notions de réunion, d'intersection, d'appartenance.

Initialisation d'un ensemble Ne pas utiliser d'ensemble vide car PYTHON considère que l'on déclare un dictionnaire.	A={'1',3,'Robert'}
Intersection de deux set	A & B
Réunion de deux set	A   B

## Graphisme avec matplotlib

Pour afficher les graphiques	Chaque graphisme devra se terminer par : plt.show()
Affichage des axes	plt.axis([0,6,0,20]) Axe des abscisses puis ordonnées.
Créer une liste de 15 valeurs entre -2 et 2 Représenter une fonction	lx=np.linspace(-2,2,15) plt.plot(lx,f(lx)) Pour changer de couleur : rouge : plt.plot(x,f,'r') en rouge plt.plot(x,f,'b') en bleu
Représenter un nuage de points	On dispose de deux listes, la première l1 contenant les abscisses et la seconde l2 les ordonnées plt.plot(l1,l2,'o') 'o' impose une représentation par des disques '-' impose une représentation par des tirets '^' impose une représentation par des triangles Pour changer de couleur : rouge : plt.plot(l1,l2,'ro') bleu : plt.plot(l1,l2,'bo') triangle rouge : plt.plot(l1,l2,'r^')
Faire une pause pour animer	plt.pause(1) Pour une pause de 1 seconde :
Indiquer une légende sur l'axe des abscisses	plt.xlabel('image de la fonction')
Indiquer une légende sur l'axe des ordonnées	plt.ylabel('image de la fonction')

Indiquer une légende sur le graphique (par exemple pour identification, dans le cas de plusieurs courbes)	<code>plt.plot(lx,f(lx),label="f(x) ")</code> <code>plt.legend( )</code> → cette commande permet l'affichage de la légende
Pour modifier la taille des objets	<code>lines=plt.plot(l1,l2)</code> <code>plt.setp(lines,color='r',linewidth=2)</code>
Afficher la grille	<code>plt.grid( )</code>
Histogramme	Avec 10 barres par défaut : <code>plt.hist(l)</code> Pour contrôler le nombre de barres <code>plt.hist(l,bins=50)</code> Pour normer les valeurs : <code>plt.hist(l,bins=50,normed=True)</code>
Exemple pour la simulation	<code>plt.hist([randint(1,6) for i in range(100)],bins=6,range=(1,6),normed=True)</code>
Diagramme en barres / bâtons	<code>plt.bar(L1, L2, 0.1, color='r')</code> : pour une liste L1 en abscisses, une liste L2 en ordonnée et des barres de largeur 0,1

## [Le module turtle : pour dessiner](#)

Ne fonctionne sous Jupyter.

Importation de la librairie	<code>from turtle import *</code>
Sortie de la fenêtre	<code>exitonclick()</code>
Dessiner	<code>down()</code>
Se déplacer sans dessiner	<code>up()</code>
Aller à	<code>goto(x,y)</code>

## [Le module sympy : le calcul formel](#)

Importation de la librairie	<code>from sympy import *</code>
Déclarer une variable	<code>x=symbol('x')</code> ou <code>Symbol('x')</code>
Dériver une fonction	<code>expr=x**2</code> <code>diff(expr,x)</code>
Obtenir un développement en 0 à l'ordre 8	<code>series(expr,x,0,8)</code>
Résoudre une équation dans C	<code>solve(x**2+x+1,x)</code>
Développer une expression	<code>expand((x+5)**2)</code>
Factoriser une expression	<code>factor(x**2-1)</code>
Limite en +infini	<code>limit(1/(x+1),x,oo)</code>
Limite en -infini	<code>limit(1/(x+1),x,oo)</code>
Limite en 0	<code>limit(1/(x+1),x,0)</code>
Limite en 0+	<code>limit(1/(x+1),x,0,'+')</code>

Primitive	<code>integrate(1/x,x)</code>
Intégrale	<code>integrate(1/x,(x,1,E))</code>
Résoudre $2x+y=2$ et $-3x-5y=1$	<code>x,y=symbols('x y')</code> <code>solve([2x+y-2,-3x-5y-1],[x,y])</code>

## [Le module Numpy : manipulation de matrices et de tableaux](#)

NumPy est destinée à manipuler des matrices ou tableaux multidimensionnels ainsi que des fonctions mathématiques opérant sur ces tableaux.

Importation de la librairie	<code>import numpy as np</code>
<code>M=np.array()</code>	Tableau d'éléments <code>M=np.array([3,4,6])</code> Matrice : <code>M=np.array([[4,2],[3,4]])</code> Matrice 2,2
<code>dot(a,b)</code>	Produit matriciel

## [La lecture de fichiers](#)

Ouvrir un fichier en mode lecture	<code>fichier = open("texte.txt", "r")</code> <code>r= read</code> <code>w=write</code> <code>a= se positionne à la fin du fichier</code>
<code>texte=fichier.read()</code>	Lit le fichier et créer une liste d'éléments
<code>split()</code>  Renvoie une liste.	Permet de transformer une chaîne en liste. Exemple : <code>liste=texte.split('\n')</code> \n est le délimiteur \n fin de ligne <code>Texte.split(',')</code> pour un fichier csv
<code>f.close()</code>	Ferme le fichier
<code>f.readlines()</code>	Lit les lignes du fichier

## [La bibliothèque csv pour le travail sur les fichiers csv.](#)

Importer la bibliothèque csv	<code>Import csv</code>
Ouvrir un fichier csv en se positionnant à la fin	<code>fichier = open("FichierErreurs.csv", 'a', newline="")</code>
Déclarer un fichier csv	<code>FichierCSV= csv.writer(fichier,delimiter=",")</code>
Ecrire à la fin du fichier la liste	<code>FichierCSV.writerow(Liste)</code>

## [La bibliothèque pandas : manipulation des données.](#)

Bibliothèque permettant la manipulation et l'analyse des données.

Importer la bibliothèque Pandas sous l'alias pd	<code>import pandas as pd</code>
Lit le fichier 'seconde_B.xlsx'	<code>df = pd.read_excel('seconde_B.xlsx')</code>
Lit les cinq premiers enregistrements	<code>df.head()</code>
Lit la ligne nb	<code>df.iloc[nb]</code>
Donne le nombre de lignes	<code>len(df)</code>
Donne la première ligne	<code>df.columns</code>
Calcule des indicateurs statistiques pour chaque colonne	<code>df.describe()</code>



## La bibliothèque networkX : étude des graphes et des réseaux.

Bibliothèque permettant le travail sur les graphes et les réseaux

Importer la bibliothèque networkx sous l'alias nx La bibliothèque est dépendante de matplotlib et de numpy	<code>import networkx as nx</code>
Créer un graphe	<code>g = nx.Graph() # DiGraph() pour un graphe orienté</code>
Sommets du graphe	<code>g.nodes()</code>
Ajouter des liaisons dans le graphe.	<code>g.add_edges_from()</code>  Exemples :  <code>g.add_nodes_from(range(5))</code>  <code>g.add_edges_from([(0,1),(3,4),(4,2),(1,3),(4,0),(1,2)])</code>
Degrés des sommets du graphe g	<code>g.degree()</code>
Nombre de sommets du graphe g	<code>g.number_of_nodes()</code>
Nombre d'arcs du graphe g	<code>g.number_of_edges()</code>
Liste des prédécesseurs du sommet i	<code>g.predecessors(i)</code>
Liste des successeurs du sommet i	<code>g.successors(i)</code>
Liste des voisins du sommet i	<code>g.neighbors(i)</code>
Pour dessiner un graphe	<code>plt.figure()</code> <code>nx.draw(g)</code> <code>plt.show()</code>
Construire la matrice d'adjacence	<code>nx.to_numpy_matrix(g)</code>
Vérifier l'existence d'un chemin	<code>nx.has_path(g,nodeA,nodeB)</code>
Tester la connexité	<code>nx.is_connected(g)</code>
Tester la caractère Eulérien	<code>nx.is_eulerian(g)</code>

## La bibliothèque PIL : traitement d'images.

Bibliothèque permettant l'ouverture des images dans plusieurs formats standards et leur traitement.

Importer la sous-bibliothèque Image, dans laquelle se trouve l'essentiel	<code>from PIL import Image</code>
Ouverture d'une image	<code>image=Image.open("fichier.ext")</code>
Sauvegarder une image	<code>image.save("fichier.ext")</code>
Créer une image semblable mais avec d'autres dimensions.	<code>image_nouvelle=image.resize(largeur, hauteur)</code>
Renvoyer la dimension d'une image	<code>L,H=image.size</code>
Afficher une image	<code>image.show()</code>
Appliquer un filtre <i>Pour l'utilisation des filtres, importer la sous-bibliothèque ImageFilter (from PIL import ImageFilter)</i>	<code>image_nouvelle=image.filter(ImageFilter.FILTRE)</code> -Exemples de ce que l'on peut tester, à placer à la place de FILTRE : BLUR, DETAIL, CONTOUR, EDGE_ENHANCE, EDGE_ENHANCE_MORE, EMBOSS, FIND_EDGES, SMOOTH, SMOOTH_MORE, SHARPEN.
Conversion de MODE : On travaille le plus souvent avec des images en mode RGB, parfois RGBA (pour ajouter la transparence) mais aussi en mode noir et blanc ou en niveaux de gris.	<code>image_nouvelle=image.convert("MODE")</code> -Exemples de MODE : 1 (pour le noir et blanc), L (pour niveaux de gris), RGB, RGBA.
Renvoie les trois composantes couleurs de chaque pixel	<code>R,V,B=image.split()</code>
Créer une image à partir des trois composantes	<code>Image_nouvelle=Image.merge("RGB",(R,V,B))</code>
Créer le tableau des pixels	<code>p=image.load()</code>
Créer la liste des pixels	<code>L=liste(image.getdata())</code>
Reconstruit une image à partir de la liste	<code>im.putdata(L)</code>
Renvoie les trois composantes du pixel désigné	<code>rouge,vert,bleu=image.getpixel((i,j))</code>
Affecte au pixel désigné les trois composantes données	<code>image_nouvelle.putpixel((i,j),(rouge,vert,bleu))</code> (rouge, vert et bleu prennent toute valeur entière de 0 à 255.)
Mélange de deux images	<code>image_nouvelle=Image.blend(image1,image2,alpha)</code>
Appliquer des transformations <i>(on peut utiliser ces transformations prédéfinies mais il est intéressant pédagogiquement d'en construire une bonne partie avec les élèves).</i>	<code>image_nouvelle=</code> <code>image.transpose(Image.TRANSFORMATION)</code> -Exemples de transformations : FLIP_LEFT_RIGHT, FLIP_TOP_BOTTOM, ROTATE_180.

## Autres bibliothèques

Bibliothèque Tkinter Interface graphique avec gestion d'évènements	<code>from tkinter import *</code>
Bibliothèque openCV Traitement d'images	<code>import cv2</code>

Bibliothèque PIL Traitement d'images	from PIL import Image
Bibliothèque pour accéder à internet	import urllib.request
Bibliothèque Pygame Interface graphique avec gestion d'évènements	import pygame from pygame.locals import *
Bibliothèque NetworkX Pour l'étude des graphes et des réseaux	import networkx as nx import numpy as np import matplotlib.pyplot as plt
Bibliothèque GPIO pour la gestion d'un Raspberry	import RPi.GPIO as GPIO
Bibliothèque pour l'utilisation des réseaux scapy Installation de la version 2.4.0 Pip install scapy==2.4.0	From scapy.all import *

Pour installer la bibliothèque openCV : pip install opencv-python

Lien intéressant : <http://www.dmi.unict.it/~furnari/teaching/CV1617/lab0/>

Lien bases pour Python : <http://www.python-simple.com/python-numpy-scipy/random-numpy.php>

## Annexes :

PIP : Les commandes pipeline pour installer des modules supplémentaires

Installer une bibliothèque	pip install nom
Installer une version d'une bibliothèque	pip install nom==version
Désinstaller une bibliothèque	pip uninstall nom
Chercher une bibliothèque avec ses dépendances	Pip search nom
Obtenir la liste des bibliothèques installées	Pip freeze

JUPYTER : Les commandes Jupyter

Activer de nombreuses extensions visibles dans nbextension	jupyter contrib nbextension install --user
--	--

## Utiliser jupyter depuis EduPython

- Aller dans le menu Outils/Outils/Installation d'un nouveau module
- Choisir conda
- Taper jupyter
- (En ligne de commande) Une fois l'installation terminée, aller dans EduPython\AppData\Scripts
- (En ligne de commande) exécuter jupyter-notebook.exe
- Par la suite, il vaut mieux créer un fichier .bat qui procède au lancement de jupyter