Mode d'emploi de Pyzo

I Raccourcis claviers et caractères spéciaux

Les raccourcis claviers sont bien plus efficaces que la navigation à la souris. Usez-en. On part du principe que vous disposez d'un clavier français, sous Windows. Sous Mac, pensez à substituer dans ce qui suit la touche `Ctrl` par la touche `cmd`.

Fonction	Raccourci clavier
Ouvrir un nouveau fichier (**N**ew)	`Ctrl` + `N`
Enregistrer le fichier (**S**ave)	`Ctrl` + `S`
Enregistrer sous ...	`Ctrl` + `Shift` + `S`
Copier (**C**opy)	`Ctrl` + `C`
Coller	`Ctrl` + `V`
Couper	`Ctrl` + `X`
Annuler la frappe dans l'éditeur (undo)	`Ctrl` + `Z`
Indenter	`Tab` (à côté de `1`)
Désindenter	`Shift` + `Tab`
Commenter	`Ctrl` + `R`
Déommenter	`Ctrl` + `T`
Exécuter le script (**E**xecute)	`Ctrl` + `E`
Exécuter la région du script en surbrillance (Execute selection)	`Alt` + `Entrée`
Se placer dans le répertoire du script et l'exécuter le script	`Ctrl` + `Maj` + `E`
Récupérer la dernière commande tapée dans la console	↑
Naviguer dans les commandes tapées dans la console	↑ et ↓
Dans la console (interprète ou shell)	
Nouvelle console	`Ctrl` + `1`
Effacer la console	`Ctrl` + `L`
Interrompre l'exécution de la console	`Ctrl` + `I`
Tuer le noyau et le redémarrer et la console	`Ctrl` + `K`
Caractères spéciaux	
# (pour les commentaires)	`Alt Gr` + `3`
% (commandes magiques)	Taper `magic`
! (commandes shell)	Dans la console. Au sens des systèmes *N*X).
_ (le fameux tiret du 8 : underscore)	`Alt Gr` + `8`
Remarque importante aux utilisateurs de Mac	
Dans tout ce qui précède	Remplacer la touche `Ctrl` par `cmd`
Crochets (pas disponibles en AZERTY)	`alt` + `Maj` + `(` / `alt` + `Maj` + `)`

II Messages d'erreurs de Python les plus classiques

	Messsage	Description et Explications éventuelles	Solution ?
1.	`ModuleNotFoundError: No module named xxxxxx`	Vous avez soit importé un module qui n'existe pas, soit qui existe mais qui est mal orthographié.	Par exemple (c'est le plus fréquent) : vous avez écrit `mathplotlib` au lieu de `matplotlib` ou `pylot` au lieu de `pyplot` ou encore `maths` au lieu de `math`.
2.	`IndentationError: unexpected indent`	Vous avez créé à tort une indentation.	Par exemple : votre ligne commence par une espace, ou l'arborescence dans votre script n'est pas correcte (un `then` pas au bon niveau d'indentation) : les `if`, `then`, `else` sont au même niveau d'indentation. Attention, après `else`, on met `:` et on va à la ligne.
3.	`NameError: name 'xxxxx' is not defined`	Le nom xxxxx est problématique	Soit vous avez utilisé une variable qui n'existe pas (attention à la casse : majuscule/minuscule), ou mal orthographié un mot-clé de Python. Normalement, dans l'éditeur, il doit apparaître coloré si il est bien orthographié.
4.	`SyntaxError: EOL while scanning`	Vous avez probablement oublié une parenthèse fermante ou un crochet fermant à la ligne précédant la ligne soulevée par Pyzo	Identifiez le(s) délimiteur(s) manquant(s) et corrigez
5.	`IndexError: list index out of range`	Problème d'indices	Votre tuple, ou votre liste, ou votre tableau `numpy` ne possède pas d'élément à la plage spécifiée. Vous souvenez-vous bien du fait que `(np.a)range(x,y)` commence à `x` et finit à `y-1` ?
6.	`TypeError: '***' object is not subscriptable`	Problème de type : vous essayez d'appeler un élément par son indice alors que l'objet de type `'***'` n'admet pas d'indice	Remplacez les crochets par des parenthèses. Vous avez dû confondre. Ou alors vous avez oublié une multiplication `*` ?
7.	`TypeError: '***' object is not callable`	Problème de type : vous essayez d'utiliser votre objet comme une fonction mais l'objet de type `'***'` n'est pas une fonction	Remplacez les parenthèses par des crochets. Vous avez dû confondre

8.	**Caractères étranges**	J'ai des caractères bizarres qui s'affichent au lieu des lettres accentuées.	Problème de configuration de l'encodage des caractères. Théoriquement, la norme d'encodage est UTF-8. Vous êtes probablement restés sur la norme Latin-1 qui tend à devenir obsolète. Bonne solution : travailler seulement avec des *chaines unicode*. Pour cela, préfixez la chaine dans print d'un u.
9.	`AttributeError: 'xxx' object has no attribute 'yyy'`	Un attribut pour un objet n'existe pas	C'est un usage à tort du point `.` Ne serait-ce pas plutôt une virgule qu'il faudrait ?
10.	`No such file or directory: 'foo.txt'`	Un dossier ou un fichier n'a pas été trouuvé	Avez-vous bien orthographié le nom du fichier ? Le fichier existe-t-il ? Votre répertoire d'exécution est-il celui contenant votre fichier ? En tous cas, il vaut mieux : **a.** placer votre script principal et le(s) fichier(s) à charger dans le même dossier **b.** exécuter votre script par le raccourci `Ctrl` + `Maj` + `E` (`run file as script`)
11.	`TypeError: '***' object is not callable`	Problème de type : vous essayez d'utiliser votre objet comme une fonction mais l'objet de type `'***'` n'est pas une fonction	Remplacez les parenthèses par des crochets. Vous avez dû confondre
12.	`Could not run code because it is incomplete`	Code non exécuté car incomplet	Vous avez ouvert un délimiteur (en général : `(`, `[`, `'`, ou même `"""` que vous n'avez jamais fermé

III Structure générale d'un script Python

Code Code Python ▼ **Commentaires**

```
1  # -*- coding: utf-8 -*-
2  """
3  Fichier créé le 1er janvier 2017
4   à 8h00. Version 1. Auteur : Moi.
5  Résumé - description   du script :
6   < à vous de compléter >
7  Mon premier Script Python.
8  """
9
10 #-------------------------------------
11 ##1.IMPORTATION DES MODULES
12 _____________________________________
13 from scipy import *
14 import numpy as np
15
16
17 #-------------------------------------
18 ##2.DEFINITION DES CONSTANTES
19 _____________________________________
20 CHEMIN=r'Chemin à mon dossier'
21 CONSTANTE_1 = ...
22 CONSTANTE_2 = ...
23
24 #-------------------------------------
25 ##3.DECLARATION DES FONCTIONS
26 _____________________________________
27
28 def maFonction(x):
29     """docstring"""
30     ...
31     ...
32     return y
33
34 ...
35 ...
36 ...
37 def monAutreFonction(x):
38     """docstring"""
39     ...
40     ...
41     return y
42
43
44 #-------------------------------------
45 ##4.DEBUT DU PROGRAMME PRINCIPAL
46 _____________________________________
47
48 print ('Bonjour')
49 ...
50 ...
51 ...
52 # Fin du fichier
```

ligne 1 Spécifie l'encodage de votre script. Le standard est `utf-8`. Depuis `Python 3`, il est inutile de le préciser, mais si vous avez des problèmes d'encodage (notamment vos lettres accentuées apparaissent bizarrement), recopiez cette ligne au début de votre script.

lignes 4-8 .

a. C'est la zone du `docstring` du fichier : mettez-y le descriptif nécessaire pour présenter rapidement l'objet de votre programme, éventuellement l'historique des versions.

b. Les délimiteurs du docstring sont trois guillemets : `"""` (lignes 2 et 8).

c. Tout ce qui se trouve entre ces délimiteurs n'est pas interprété par Python.

d. Il est important de bien documenter ses programmes : afin de les comprendre rapidement si on les réutilise (surtout si il s'écoule beaucoup de temps entre deux utilisations) et aussi pour que ceux qui s'en servent le comprennent !

ligne10 Le caractère # sert à insérer des commentaires dans votre programme. Tout ce qui suit un # est ignoré, et donc, n'est pas interprété par Python. Sous Pyzo, ## crée une section dans le script, et la fenêtre `source structure` permet une navigation aisée dans le script.

lignes 10,18,26,46 Structurez toujours ainsi vos scripts :

a. *Importation des modules.* Tout ce qui vous sera utile.

b. *Les constantes* : c'est-à-dire les paramètres fixes qui servent régulièrement dans le programme.

1) Intérêt : si vous modifiez vos constantes, vous n'aurez pas à reparcourir tout votre scipt pour faire les modifications.

2) **ligne 22.** Notez qu'une constante peut-être une chaîne de caractères, par exemple, le chemin d'accès à un dossier [a].

c. *Déclaration des fonctions.* Elles sont de cette manières toutes regroupées dans le script.

d. *Corps du programme.* Si vous avez bien découpé vos tâches en fonctions, le corps du programme consiste essentiellement en l'utilisation de vos fonctions.

a. Noter que la chaîne est préfixée d'un `r` pour dire que la chaîne est une chaîne brute

IV Python, IEP, Pyzo, conda ?

a. Python est un langage de script ou de programmation, c'est tout. Donc, **ne dites pas «Le logiciel Python»** (surtout à un oral de TIPE, ou à votre prof de maths, c'est hyper agaçant). Un langage de script permet de formuler des suites d'ordres (par le biais d'instructions) que votre texttt exécute.

b. IEP est un acronyme pour *Interactive Editor Python*. C'est bien un logiciel. Il s'agit d'un éditeur de texte amélioré (ou plutôt adapté) fournissantt un environnement de développement efficace pour rédiger des scripts en Python. En résumé, c'est du word amélioré pour écrire du Python.

c. Pyzo est une distribution de Python (une distribution est *grosso modo* un ensemble contenant de quoi permettre à votre machine d'utiliser le langage Python ainsi que des (bouquets de) modules prêts à l'emploi, et éventuellement des logiciels annexes). Ce qui différencie deux distributions est la richesse de ce qui s'ajoute à la version de Python fournie. Ainsi Pyzo fournit IEP

d. Ana/Mini-conda sont aussi des distributions de Python, particulièrement adaptées à un usage scientifique de Python (c'est-à-dire, orientées vers le calcul scientifique, numérique, l'ingénierie, et depuis peu, les *data sciences*, terme à la mode pour parler de l'analyse statistique. Le *data scientist* est très prisé en ce moment.). Miniconda est une version minimaliste d'Anaconda au sens où sa version nue dispose de moins de modules qu'Anaconda [1], ce dernier restant le couteau suisse du scientifique pythoniste.

e. Conda est un outil intégré à ana/mini-conda de gestion de packages : il permet simplement d'installer, de mettre à jour ou de personnaliser sa distribution de Python.

f. Alors, Pyzo ou Anaconda ? Depuis juin 2016 environ, Pyzo est revenu avec une distribution de Python. Elle est donc opérationnelle sans recourir à Anaconda. En outre, il est désormais possible d'installer des packages depuis la console incluse dans Pyzo puisque le gestionnaire conda est fourni ! Personnellement, j'utilise anaconda mais Pyzo et sa distribution suffisent. Toutefois, si vous avez installé une ancienne version de Pyzo, il est possible que vous ne disposiez que de miniconda, en quel cas

V Modules

a. Un *module* est simplement un fichier python tel que math.py. En général, ce dernier contient des définitions de fonctions, variables *etc.*

b. Un *package* est un répertoire structuré avec son système d'arborescence spécifique, contenant un ensemble de modules et un fichier d'initialisation décrivant l'arborescence de ce répertoire. Le chargement du package entraîne le chargement de ses variables, fonctions. objets et méthhodes [2] mais pas des modules sous-jacents. Par exemple, le package numpy permet de définir l'objet array en vue du calcul matriciel optimisé en termes de temps d'exécution, ou des fonctions spécifiques aux problèmes d'algèbre linéraire, *etc.*. Chacune de ces caractéristiques est implémentée précisément par des modules dédiés (p.ex le module linalg.py est destiné à implémenter des fonctions spécifiques aux problèmes d'algèbre linéaire, notamment le calcul des puissances de matrices).

VI Démarrage de Pyzo

a. Ouvrir sa session (vous devez disposer de votre identifiant réseau et de votre mot de passe fournis au début de l'année).

b. 1) Sur le bureau, ouvrir le dosssier Math-info et double-cliquer sur le raccourci Pyzo.

2) Vous n'avez pas les droits d'écriture sur le bureau. Toutefois, vous pouvez créer un raccourci pour lancer Pyzo depuis le menu démarrer de Windows : pour cela, glissez-déposez le raccourci Pyzo vers le menu démarrer. Ce dernier va se développer. Déposez-y votre raccourci (juste en dessous de votre nom).

c. Attendre un petit peu le démarrage de Pyzo.

VII Règles générales

a. Sauvegardez souvent pour éviter les pertes (un petit Ctrl + S régulièrement) !

1. Par exemple, il vous faudra installer vous-même numpy. La distribution Miniconda est très bien pour rajouter ses modules au fur et a mesure des besoins et économiser de l'espace sur son disque dur.
2. Terminologie des langages orientés objet, mais ce n'est précisément pas…l'objet du propos

b. Structurez en un algorithme avant de programmer. Vous serez plus efficace et moins suceptibles de commettre des erreurs.

c. **Pas de caractères spéciaux ni espaces, ni lettres accentuées dans les noms de répertoires, de fichiers**. On évite ainsi d'éventuels problèmes d'encodage, surtout pour les chemin d'accès de fichiers.

d. Idem pour les noms des variables. Caractère toléré : _.

e. Préferez les raccourcis clavier.

f. Documentez vos scripts : si vous produisez un travail collaboratif, tout le monde doit comprendre ce que vous faites, vous le premier ! (surtout si vous reprenez votre programme plusieurs semaines plus tard).

g. Donnez des noms évocateurs à vos variables.