

Raccourcis clavier usuels

`Ctrl+n` crée un nouveau fichier, `Ctrl+o` ouvre un fichier existant, `Ctrl+s` enregistre le fichier
`Ctrl+i` interrompt l'exécution, `Ctrl+k` interrompt l'exécution puis redémarre le *shell*
`Ctrl+e` ou `f5` exécute l'intégralité du fichier
`Ctrl+Retour` exécute la cellule courante (délimitée par `##` au début et en fin de cellule)
`alt+Retour` ou `f9` exécute la sélection
`Ctrl+z` annule la dernière manipulation, `Ctrl+c`, `Ctrl+x`, `Ctrl+v` copie, coupe, colle la sélection
`Ctrl+r`, `Ctrl+t` commente, décommente la sélection

Fondements

• Déclaration et affectation de variables

`var = 1`, `var = 1.` affecte l'entier 1, le flottant 1, à la variable *var*
`var1, var2, var3 = 1, "abc", [x,y,z]` affecte respectivement l'entier 1, la chaîne *abc* et la liste *[x,y,z]*
`var1, var2 = var2, var1` échange les contenus des deux variables *var1* et *var2*

• Changer le type d'une variable

`int(var)`, `float(var)`, `str(var)` transforme la variable en entier, flottant, chaîne de caractères

• Variable saisie au clavier

`var = input("texte s'affichant à l'écran")` affecte la chaîne de caractères saisie (convertir si besoin)

• Affichage

`print(var)` affiche à l'écran le contenu de la variable *var*

• Opérateurs spéciaux

`x // y` quotient, `x % y` reste de la division euclidienne de *x* par *y*, `x**y` puissance x^y
`lep` écriture scientifique de 10^p

Conditions et boucles

• Opérateurs logiques

`x == y` (*x* égale *y*), `x != y` (*x* ≠ *y*), `x < y`, `x <= y`, `x < y < z` renvoie True ou False

`not A`, `A or B`, `A and B` renvoient respectivement les booléens

contraire de *A*, True si *A* ou *B* est True et False sinon, True si *A* et *B* sont True et False sinon

• Condition

`if conditions : instructions` ou

```
if conditions1 : instruction1
elif conditions2 : instruction2
else : instructions3
```

• Boucle while, quand on ne connaît pas a priori le nombre de tours de boucle à faire

```
while conditions :
    instructions (à répéter tant que conditions est vraie)
```

• Boucle for, quand on connaît a priori le nombre de tours de boucles à faire

```
for k in range(a, b) :
    instructions (à répéter pour k allant de a à b - 1)
```

 par défaut $a = 0$

```
for elt in L :
    instructions (à répéter pour chaque élément de la liste L)
```

`break` ou `return` dans les instructions permet d'arrêter la boucle

• Importer une bibliothèque

`import bibliothèque as raccourci` importe toutes les fonctions, le raccourci comme préfixe

`from bibliothèque import noms des fonctions séparées par des virgules` pas besoin de préfixe

`from bibliothèque import *` importe toutes les fonctions, pas besoin de préfixe

• Les modules et fonctions courantes

le module `random` pour les nombres aléatoires : `import random as rd`

`rd.random()` renvoie un flottant aléatoire choisi au hasard entre 0 et 1

`rd.randint(i,j)` renvoie un entier aléatoire choisi au hasard entre les entiers i et j compris

`rd.choice(L)` renvoie un élément choisi au hasard dans la liste L

le module `math` pour les fonctions mathématiques (sauf `abs`) : `import math as m`

`m.pi` est une valeur approchée de π

`m.sqrt(x)`, `m.exp(x)`, `m.log(x)` renvoient \sqrt{x} , e^x , $\ln(x)$

`m.sin(x)`, `m.cos(x)`, `m.tan(x)`, `m.asin(x)`, `m.acos(x)`, `m.atan(x)`

`m.floor(x)`, `m.ceil(x)` renvoient la partie entière de x et l'entier supérieur ou égal à x

le module `pyplot` de la bibliothèque `matplotlib` pour les graphiques :

`import matplotlib.pyplot as plt`

`plt.plot(Lx,Ly,options)` définit le polygone reliant les points de coordonnées dans Lx et Ly

options permet de définir le format et la couleur du tracé :

`color = 'red'` ou `'r'` ou toute autre nom de couleur en anglais

`marker = '+'` ou `'.'` ou `'o'` ou `'*'` etc.

`linestyle = '-'` (continu) ou `'--'` (tiret) ou `'-.'` (pointillés)

`linewidth = x` définit l'épaisseur du tracé

`plt.plot(Lx,Ly,'o')` affiche le nuage de points de coordonnées dans Lx et Ly

`plt.show()` affiche, `plt.figure(n)` numérote, `plt.clf()` efface la figure courante

`plt.savefig(nomdefichier)` sauvegarde la figure courante dans le fichier `nomdefichier`

`plt.xlabel('nomx')` et `plt.ylabel('nomy')` affiche un nom sur les axes

`plt.title('titre')` affiche le titre à la figure

`plt.hist(L,options)` affiche l'histogramme de la liste L

options permet de définir le format et les couleurs

`bins = liste des bornes des rectangles` par défaut ce sont les entiers de $\min(L)$ à $\max(L)$

`normed = True` représente les fréquences (dont la somme vaut 1)

`orientation = 'horizontal'` permet de tracer l'histogramme horizontalement

`rwidth = 0.5` réduit la largeur des rectangles de 50% (avec un espace entre elles)

`color = 'red'` ou toute autre nom de couleur pour les rectangles

`edgecolor = 'red'` ou toute autre nom de couleur pour les bordures

`hatch = '/', '\', '|', '-.', '+', 'x', 'o', 'O', '.', '*'` permet de hachurer les rectangles

le module `numpy` pour les matrices (`array`) : `from numpy import array`

les vecteurs sont considérés comme des matrices lignes

Les `array` permettent une bonne performance de calculs car beaucoup d'opérations sont implémentées mais elles ont une taille fixée à la création et tous les éléments d'une `array` doivent être du même type

Les fonctions

• Déclaration d'une fonction (définie une fonction mais ne la fait pas tourner !)

```
def nom_fonction(arguments séparés par des virgules) :  
    instructions  
    return valeur(s) attendue(s) séparées par des virgules
```

• Appel d'une fonction, qui doit être déclarée au préalable

```
variable(s) séparée(s) par des virgules = nom_fonction(valeurs des paramètres)
```

Les listes

• Création d'une liste

directement $L = [x, y, \dots]$ liste des éléments $[x, y, \dots]$, $L = []$ liste vide, $L = [x] * n$ liste $[x, x, \dots, x]$

avec range $L = \text{range}(i, j, p)$ vecteur des entiers de i à $j - 1$ avec pas de p (par défaut $i = 0, p = 1$)

$\text{range}(n)$ est le vecteur $[0, 1, \dots, n - 1]$ et $\text{list}(\text{range}(n))$ est la liste $[0, 1, \dots, n - 1]$

par compréhension $L = [f(x) \text{ for } x \text{ in } \text{list if } C]$ liste des images par f des éléments de list vérifiant C

• Extraction d'éléments d'une liste sans modification de la liste

$L[i]$ est l'élément d'indice i de la liste L

$L[-i]$ est le i -ième élément de la liste L en commençant par la fin

$L[i : j : p]$ est la liste des éléments d'indices i à $j - 1$ de la liste L par pas de p

par défaut $i = 0, j = \text{dernier indice}, p = 1$

en particulier $L[i :]$ est la liste des éléments de la liste L à partir de celui d'indice i

$L[: j]$ est la liste des éléments de la liste L jusqu'à celui d'indice $j - 1$

$L[:: 2]$, $L[1 :: 2]$ est la liste des éléments de L d'indices pairs, impairs

$L[::-1]$ est la liste des éléments de L en commençant par la fin

• Modification d'une liste

$L.append(x)$ ajoute l'élément x à la fin de la liste L

$\text{del}(L[i])$ supprime l'élément d'indice i de la liste L (le reste est décalé)

$L.insert(i, x)$ insère l'élément x avant l'élément d'indice i dans la liste L (le reste est décalé)

$x = L.pop(i)$ affecte à x la valeur de l'élément d'indice i de la liste L et le supprime de L

$L[i] = x$ affecte la valeur x à l'élément d'indice i de la liste L

$L.sort()$ trie la liste L dans l'ordre croissant

$L.reverse()$ inverse l'ordre des items de la liste L

$L.remove(x)$ supprime la première occurrence de l'élément x dans la liste L (le reste est décalé)

• Opérations et fonctions sur les listes

$L1 + L2$ est la concaténation des deux listes $L1$ et $L2$ $n * L$ est la concaténation de n fois L

$L1 = \text{list}(L)$ ou $L1 = L[:]$ affecte à $L1$ la liste des éléments de L

Attention : $L1 = L2$ ne crée pas une nouvelle liste mais donne deux noms à une même liste

$n = \text{len}(L)$ affecte à n la longueur de la liste L (les indices vont de 0 à $n - 1$)

$L.count(x)$ donne le nombre d'occurrences de l'élément x dans la liste L

$L.index(x)$ donne l'indice de la première occurrence de l'élément x dans la liste L

• **Création d'un tableau**

`np.array([L1, L2, ...])` crée le tableau dont les lignes sont $L1, L2, \dots$

Attention : les listes $L1, L2, \dots$ doivent être toutes de même type et de même longueur

`np.zeros(taille, type)` crée le tableau nul de taille $taille$ et de type $type$ (flottant par défaut)

pour un vecteur $taille = n$, pour une matrice $taille = (n, p)$

`np.ones(taille, type)` crée le tableau de taille $taille$ et de type $type$ dont tous les éléments valent 1

`np.identity(n)` crée le tableau unité d'ordre n

`np.random.rand(n, p)` crée un tableau de taille (n, p) de nombres aléatoires de $[0, 1]$

`np.diag(L)` crée le tableau carré d'ordre $len(L)$ dont les éléments diagonaux sont ceux de L

`np.linspace(a, b, n)` crée le vecteur de n valeurs régulièrement espacées entre a et b compris

`np.arange(a, b, p)` crée le vecteur des entiers de a à $b - 1$ avec pas p (par défaut $a = 0, p = 1$)

• **Extraction d'éléments d'un tableau sans modification**

`M[i, j]` est l'élément d'indices i, j de M

`M[i : j : p, a : b : q]` est le tableau extrait de M

pour les lignes de i à $j - 1$ par pas de p (par défaut $i = 0, j =$ dernier indice, $p = 1$)

pour les colonnes de a à $b - 1$ par pas de q (par défaut $a = 0, b =$ dernier indice, $q = 1$)

en particulier `M[i, :]` ou `M[i]` est le vecteur constitué de la ligne d'indice i de M

`M[:, j]` est le vecteur constitué de la colonne d'indice j de M

`M[:, : 2, :]` est le tableau constitué des lignes d'indices pairs de M

`M[1 :: 2, :]` est le tableau constitué des lignes d'indices impairs de M

`M[:, : -1, :]` est le tableau constitué des lignes de M lues dans le sens inverse

• **Modification d'un tableau**

`np.append(A, B)` est le vecteur obtenu par concaténation des lignes de A puis de B

`np.append(A, B, axis = 0)` est le tableau obtenu par concaténation du tableau B sous le tableau A

`np.append(A, B, axis = 1)` est le tableau obtenu par concaténation du tableau B à droite de A

`B = A.copy()` affecte à B le tableau A (attention : $B = A$ donne deux noms à un même tableau)

• **Opérations, méthodes et fonctions sur les tableaux**

`V.size` renvoie la longueur du vecteur V

`M.shape` renvoie la taille du tableau M sous forme d'un tuple

`M.reshape(n, p)` transforme le tableau M en un tableau de taille (n, p)

Une fonction f peut être appliquée à chaque élément d'un tableau M par la commande $f(M)$:

`M + x`, `M * x`, `x ** M`, `1/M`, `sin(M)`, `exp(M)` etc.

`M1 + M2`, `M1 * M2` renvoie la somme, le produit terme à terme des deux tableaux $M1$ et $M2$

`np.dot(M1, M2)` renvoie le produit matriciel des tableaux $M1$ et $M2$

`np.matrix_power(M, n)` renvoie le tableau M^n

`np.linalg.inv(M)` renvoie le tableau inverse de M

`np.transpose(M)` renvoie le tableau transposé de M