# Aide-mémoire PYTHON3 - v3.7.1

## Fonctions prédéfinies

- `abs(x)` : valeur absolue de x
- `int(x)` : valeur x convertie en entier
- `float(x)` : valeur x convertie en réel
- `str(x)` : valeur x (int ou float), convertie en str
- `list(x)` : valeur x convertie en liste
- `tuple(x)` : valeur x convertie en tuple
- `dict(x)` : séquence de couples x convertie en dictionnaire
- `set(x)` : x converti en ensemble
- `help(x)` : aide sur x
- `dir(x)` : liste des attributs de x
- `type(x)` : type de x
- `print(...)` : imprime
- `input(x)` : imprime le string x et lit le string qui est introduit
- `round(x [,ndigits])` : valeur arrondie du float x à ndigits chiffres (par défaut 0)
- `range([start], stop, [step])` : retourne une suite d'entiers
- `sorted(s)` : retourne une liste avec les éléments de s triés

## Gather, scatter et keyword arguments

- `def fun(*args)` : *gather* des arguments en un tuple args
- `fun(*s)` : *scatter de la séquence s lors de l'appel

## Opérations et méthodes sur les séquences (str, list, tuples)

- `len(s)` : longueur de la séquence s
- `min(s), max(s)` : élément minimum, maximum
- `sum(s)` : (ne fonctionne pas pour les string) : somme de tous les éléments (valeur numérique)
- `s.index(value, [start, [stop]])` : premier indice de value dans s[start :stop]
- `s.count(sub [,start [,end]])` : nombre d'occurrences sans chevauchement de sub dans `s[start:end]`
- `enumerate(s)` : construit une séquence de couples dont le ième élément (à partir de 0) vaut le couple (i, s[i])
- `zip (a,b), zip(a,b,c)`, ... : construit une séquence de couples, resp. triples, ..., dont le ième élément reprend le ième élément de chaque séquence a, b [,c]

## Méthodes sur les chaînes de caractères (str)

- `s.lower(), s.upper()` : string avec caractères en minuscules respectivement en majuscules
- `s.islower()`, `s.isdigit()`, `s.isalnum()`, `s.isalpha()`, `s.isupper()` : vrai si s n'est pas vide et n'a (respectivement) que des minuscules, des chiffres, des car. alphanumériques, alphabétiques, majuscules
- `s.find(sub [,start [,end]])` : premier indice de s où le sous string sub est trouvé dans `s[start:end]`
- `s.replace( old, new[, co])` : copie de s en remplaçant toutes les (ou les co premières) occurrences de old par new.
- `s.format(...)` : copie de s après formatage
- `s.capitalize()` : copie de s avec première lettre en majuscule
- `s.strip()` : copie de s en retirant les *blancs* en début et fin
- `s.join(t)` : crée un str qui est le résultat de la concaténation des éléments de la séquence de str t chacun séparé par le str s
- `s.split([sep [,maxsplit])` : renvoie une liste d'éléments séparés dans s par le caractère sep (par défaut *blanc*) ; au maximum maxsplit séparations sont faites (par défaut l'infini)

CC BY SA

## Opérateurs et méthodes sur les listes

- `s.append(v)` : ajoute un élément valant v à la fin de la liste
- `s.extend(s2)` : rajoute à s tous les éléments de la liste s2
- `s.insert(i,v)` : insère l'objet v à l'indice i
- `s.pop([i])` : supprime l'élément d'indice i de la liste (par défaut le dernier) et retourne la valeur de l'élément supprimé
- `s.remove(v)` : supprime la première valeur v dans s
- `s.reverse()` : renverse l'ordre des éléments de la liste, le premier et le dernier élément échangent leurs places, ....
- `s.sort(key=None, reverse=False)` : trie s en place
- `s.copy()` : *shallow* copie superficielle de s
- `del s[i], del s[i:j]` : supprime un ou des éléments de s

## Méthodes sur les dictionnaires (dict)

- `d.clear()` : supprime tous les éléments de d
- `d.copy()` : *shallow* copie de d
- `{}.fromkeys(s,v)` : crée un dict avec les clés de s et la valeur v
- `d.get(k [,v])` : renvoie la valeur d[k] si elle existe v sinon
- `d.items()` : liste des items (k,v) de d
- `d.keys()` : liste des clés
- `d.pop(k [,v])` : enlève d[k] s'il existe et renvoie sa valeur ou v sinon
- `d.popitem()` : supprime un item arbitraire (k,v) et retourne l'item sous forme de tuple
- `d.setdefault(k [,v])` : d[k] si elle existe sinon v et rajoute d[k]=v
- `d.update(s)` : s est une liste de tuples que l'on rajoute à d
- `d.values()` : liste des valeurs de d
- `del d[k]` : supprime l'élément de clé k de d

## Méthodes sur les ensembles (set)

- `s = set(v)` : initialise s : un set contenant les valeurs de v
- `s.add(v)` : ajoute l'élément v au set s (ne fait rien s'il y est déjà)
- `s.clear()` et `s.copy()` : idem dictionnaires
- `s.remove(v)` : supprime l'élément v du set (erreur si v n'est pas présent dans s)
- `s.discard(v)` : si v existe dans s, le supprime
- `s.pop()` : supprime et renvoie un élément arbitraire de s

## Modules

- `math` : accès aux constantes et fonctions mathématiques (`pi`, `sin()`, `sqrt(x)`, `exp(x)`,`floor(x)` (valeur plancher), `ceil(x)` (valeur plafond), ...) : exemple : `math.ceil(x)`
- `copy: copy(s), deepcopy(s)` : *shallow* et *deepcopy* de s

## Méthodes sur les fichiers

- `f = open('fichier')` : ouvre 'fichier' en lecture (autre paramètres possibles : 'w'(en écriture), 'a'(en écriture avec ajout), encoding ='utf-8' : encodage UTF-8)
- `with open('fichier'...) as f` : ouvre 'fichier' pour traitement à l'intérieur du with
- `for ligne in open('fichier'...)` : ouvre et traite chaque ligne de 'fichier' et le ferme à la fin du for
- `f.read()` : retourne le contenu du fichier texte f
- `f.readline()` : lit une ligne
- `f.readlines()` : renvoie la liste des lignes de f
- `f.write(s)` : écrit la chaîne de caractères s dans le fichier f
- `f.close()` : ferme f