Mémento Python

Remarque préliminaire : les indentations sont très importantes dans Python ; un programme mal indenté ne sera pas compris par la machine, ou il sera compris d'une autre façon que celle que vous souhaitiez au départ.

	Langage Python
Commentaires	`♯` ...
Affectations	$a = ...$ pour les identifiants des variables, caractères spéciaux interdits, sauf _
Affichages	`print ('coucou')` (pour un message) `print (a)` (pour la valeur d'une variable a) `print ('voici la valeur de a et b : ',a,' ',b)` (pour faire plusieurs affichages) `print('coucou',end=' ')` (pour annuler le retour à la ligne après l'affichage)
Saisies	`input()` // ATTENTION : le type de la saisie est toujours une chaine de caractères `a=input('donne moi la valeur de a')` (pour afficher un texte avant la saisie)
transtypages	`int(ch)` Transforme une chaine en entier : '1' devient 1 `str(a)` Transforme un entier en chaine : 1 devient '1'
Struct. conditionnelles	`if` condition `:` `else :`
Tests	`==` (égal) `!=` (différent) `>` `<` `>=` `<=`
Opérateurs booléens	`and` `or` `not` (négation)
Opérations mathématiques	$+$ $-$ $/$ $*$ (produit) $**$(puissance) `//`(division entière) % reste de la division entière la virgule d'un nombre décimal se note avec un .
Autres fonctions mathématiques	accessibles en mettant au début du script `from math import *` entre autres : `e`, `pi` `exp()` (exponentielle) `log()` (ln) `sqrt()` (racine carrée)
Nombres aléatoires	accessibles en mettant au début du script `from random import *` entre autres : `random()` ($\in [0;1[$ et `randrange(n,p)` (entier entre n et $p-1$).

	Langage Python
Struct. itérative TANTQUE	• initialisation de la condition • `while` condition `:` • traitement • recalcul de la condition
Struct. itérative POUR	`for` variable `in` **liste** `:` **liste** peut être • un mot, dans ce cas la variable est automatiquement de type caractère. • `range(n)` : liste des entiers compris entre 0 et n-1. • `range(p,n)` : liste des entiers compris entre p et n-1. • `range(p,n,i)` : liste des entiers compris entre 0 et n-1, avec un pas de i • une liste donnée explicitement avec le format de liste.
fonctions	• `def` nomfonction `(` var$_1$, var$_2$, ... `)` `:` # ne pas utiliser de variables globales • traitement • `return(...)`
Chaines de caractères	• `len`(chaine) donne la longueur de la chaine. • `chaine[i]` renvoie le caractère de rang i. (le premier est de rang 0, et le dernier de rang len(chaine)-1. • `chaine[-1]` renvoie le dernier caractère, `chaine[-2]` l'avant dernier, etc... • `chaine[n:p]` renvoie les caractères de rang n à p-1. • `chaine[:n]` renvoie les caractères de rang 0 à n-1, soit les n premiers caractères. • `chaine[n:]` renvoie les caractères de rang n à la fin, soit la chaine tronquée des n premiers caractères. • On ne peut pas modifier une chaine de caractères.
Type Liste (tableau)	• `[]` est la liste vide. • `[0]*8` est une liste contenant 8 zéros. • On accède aux éléments comme dans les chaines de caractères : `L[1]` renvoie le 2ème élément d'une liste L. • On manipule les listes comme les chaines de caractères (concaténation, multiplication ...). Contrairement aux chaines, on peut les modifier : • L.append(x) : ajoute l'élement x à la fin de la liste L. • del(L[i]) supprime l'élément de rang i de la liste L. • del(L[n :p]) supprime les éléments de rang n à p-1 de la liste L. • L[i]=3 remplace l'élément de rang i par 3. • L[n :p]=[1,2,"oiu",...] remplace les éléments de rang n à p-1 par toute la liste (de taille quelconque : on peut donc insérer des éléments) • L[2 :3]=[p,q,r] remplace l'élément de rang 2 par les 3 éléments p, q et r. • M=L[:] recopie la liste L dans la liste M. Attention, c'est un système de pointeurs. Si un élément de L est une liste, et qu'on en modifie un élément par M, alors il se modifiera aussi dans L.