Syntaxe de base en Python

1 Les principaux types de variables

	Syntaxe Python	Description
int	`a=-3, b=7`	Nombres entiers (integer)
float	`a=2/7, b=7.0, c=-3.3`	Nombres décimaux (floating point)
str	'Salut', "Quoi d'neuf?"	Chaînes de caractères (string)
bool	`2+3==5, 2.7>=5.5`	Booléens (valeur **True** (= 1) ou **False** (= 0))
list	`L1=[2,5,7], L2=["toto", 3.8]`	Liste d'éléments de types quelconques.

- Python attribue lui-même le type adéquat. On peut vérifier le type d'un élément avec l'instruction `type()`. Par exemple : l'instruction « type(1.0) » renvoie « <class 'float'> ».
- Conversion de type. On peut effectuer une conversion de type avec les instructions `int()`, `float()`, `str()`. Par exemple : l'instruction « a=3.0 » génère un float et l'instruction « a=int(a) » transtype a en entier (a=3).

2 Instructions

Une variable peut être conceptualisée par une boite (un espace mémoire) portant un nom et contenant une valeur On affecte une valeur à une variable à l'aide du signe =, en évaluant le membre de droite puis en l'affectant dans la variable à gauche.

	Syntaxe Python	Description
Affectations	`a=3, b=a+2`	Affecte la valeur 3 à `a` et de la valeur 5 à `b`.
Affectations multiples	`m, n, p = 3, 2, 1`	Affecte les valeurs 3, 2 et 1 aux variables `m`, `n`, `p`.
Permutation simultanée	`m, n = n, m`	Les valeurs des deux variables `m` et `n` sont échangées.
Opérations élémentaires	`a+b a-b a*b a**b` `a/b a//b a%b`	Somme, différence, produit, puissance, division, quotient entier, reste division euclidienne.
Opérations sur les "str"	`"Sa" + "lut"` `'Tchou ' * 3`	Concaténation ("Sa"+"lut" renvoie 'Salut') Répétition ('Tchou '*3 renvoie 'Tchou Tchou Tchou ')

Tests	`a==b a!=b`
	`a<b a<=b a>b a>=b`
	`not and or`
		Test d'égalité, de non-égalité.
		Test de comparaisons. Renvoie un booléen (True/False)
		Opérateurs logiques (combinables).

3 Interactions avec l'utilisateur.

	Syntaxe Python	Description
Affichage	`print('message', var)`	Ecrit le "message" suivi du contenu de la variable « var ».
Question	`rep = input('message')`	Ecrit le "message" et attend une réponse. La variable "rep" est toujours de type "str" et peut donc nécessiter une conversion de type (notamment si on souhaite travailler avec des nombres).

4 Instructions conditionnelles

	Syntaxe Python	Description
Instruction conditionnelle simple	**if** condition **:** —> Instructions_V si vrai **else :** —> Instructions_F si faux	Réalisation des instructions_V si la condition est True. Réalisation des instructions_F si la condition est False. Le bloc "else" est facultatif. **Les " :" et la tabulation "—>" sont obligatoires.**
Instruction conditionnelle multiple	**if** condition **:** —> Instructions **elif** condition **:** —> Instructions ... **else :** —> Instructions	L'instruction "elif" permet d'ajouter des conditions successives.

5 Boucles

	Syntaxe Python	Description
Boucle "for"	**for** elt **in** liste **:** —> Instructions	Boucle effectuée sur tous les éléments de la liste. **Les " :" et la tabulation "—>" sont obligatoires.**
Utilisatoin de "range"	**for** k **in range**(n) **:** —> Instructions	Boucle effectuée pour k prenant les valeurs entières de 0 inclus à n exclu ($0 \leqslant k < n$ donc n valeurs générées). **Les " :" et la tabulation "—>" sont obligatoires.**
Boucle "while"	**while** condition **:** —> Instructions	Boucle effectuée tant que la condition (de type bool) est vérifiée. Attention aux boucles infinies (la condition doit être évaluée à False pour sortir de la boucle). **Les " :" et la tabulation "—>" sont obligatoires.**

- On peut boucler sur les éléments d'une liste ou d'une chaine de caractère. Par exemple :
  - ⋆ **for** mot **in** ['rouge','vert','bleu'] ou **for** lettre **in** 'python'
- Options de l'instruction "range" :
  - ⋆ **range**($n1, n2$) permet de générer les entiers k tels que $n1 \leqslant k < n2$.
  - ⋆ **range**($n1, n2, p$) permet de générer les entiers k tels que $n1 \leqslant k < n2$ avec un pas p entre chaque valeur.

6 Fonctions

	Syntaxe Python	Description
Création d'une fonction	**def** toto(x) **:** —> Instructions —> **return** resultat	Définition d'une fonction "toto" ayant pour argument "x" et renvoyant "resultat" (tous types possibles). **Les " :" et la tabulation "—>" sont obligatoires.**
Appel de la fonction	toto(x)	Renvoie la valeur de l'image de x par la fonction toto. La fonction doit d'abord être compilée.

- Une fonction peut ne pas avoir d'argument : "def tata() :".
- Une fonction peut avoir plusieurs arguments, par exemple "def titi(x,y,z) :".
- L'exécution d'un return interrompt l'exécution de la fonction.
- Une fonction peut ne pas renvoyer de résultat (pas de return) et on parle alors plutôt de processus.

7 Les listes

	Syntaxe Python	Description
Création	L = [] L = [2,-3,0,4]	Création d'une liste vide. Création d'une liste contenant 4 entiers.
Longueur	**len**(L)	Fonction renvoyant le nombre d'éléments de la liste L (len=length) .
Accès à une valeur	L[k] L[0] L[-1]	Donne accès à la valeur de rang k de la liste L. Attention : numérotation à partir de 0. Premier élément de la liste. Dernier élément de la liste.
Modification d'une valeur	L[k] = v	Ecrase la valeur de rang k de la liste L par la valeur v. k doit être un indice compris entre 0 et len(L).
Test	v **in** L	Renvoie un booléen indiquant si la valeur v est dans la liste L.
Ajout et suppression d'éléments	L.append(v) L.remove(v) L.pop(k)	append : Place la valeur v à la fin de la liste L (ajout d'un élément). remove : Supprime la première apparition de la valeur v dans la liste L. pop : Supprime (et renvoie) la valeur de rang k de la liste L.
Création d'une liste en compréhension	L=[0 **for** k **in range**(11)] L=[k**2 **for** k **in range**(1,5)]	Création d'une liste de 11 zéros. Création d'une liste des carrés des quatre premiers entiers non nuls.
Parcours de liste	**for** elt **in** L **:** —> Instructions utilisant elt **for** i **in range**(len(L)) **:** —> Instructions utilisant L[i]	Boucle effectuée sur tous les éléments "elt" de la liste L (parcours par éléments). Boucle effectuée sur les éléments d'indice i de la liste (parcours par indices).

8 Bibliothèques.

Il existe des bibliothèques de fonctions utiles, que l'on peut importer et utiliser. Par exemple la bibliothèque "math" qui contient toutes les fonctions usuelles mathématiques.

	Syntaxe Python	Description
Importer un module	from math import *	Récupère toutes les fonctions du module "math". "*" permet de tout récupérer.
Importer des fonctions d'un module	from math import sqrt, exp	Récupère seulement les fonctions "racine carrée" et "exp" du module "math".

9 Les dictionnaires

	Syntaxe Python	Description
Création	dico = { } tel = {'Olive' : '0698765432', 'Tom' : '0612345678'}	Création d'un dictionnaire "dico" vide. Création d'un dictionnaire "tel" contenant 2 clés ('Olive' et 'Tom') et les deux valeurs associées.
Modificiation et ajout	tel['Olive'] = '0611223344' tel['Bruce'] = '0619283746'	La clé 'Olive' existe déjà. L'ancienne valeur est écrasée par la nouvelle. La clé 'Bruce' n'existe pas. Elle est ajoutée au dictionnaire avec la valeur associée.
Suppression	**del** dico[Cle]	Supprime la clé "Cle" et la valeur associée du dictionnaire "dico".
Longueur	**len**(dico)	Fonction renvoyant le nombre d'éléments du dictionnaire "dico" (len=length).
Accès à une valeur	tel['Olive']	Donne accès à la valeur associée à la clé 'Olive' du dictionnaire "tel".
Test	Cle **in** dico	Renvoie un booléen indiquant si la clé "Cle" est dans le dictionnaire "dico".
Parcours	**for** Cle **in** dico **:** —> print(Cle) **for** Cle **in** dico **:** —> print(dico[Cle])	Parcourt le dictionnaire "dico" et affiche ses clés. Parcourt le dictionnaire "dico" et affiche ses valeurs.