MEMENTO PYTHON

Variables		Une variable est une donnée évolutive stockée dans la mémoire de l'ordinateur. Une variable a 2 caractéristiques : son nom et sa valeur	
Int	**nombre entier**		*a=5 b=56 c=2345*
Float	**nombre à virgule**	S'écrit avec un point (et non une virgule)	*a=2.23 b=6.1 c=0.12*
Str	**Chaîne de caractères**	S'écrit entre « » ou entre ' '	*a= « bonjour » b=« 45 » c= 'coucou'*
		Délimiter par des apostrophes une chaîne contenant des guillemets (mais pas d'apostrophes).	*print('Elle a dit : "Non mais allo koi !". Lol.')* → *Elle a dit : "Non mais allo koi !". Lol.*
		Délimiter par des guillemets une chaîne contenant des apostrophes (mais pas de guillemets).	*print("C'est celui qui dit qui y est !")* → *C'est celui qui dit qui y est !*
Bool	**Booléen**	Objet qui ne prend que 2 valeurs : soit **True** (vrai), soit **False** (faux)	Print(3>2) → *True*

Liste de variables			Créer des variables ordonnées dans une liste.	**liste[**«nom**1**»,«nom**2**»,«nom**3**»,«nom**4** »**]**
afficher			Les variables d'une liste sont ordonnées selon un nombre **commençant par 0** (**première variable**), puis 1,2,3, etc ….	Liste → «nom**1**»,«nom**2**»,«nom**3**»,«nom**4** » Liste**[3]** → « nom**4** » Liste**[0]** → « nom**1** »
ajouter		**append**	Ajoute une variable en fin de liste	**liste.append**("ok") → «nom**1**»,«nom**2**»,«nom**3**»,«nom**4** », «ok »
supprimer		**del**	Supprime une variable dans une liste	**Del liste[2]** → «nom**1**»,«nom**2**»,«nom**4** »
Nombre éléments		**len**	Nombre d'éléments dans la liste	**Len**(liste) → 5
Boucle sur liste	**for** var **in liste:**		Boucle de lecture des variables	**For var in liste :** **print var** → «nom**1**»,«nom**2**»,«nom**3**»,«nom**4** »

Opérateurs		Un opérateur est un symbole (ou un mot réservé) utilisé pour effectuer une opération entre 2 opérandes			
=	**Affectation**	instruction qui attribue une valeur à une variable : variable ← valeur		*Var1 = « toto »*	*var2=34*
*****	**Multiplication**	Int ou Float	Multiplication	*2 * 5.0* → *10.0*	
		Int et str	Accole les caractères autant de fois que le nombre	*3 * "ab"* → *"ababab"*	
		Str et str	Provoque une erreur	*"Pi" * "Pi"* → *erreur*	
/	**Division**	Int ou Float	renvoie le résultat toujours avec un type float (virgule)	*2 / 4* → *0.5*	
+	**Addition**	Int ou Float	Additionne les nombres	*2 + 3* → *5*	
		Str et Str	Accole les chaînes de caractères (concaténation)	*"A" + "ne"* → *"Ane"*	
-	**Soustraction**	Int ou Float	soustraction	2 - 3 → -1	
******	**Puissance**	Int ou Float	Premier nombre à la puissance du nombre à droite	*4**3* → *64 (4x4x4)*	
//	**Quotient entier**	Int ou Float	Renvoie la partie entière du quotient	5 // 3 → 1 (entier)	5.0 // 3 → 1.0 (float)
%	**Reste (ou Modulo)**	Int ou Float	Renvoie le reste de la division (entier int quand tous les opérandes sont int, sinon réel float).	5 % 3 → 2 (entier)	5.1 % 3 → 2.1 (float)

Opérateurs Logiques		Les expressions avec un opérateur logique renvoient toutes un booléen	
and	**ET logique**	➢ Renvoie True seulement si les 2 opérandes sont vrais (V et V) ➢ Renvoie False dans tous les autres cas (V et F, F et V, F et F)	(1 > 0) and (1 < 2) → True (1 > 0) and (1 > 2) → False
or	**OU logique**	➢ Renvoie False seulement si les 2 opérandes sont faux (F or F) ➢ Renvoie True dans les 3 autres cas (V or F, F or V, V or V).	(1 > 0) and (1 < 2) → True (1 > 0) and (1 > 2) → True
not	**Non logique**	Renvoie l'opposé booléen (la négation)	not (1 > 3) →True

Opérateurs de comparaison		Une expression avec un opérateur de comparaison pose une question dont la réponse est soit True, soit False.		
= =	**Egalité de valeur ?**	Compare 2 objets de « même » type pour voir s'ils ont la même valeur.	3 = = 1 + 2 → True	"1" = = 1 → False
!=	**Différents ?**	Contraire de ==. A != B équivaut donc à not (A = = B).	3!= 1 + 2 → False	"1" != 1 → True
<, <= **> , > =**	**Inférieur ? ou égal ?** **Supérieur ? ou égal ?**	Ordre naturel des nombres et des lettres avec (MAJUSCULES< minuscules)	1 < 0 → False	'fdb'> = 'fde' → True
is **is not**	**Identiques ?** **Non identiques ?**	Retourne True si A et B sont exactement le même objet. L'opérateur is est plus restrictif que l'opérateur = =	1.0 is 1 → False	1 + 2 is not 3 → False
in **not in**	**Dans (pas dans) la séquence ?**	« k in Séquence » : renvoie True si la valeur de k est égale à la valeur de l'un des items de Séquence	'py' in 'papy' →True 1 in [0 , 'a' , 1.0] → True	'k' in ('ko', 1) → False

MEMENTO PYTHON

Affichage / Question			
print	**print("texte1",variable1, …)**	Affiche du texte à l'écran : elle ne retourne aucune valeur	**Print** ("Bonjour, ",var1," 3 novembre ",var2) (avec var1="nous sommes le " et var2=2024) → Bonjour, nous sommes le 3 novembre 2024
input	**var=input("question ?")**	Renvoie la valeur saisie par l'utilisateur dans la variable nommée à gauche du « = » suite à la question formulée entre parenthèse. La réponse revient toujours en format texte. Si la réponse attendue est un nombre, il faut mettre int() ou float() avant.	var1= **input**(« ton nom ? ») Le mot réponse est mis dans var1
	var=int(input("question ?"))		Var2= **int(input**(« ton age ? »)) Le nombre entier mis dans var2
	var=float(input("questio"))		Var3= **float(input**(« ton poid ? »)) Le nombre décimal est mis dans var3

Boucle for	On utilise la **boucle for** quand on veut répéter des actions similaires (calcul, affichage, etc) et que l'on connait le nombre de fois que l'on veut faire cette répétition.		
	Explication	Exemple de programme	Résultat du programme
for i in range(a,b,c) : **Instructions** 1ere instruction hors boucle	Les instructions vont être répétées avec la variable i qui va évoluée en partant de la valeur **début (a)** jusqu'à la valeur **fin (b)** avec une progression d'un **« pas » (c)** à chaque itération.	For i in range (3,12,2) Print("la valeur de i est ",i) Print("fin de la boucle")	La valeur de i est 3 La valeur de i est 5 La valeur de i est 7 Fin de la boucle
For i in range(a)	Les valeurs **début** et le **«pas»** sont omis. Elles prennent la valeur de 1 par défaut	For i in range (5)	i prendra les valeurs : 1,2,3,4,5
For i in range(a,b)	La valeur du **«pas»** est omis. Elle prendra la valeur de 1 par défaut.	For i in range (3,8)	i prendra les valeurs : 3,4,5,6,7,8

Boucle While	On utilise la **boucle while** quand on veut répéter des actions similaires (calcul, affichage, etc) et que le nombre de fois que l'on veut faire cette répétition n'est pas connu à l'avance : Elle dépend d'une condition qui est vérifiér (true) ou non (false). Si la condition est vérifiée (true), la boucle continue, si elle ne l'est pas (false), on arrête la boucle.		
	Explication	Exemple de programme	Résultat du programme
while condition: **Instructions** 1ere instruction hors boucle	Les instructions vont être répétées tant que la condition est vérifiée (c'est-à-dire que le résultat de la condition est à true).	i=2 While i < 5 : Print("la valeur de i est ",i) i = i + 1 Print("fin de la boucle")	La valeur de i est 2 La valeur de i est 3 La valeur de i est 4 Fin de la boucle

IF Else	Une **instruction conditionnelle**, ou instruction de test, permet de *faire des choix* en fonction de la valeur d'une *condition*. On parle souvent d'une instruction *si-alors*. Une **condition** est une instruction qui est soit vraie, soit fausse. On dit qu'il s'agit d'un *booléen*. → Si le résultat est True, alors les instructions suivant la condition **If** sont exécutées. → Si le résultat est False, alors les instructions suivant la condition **Else** sont exécutées. Le bloc **else** n'est pas obligatoire		
	Explication	Exemple de programme	Résultat du programme
if condition: **Instructions** **else:** **Instructions**	if condition : instructions à effectuer dans le cas où la condition est remplie else : instructions à effectuer dans le cas contraire	moyenne = 14 if moyenne < 10 : print("moyenne insuffisante") else : print("moyenne satisfaisante")	moyenne satisfaisante

fonction	Une fonction est une série de lignes de programmation que l'on pourra appeler dans un programme		
		Exemple de programme	Résultat du programme
Def nomfonction(param1,param2,param3,..) : Lignes de programmations utilisant param1,2,et 3. **Return resultat**		def **calculage**(**jour**,**mois**,**année**) Lignes de calcul de l'age Return **age**	L'appel à la fonction **calculage** calculera et retournera l'**age** d'une personne à partir de ses **jour**, **mois**, et **année** de naissance
Réponse= nomfonction(param1,param2,param3,..) :		Le programme appelant **nomfonction** recevra l'age dans la variable **réponse**	Réponse=calculage(24,08,2004) La fonction **calculage** va calculer l'âge et alimenter **réponse** à la valeur **20** (ans)