1 Mémo Python

Les commandes de base

Entrées, sorties, calculs, opérateurs

Remarque 1 : L'éditeur MU nécessite des espaces autour des opérateurs (=, <, >, ...) et après une virgule.

Saisir une variable **a** de type décimale (**float**) avec texte de demande	**a = float(input("**Choisir une valeur : **"))**
Saisir une variable **a** de type entière (**int**) avec texte de demande	**a = int(input("**Saisir un nombre entier : **"))**
Saisir une variable **a** de type caractère (**str**) avec texte de demande	**a = str(input("**Saisir votre prénom : **"))**
Afficher : → du texte → la valeur d'une variable a → un mélange de texte et de variables (séparées par des virgules)	**print("**Vive les maths**")** **print(a)** **print("**La valeur de a est :**", a)** **print("**La distance est de **", d,**"cm")**
Affecter à **pi** la valeur 3,14 Affecter à **b** la valeur a Affecter à **c** le calcul 2*a+b	**pi = 3.14** **b = a** **c = 2*a+b**
Additionner ; Soustraire ; Multiplier ; Diviser ; Puissance	**+** **–** ***** **/** ******
Calculer a^3	**a**3**
Arrondir le nombre **a** à **n** décimales	**round(a, n)**

Tests conditionnels

Remarque 2 : Certaines commandes (if, for, while, ...) comportent différentes instructions qui doivent être décalées, on appelle cela l'indentation. Après validation, ce décalage se fait automatiquement après ":"

Tester une *condition* et, si elle est vraie, exécuter une suite d'instructions	**if** *condition*:
	  instruction1
	  instruction2	

Tester une *condition*, si elle est vraie, exécuter les instructions 1 et 2, si elle est fausse (**else**), exécuter les instructions 3 et 4
On peut aussi intercaler la commande **elif** (*sinon si* ...) autant de fois que nécessaire pour introduire d'autres conditions.
	if *condition*:
	  instruction1
	  instruction2
	else:
	  instruction3
	  instruction4
		if *condition1*:
		  instruction1
		elif *condition2*:
		  instruction2
		elif condition3:
		  instruction3
		...
		else:
		  instruction4

Tester la *condition 1* **et** la *condition 2* : le test est vrai si les 2 conditions sont vraies	**if** *condition 1* **and** *condition 2*:	
Tester la *condition 1* **ou** *la condition 2* : le test est vrai si au moins une condition est vraie	**if** *condition 1* **or** *condition 2*:	
Tester *être égal à* ; *être différent de* ; *être inférieur à* ; *être supérieur à* ; *être inférieur ou égal à* ; *être supérieur ou égal à*	**==** **!=** **<** **>** **<=** **>=**	

Les boucles			
Boucle bornée : Répéter **n** fois une suite d'instructions, la variable **i** parcourt tous les entiers : → de **0** à **n-1** → de **a** à **b-1** → de **m** à **n-1** avec un pas **p**	**for** i **in range(**n**):** *instruction1* *instruction2*	**for** i **in range(**a, b**):** *instruction1* *instruction2*	**for** i **in range(**m, n, p**):** *instruction1* *instruction2*
Boucle non bornée : Répéter une suite d'instructions → **tant que** la *condition* est vraie (**while**) → **jusqu'à ce qu**'elle soit vraie (**while not**)		**while** *condition***:** *instruction1* *instruction2*	**while not** *condition***:** *instruction1* *instruction2*

Listes	
Déclarer une variable **L** de type liste vide	**L = []** *ou* **L = list()**
Définir une liste **L**	**L = [1, 2, 3, 4]** *ou* **L = ['A', 'B', 'C', 'D']**
Déterminer la longueur **n** d'une liste **L**	**n = len(L)**
Somme **s** des termes d'une liste **L**	**s = sum(L)**
Renvoie le nombre **n** d'éléments ayant la valeur **x** dans la liste **L**	**n = L.count(x)**
Ajouter un terme **x** à une liste **L**	**L.append(x)**
Eléments d'une liste **L**	**L[0]**, **L[1]**, **L[2]**, ...

Commande import (voir Mémo 2)	
Remarque 3 : Certaines commandes ou variables n'étant pas présentes dans la version de base, il peut être fait appel à des modules sous forme d'une importation placée en début de programme.	
Importer la totalité du module math, random, ...	**import** math **import** random ...
Module math : Racine carrée d'un nombre **a**	**math.sqrt(a)**
Module random : Générer un nombre entier aléatoire compris entre **a** et **b**	**random.randint(a, b)**