2de TD n°3 : utilisation de la calculatrice en mode « Programme »

Algorithmes et programmes : Un *algorithme* est un ensemble d'instructions structuré de manière à atteindre un but. Ces instructions manipulent des données (nombres, textes, etc.) : entrées, stockages en mémoire, calculs, affichages. Comme un algorithme est souvent exécuté par une machine (calculatrice, ordinateur, robot), il doit être traduit dans un langage que cette machine comprend : les *programmes* pour calculatrices *Casio* et *TI* utilisent une syntaxe dérivée du langage « Basic » (le Basic Casio est différent du Basic TI) tandis que les programmes de la calculatrice *Numworks* sont en « Python » (un autre langage de programmation, utilisé plutôt sur ordinateur).

Le mode « programmation » de la calculatrice :

Casio : On y entre avec la touche PRGM. Pour commencer un nouveau programme, sélectionner NEW (bouton F3), taper ensuite un nom pour le programme (ALGO1 par exemple) et EXE. Il ne reste plus qu'à écrire le programme. Pour modifier un programme existant, choisir EDIT dans le menu PRGM ; pour l'exécution, choisir EXE.

TI : On y entre avec la touche « prgm ». Pour commencer un nouveau programme, sélectionner NOUV, taper ensuite un nom pour le programme (ALGO1 par exemple), valider avec Entrer. Il ne reste plus qu'à écrire le programme. Pour modifier un programme existant, choisir EDIT dans le menu prgm ; pour l'exécution, taper Entrer deux fois.

Numworks : Faire la mise à jour (module Python indisponible sans cela). L'accès est simple depuis le menu « Accueil ». Il est précisé que c'est une version bêta (test) de Python qui est limitée et va bientôt être améliorée. On commence par éditer le programme (un seul à la fois, pas besoin donc de le nommer)

1) Entrées, affectations, sorties

	Algorithme	Basic Casio	Basic TI	Python Numworks
Affectation : instruction qui place une valeur en mémoire	A=A+1	A+1→A	A+1 $^{STO}\rightarrow$ A	a=a+1
Entrée (lecture) : affectation qui utilise la saisie de l'utilisateur	Saisir A	?→A	Input A	a=float(input())
Initialisation : première affectation pour une variable	A=0	0→A	0 $^{STO}\rightarrow$ A	a=0
Sortie (écriture) : instruction qui affiche une donnée.	Afficher A	A◢	Disp A	print(a)

Remarques : L'instruction A=A+1 signifie que l'on augmente de 1 la valeur de A. On note parfois cela A:=A+1 ou bien A←A+1 ou encore A+1→A. Nous avons choisi la notation la plus simple (celle de Python).
En Python, il faut convertir une entrée numérique (par défaut, une chaîne de caractères) en entier ou en flottant (nombre avec virgule). Utiliser l'instruction int() ou float() selon le but recherché. Dans la version actuelle, cette instruction n'est pas disponible. Taper l'initialisation souhaitée directement dans le programme (ex : A=0,39)

Avec ces instructions, on va écrire un algorithme qui calcule l'image y d'un nombre x (donnée saisie en entrée) par une fonction f (on va prendre ici $f(x)=\frac{1}{(5x-2)(2-x)}$) et qui affiche le résultat.

- Traduire l'algorithme simplissime ci-dessous dans le langage de votre calculatrice

Saisir A	*Entrée de la valeur de A*
A=1÷((5A-2)×(2-A))	*Affectation de la nouvelle valeur de A*
Afficher A	*Sortie de la valeur de A*

- Programmer cet algorithme sur votre calculatrice, puis tester le avec la valeur x=1 (on trouve $y=\frac{1}{3}$, soit ≈0,3333)
- Compléter le tableau de données

x	0	0,39	0,41	0,75	1	1,5	1,99	2,01	2,5	3
$f(x)$										

2) Boucles

Une boucle permet d'exécuter plusieurs fois une succession d'instructions. Il en existe deux sortes :

	Algorithme	Basic Casio	Basic TI	Python Numworks
Boucle « pour » : on fait un nombre fixé (N) de tours	Pour I allant de 1 à N : [instructions] fin de la boucle pour	For 1→I To N [instructions] Next	For (I,1,N) [instructions] End	For i in range(n) : [instructions] ↑ noter le décalage (indentation)
Boucle « tant que » : on fait des tours tant qu'une condition est vraie	Tant que I<N : [instructions] fin de la boucle while	While I<N [instructions] WEnd	While I<N [instructions] End	While i<n : [instructions] ↑ noter le décalage (indentation)

Remarques : En Python, il y a deux points à la fin de l'en-tête de boucle ; ensuite il faut une indentation (un seul espace suffit). Avantage : il n'y a pas de marque de fin de boucle, juste un retour sans indentation. La syntaxe Python range(n) correspond aux valeurs 0,1,2,...,(n−1). Si on veut aller de 1 à n (pas seulement n valeurs), il faut écrire range(1,n+1).
D'une façon générale, l'instruction « For I... » gère l'incrémentation de la variable I (elle augmente de 1 à chaque tour) ; l'instruction « While I... » ne fait pas cela. Si on utilise une variable I qui doit être incrémentée à chaque tour, il faut le faire dans les instructions (I=I+1).

a) L'algorithme suivant calcule et affiche la somme s des n premiers carrés d'entiers : $s=1^2+2^2+3^2+...+n^2$:

- Traduire l'algorithme ci-dessous dans le langage de votre calculatrice

Saisir N	*Entrée de la valeur de N*	
S=0	*Initialisation de la valeur de S*	
Pour I allant de 1 à N	*En-tête de la boucle « Pour »*	
S=S+I²	*Affectation de la nouvelle valeur de S*	
Fin de boucle	*Marque de fin de la boucle « Pour »*	
Afficher S	*Sortie de la valeur de S*	

*Remarque : En Python, le carré se note i**2 (2 fois le signe de la multiplication)*

- Programmer cet algorithme sur votre calculatrice. Tester le programme avec la valeur n=5 (on trouve s=55). Calculer $S_{100}=1^2+2^2+3^2+...+100^2$: ..... ; $S_{1000}=1^2+2^2+3^2+...+1000^2$: .....

b) L'algorithme suivant détermine la valeur de n à partir de laquelle la somme $s=1^2+2^2+...+n^2$ dépasse un nombre m donné (m étant une donnée saisie par l'utilisateur) et qui affiche cette valeur de n.

- Traduire l'algorithme ci-dessous dans le langage de votre calculatrice

Saisir M	*Entrée du maximum M*	
I=0	*Initialisation du compteur I*	
S=0	*Initialisation de la somme S*	
Tant que S<M	*En-tête de la boucle « While »*	
I=1+I	*Incrémentation du compteur*	
S=S+I²	*Affectation de la nouvelle valeur de S*	
fin de la boucle « tant que »	*Marque de fin de la boucle « While »*	
Afficher I	*Sortie de la valeur de I*	
Afficher S	*Sortie de la valeur de S*	

- Programmer cet algorithme sur votre calculatrice (modifier le programme précédent ou en écrire un nouveau). Tester le programme avec la valeur m= 50 (on trouve n=5).
- Calculer n pour m= 10^6 : n=.....

3) Instructions conditionnelles

	Algorithme	Basic Casio	Basic TI	Python Numworks

Si une condition (I<N) est vraie on exécute les instructions I_1, sinon (facultatif) on exécute les instructions I_2.	Si I<N
	alors I_1
	sinon I_2
	fin du « Si »
		IF I<N
		Then I_1
		Else I_2
		IfEnd
			IF I<N
			Then I_1
			ELSE I_2
			End
				If i<n :
				  I_1
				else :
				  I_2

a) Test simple

- Écrire l'algorithme qui demande d'entrer un nombre x (entre 0 et 10) et qui calcule l'aire du polygone restant (situation vue en cours). Traduire ensuite l'algorithme dans le langage de votre calculatrice.

Rappel : $A(x)=100-2x^2$ si $x\leq 5$ et $A(x)=100-2x^2+4(x-5)^2$ si $x>5$.

Entrée de la valeur de X *Test sur la valeur de X* *Calcul de A(x) dans le cas x≤5* *Calcul de A(x) dans le cas x>5* *Sortie de la valeur de A(x)*	

- Programmer cet algorithme sur votre calculatrice.
  Compléter le tableau de données de la fonction A.

x	0	1	2	3	4	5	6	7	8	9	10
$A(x)$											

Remarque : Dans une boucle « While » ou une instruction conditionnelle, on utilise généralement un test simple, c'est-à-dire un test qui utilise un des symboles <, >, =, ≤, ≥ ou ≠. On peut associer des tests simples pour examiner des situations plus complexes :

- dans le test $x > 0$ et $i < 100$ (et s'écrit *and*), pour effectuer I_1, il faut que *les deux* conditions soient vraies
- dans le test $x > 0$ ou $x < -1$ (ou s'écrit *or*), pour effectuer I_1, il faut qu'*une au moins* des conditions soit vraie

b) Application

On veut calculer la somme $Z(n)= 1-\frac{1}{2}+\frac{1}{3}-\frac{1}{4}+\frac{1}{5}-....$ (somme des inverses des entiers successifs affectés alternativement d'un signe + ou d'un signe –) pour différentes valeurs de l'entier n.

- Écrire un algorithme qui utilise un test (le reste de la division euclidienne de n par 2 est noté n%2) et un qui n'utilise pas de test (penser à la fonction $n \mapsto (-1)^n$)
- Traduire un des deux algorithmes en programme pour votre calculatrice et déterminer Z(100).