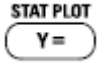
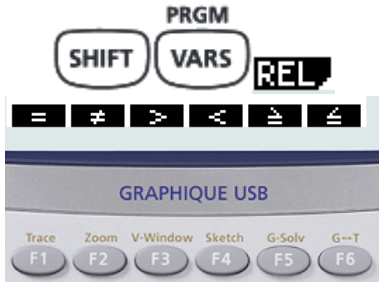
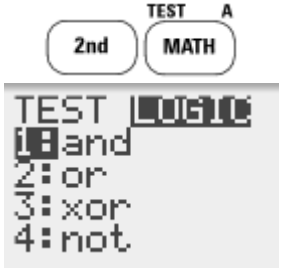
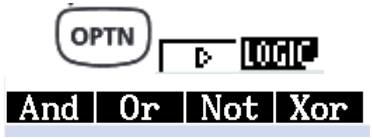
Langage algorithmique	Langages de programmation			
	Sur TI	**Sur Casio**	**Logiciel Algobox**	**Logiciel Xcas**
Déclarer une variable A	Inutile	Inutile	Déclarer nouvelle variable	`local A ;`

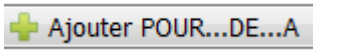
Saisir A	
	`Input "A=",A`
	ou
	`Prompt A`
		
		`"A=":?→ A`
			Ajouter LIRE variable	`saisir("Entrer A",A);`
				ou
				`saisir(A);`
				ou si on a une fonction :
				`nom_programme(A):={`
				`instruction(s); }:;`

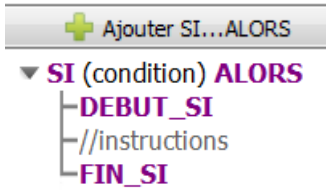
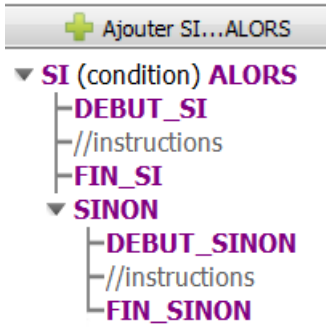
Afficher A	
	`Disp"A=",A`
		
		`"A=":A◢`
			Ajouter AFFICHER Variable	`afficher("A vaut :",A);`
				ou
				`afficher(A);`
				ou si on a une fonction :
				`nom_programme(paramètres):={`
				`instruction(s);`
				`retourne A;}:;`

Affecter à A la valeur val	 `val→A`	 `val→A`	AFFECTER valeur à variable	`A:=val;`

Utiliser une fonction externe dans un programme	Saisir la fonction dans l'éditeur graphique
	 
	 
	puis la rappeler dans un programme :
	 
	$Y_1(...)$
		Saisir la fonction dans le menu  ou 
		
		puis la rappeler dans un programme :
		 
		`Y1(...)`
			cliquer sur l'onglet :
			Utiliser une fonction numérique
			Saisir la fonction :
			☑ Utiliser la fonction F1
			F1(x)= pow(x,3)-x-1
			puis la rappeler dans un programme :
			`F1(...)`
				Définir la fonction (3 méthodes) :
				`f(x):=x^3-x-1`
				`f:=x->x^3-x-1`
				`f:=unapply(x^3-x-1,x)`
				On peut aussi utiliser une fonction comme variable d'un programme :
				`nom_programme():={`
				`local f,...;`
				`saisir(f);...}:;` Dans ce cas il faudra saisir dans l'invite :
				`x->...`

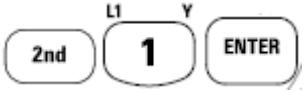
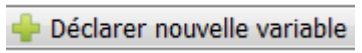
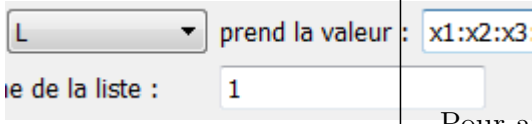
Langage algorithmique	Langages de programmation			
	Sur TI	Sur Casio	Logiciel Algobox	Logiciel Xcas
Opérateurs de test et de logique				
Opérateurs de tests $=$, $\neq$, $>$, $<$, $\geqslant$, $\leqslant$			• "$x = 2$" s'écrit `x==2` • "$x \neq 2$" s'écrit `x!=2` • "$x < 2$" s'écrit `x<2` • "$x > 2$" s'écrit `x>2` • "$x \leqslant 2$" s'écrit `x<=2` • "$x \geqslant 2$" s'écrit `x>=2`	• "$x = 2$" s'écrit `x==2` • "$x \neq 2$" s'écrit `x!=2` • "$x < 2$" s'écrit `x<2` • "$x > 2$" s'écrit `x>2` • "$x \leqslant 2$" s'écrit `x<=2` • "$x \geqslant 2$" s'écrit `x>=2`
Opérateurs logiques et, ou, ou exclusif, non			• le "et" s'écrit `ET` • le "ou" s'écrit `OU`	• le "et" s'écrit `et` • le "ou" s'écrit `ou` • le "ou exclusif" s'écrit `xor` • le non s'écrit `non`
Boucle `Pour ...de ...jusque ...faire ...Fpour`				

Pour I de 1 jusque N
faire
instructions
Fpour
	
	`For(I,1,N)`
	instructions
	`End`
		
		`For 1→I To N`
		instructions
		`Next`
			Il faudra déclarer auparavant la variable `I`
			 
			POUR I ALLANT_DE 1 A N
			DEBUT_POUR
			//instructions
			FIN_POUR
				`pour j de 1jusque N faire`
				instructions;
				`fpour ;`
				Ne pas utiliser la variable `i` comme compteur car c'est une lettre prédéfinie qui désigne le `i` des complexes.

Langage algorithmique	Langages de programmation			
	Sur TI	Sur Casio	Logiciel Algobox	Logiciel Xcas
Instruction conditionnelle **Si...alors...[Sinon]...Fsi**				
Si *conditions* alors *instructions* Fsi	 **If** *conditions* **Then** *instructions* **End**	 **If** *conditions* **Then** *instructions* **IfEnd**		**si** *conditions* **alors** *instructions* ; **fsi** ;
Si *conditions* alors *instructions* Sinon *instructions* Fsi	 **If** *conditions* **Then** *instructions* **Else** *instructions* **End**	 **If** *conditions* **Then** *instructions* **Else** *instructions* **IfEnd**		**si** *conditions* **alors** *instructions*; **sinon** *instructions*; **fsi** ;
Boucle avec arrêt conditionnel **Tantque ...faire ...Ftantque**				
Tant que *conditions* faire *instructions* Ftantque	 **While** *condition* *instructions* **End**	 **While** *condition* *instructions* **WhileEnd**		**tantque** *condition* **faire** *instructions*; **ftantque** ;

Langage algorithmique	Langages de programmation			
	Sur TI	Sur Casio	Logiciel Algobox	Logiciel Xcas
Fonctions mathématiques				
Racine carrée $\sqrt{x}$	`√x`	`√x`	`sqrt(x)`	`sqrt(x)`
Puissance x^n	`x^n`	`x^n`	`pow(x,n)`	`x^n`
Partie entière de x	TEST A MATH NUM `int(x)`	OPTN ▷ NUM `Intg(x)`	`floor(x)`	`floor(x)`
Arrondi à l'unité de x	TEST A MATH NUM `round(x,0)`	OPTN ▷ NUM RndFi `RndFix(x,0)`	`round(x)`	`round(x)`
Reste de la division euclidienne de A par B	`A-B*int(A/B)`	`MOD(A,B)` (certaines calculatrices) `A-B*Intg(A÷B)`	`A%B`	`irem(A,B)`
Logarithme népérien de x : $\ln(x)$	`ln(x)`	`ln(x)`	`log(x)`	`ln(x)`
Exponentielle de e^x	`e^x`	`e^x`	`exp(x)`	`exp(x)`
Nombre réel pseudo-aléatoire dans $[0\,;\,1[$	TEST A MATH PRB `rand`	OPTN ▷ PROB `Rand#`	`random()`	`rand(0,1)`

Entier aléatoire dans $[\![a\,;\,b]\!]$, avec a et b deux entiers donnés	avec la partie entière :
	`a+int((b-a+1)*rand)`
		avec la partie entière :
		`a+Intg((b-a+1)*Rand#)`
			`ALGOBOX_ALEA_ENT(a,b)`
			ou
			`a+floor((b-a+1)*random())`
				`a+rand(b-a+1)`

Langage algorithmique	Langages de programmation			
	Sur TI	**Sur Casio**	**Logiciel Algobox**	**Logiciel Xcas**
Listes				
Créer et remplir une liste	Les listes $\mathtt{L_1}$,$\mathtt{L_2}$,...$\mathtt{L_2}$ existent déjà dans le mode **STAT** :  On peut donc les remplir directement avec ce menu. Cela peut se faire aussi dans le menu courant avec la commande : $\mathtt{\{x_1,...,x_n\}\rightarrow L_1}$ On peut l'afficher dans le menu courant en tapant : 	Les listes **List 1**, **List 2**,..., **List 26** existent déjà dans le menu **STAT** :  On peut donc les remplir directement avec ce menu. Cela peut se faire aussi dans le menu courant avec la commande : $\mathtt{\{x_1,...,x_n\}\rightarrow List\ 1}$ On peut l'afficher dans le menu courant en tapant : 	 puis préciser le type **Liste**. Pour la remplir :  puis  en mettant **1** au rang de la liste et en séparant chaque valeur par **:** Pour afficher le contenu d'une liste, on utilise une boucle.	Pour créer une liste $\mathtt{L:=[x_1,...,x_n]}$ Pour afficher le contenu d'une liste : **retourne L**
Élément de rang k d'une liste	Le premier rang d'une liste $\mathtt{L_1}$ est 1 et le dernier rang est $\mathtt{Dim(L_1)}$. $\mathtt{L_1(k)}$ est le terme de rang k de la liste 1.	Le premier rang d'une liste **List 1** est 1 et le dernier rang est **Dim List 1**. **List 1[k]** est le terme de rang k de la liste 1.	**L[1]** est le premier terme de la liste **L** (on peut débuter à 0 : **L[0]**). **L[k]** est le terme de rang k de la liste **L**. La longueur d'une liste commençant à **1** est donnée par **L.length-1**	 **L[0]** ou **L(1)** désignent le premier terme de la liste **L**. **L[k]** est le terme de rang k de la liste **L** donc le $(k+1)$-ème terme de cette liste La longueur d'une liste est donnée par **dim(L)**
Remplir une liste avec p entiers aléatoires pris dans $[\![a\,;b]\!]$, avec a et b deux entiers donnés	Avec la commande **seq**  **seq(a+int((b-a+1)* rand),K,1,p,1)→L1**	Avec la commande **Seq**  **seq(a+Intg((b-a+1)* Rand#),K,1,p,1)→List 1**	Il faut créer une boucle pour remplir la liste terme après terme : POUR k ALLANT_DE 1 A p DEBUT_POUR L[k] PREND_LA_VALEUR ALGOBOX_ALEA_ENT(a,b) FIN_POUR	**L:=** **[(a+rand(b-a+1))$(k=1..p)]**