Tableau des principales syntaxes pour le lycée : Scilab, Python, TI, Casio, Xcas

Sommaire

Entrées-Sorties, instructions conditionnelles, répétitions 2

Nombres aléatoires, listes 3

Graphiques, constantes, fonctions 4

Illustrations sur le site planète maths : fiche thématique Algorithmique et logiciels

Entrées-Sorties - instructions conditionnelles - répétitions

	Langage algorithmique					
	Scilab Certaines fonctions nécessitent l'installation du module « lycée » **Le point virgule permet d'écrire plusieurs instructions sur la même ligne, Il supprime aussi l'affichage**	**Python 2.6** ***3.x seulement*** **Attention : en v 2.6, 3/2 vaut 1, et non 1,5**	**TI** **82-84**	**Casio** **35+** **(non USB)**	**XCAS 0.8.5** **Le point virgule sépare les instructions**	
Saisir *a*	*a*=input ("donner a ")	*a*=input("donner a ") a=float(input("réel a ?")) a=int(input("entier a ?"))	Prompt A Input "X1=", X	"A=":?→A dans shift PRGM	input("a= ",a)	saisir ("a= ",a)
Afficher *a* *(Xcas : Unquoted : sans guillemets)*	*a ou afficher(" a= "+string(a))* *ou disp(" a= "+string(a))*	V 2.6 : print «'a= ', *a* *V 3.x : print ('a= ', a)*	Disp "A=" , A	"A= " : A◢ dans shift PRGM	print ("a=",a) print ("a=",a)	afficher ("a=",a) afficher ("a=",a)
affectation: *a*→ *b* *b* prend la valeur *a,*	*b = a*	*b = a*	A STO B	A→ *B* *touche directe*	b := a	
Tests, logique =, ≠ , ≤ , ≥ et, ou, non, ou exclusif,	== , <> , <= , >= **&** , **|** (AltGr+6) , **~** (AltGr+2),voir	==, != , <=, >= and, or, not, xor	Menu 2nd TEST =, ≠ , ≤ , ≥ and, or, not, xor	=, ≠ , ≤ , ≥ Dans shift PRGM REL and, or, not dans OPTN LOGIC	=, != , <=, >= and, or, not, xor	
Bloc d'instructions	Les blocs sont définis par la structure	Les instructions d'un bloc ont la même marge à gauche (instructions indentées)	Le bloc est terminé par End	Le bloc est terminé par End, ifEnd, whileEnd,…	Le bloc est encadré par des accolades: {*instructions* }, **sauf** s'il est encadré par la structure, voir ci-dessous. Pour ne pas se tromper utiliser le menu **Add**	
Si *condition* Alors *Instruction*1 *Facultatif* [Sinon *Intruction*2] Fin du si *Condition2 est la négation de condition1*	if *condition* then *Instructions1* else *Intructions2* *end* (if et then doivent être sur la même ligne)	if *condition*: *Instruction*1 [else: *Intruction*2]	If *condition* Then *Instructions*1 [Else *Intructions*2] End	If *condition* Then *Instructions*1 [Else *Intructions*2] IfEnd dans PRGM COM	if (*condition*) then *{Instructions1}* [else *{Intructions2}]*	si (condition) alors *Instructions1* [sinon *Intructions2]* *fsi*
Répéter *Instruction(s)* Jusqu'à *condition*1	while *%T* then *Instruction(s)* if *condition* then break end	while True *:* *Instruction(s)* if *condition* : break	Repeat *condtion*1 *Instruction(s)* End	Do *Instruction(s)* LpWhile *condition*2 dans PRGM COM	repeat *instruction(s)* until *(condition*1)	repeter *instruction(s)* jusqu_a *(condition*1)
Tant *condition* *Instruction(s)* Fin TantQue	while *condition* then *Instruction(s)* *end*	while *condition* : *Instruction(s)*	While *condtion* *Instruction(s)* End	While *condition* *Instruction(s)* Endwhile dans PRGM COM	while (*condition)* {*Instruction(s)*}	tantque *(condition)* faire *instruction(s)* ftantque;
Pour i variant de 1 à *n* Faire *Instruction(s)* Fin du pour	for i=1:n *Instruction(s)* end	for i in range(1,n+1): *Instruction(s)*	For (I,1,N) *Instruction(s)* End	For 1->A To 10 *Instruction(s)* Next dans PRGM COM	for j from 1 to n do *instruction(s)* end_for (ne pas utiliser « *i* »)	pour j de 1 jusque n faire *instruction(s)* fpour (éviter la lettre *i*)

Nombres aléatoires - Listes

	Scilab	Python 2.6 *3.x*	TI	Casio	XCAS
Nombres aléatoires	**Module Lycée**	**Avec from random import***	**Touche MATH/ PRB**	**Run / touche OPTN / PROB**	**Le point virgule sépare les instructions**
Initialisation	rand("seed")	seed()			srand
Nombre **réel** aléatoire dans [0;1[	`tirage_reel (1,0,1)` *(liste de 1 seul réel aléatoire)*	random()	**rand**	**Rand#**	rand(0,1) ou alea(0,1)
Nombre réel aléatoire dans [*a;b*[	`tirage_reel(p,a,b)` *(liste de p réels aléatoires)*	a + (b-a) x random() *uniform(a,b) dans [a,b]*	a + (b-a) x **rand**	a + (b-a) x **Rand#**	rand(a,b) ou alea (a,b)
Entier aléatoire dans {a ;a+1 ; … ; b} avec a et b entiers	`tirage_entier(p,a,b)` *(liste de p nombres entiers aléatoires)*	randint(a,b)	**rand(**a,b)	a + **Intg**((b-a+1) x **Rand#)**	a+rand(b-a+1) ou a+alea(b-a+1)
Exemple : entier aléatoire dans : {0 ;1} puis dans {1 ;2 ;3 ;4 ;5 ;6}	L= `tirage_entier (1,0,1)` *(liste de 1 seul entier aléatoire)* L= `tirage_entier (1,1,6)`	randint(0,1) randint(1,6)	**rand**(0,1) **randInt**(1,6)	**Intg**(2***Rand**#) 1+ **Intg**(6***Rand**#)	rand(2) ou alea(2) 1+rand(6) ou 1+alea(6)
LISTES					
Créer une liste	*En Scilab, les listes sont aussi appelées vecteurs, indice minimum : 1* *l*=[5 ,8,9] *l*(1) vaut 5, *l*(2) vaut 8…	*l*=[5 ,8,9] l'indice commence à zéro, *l*[0] vaut 5, *l*[1] vaut 8,…	Les listes L_1, L_2, existent dans le mode Statistique	Les listes **List 1, List 2** existent dans le menu STAT	*l*:=[5 ,8,9] l'indice commence à zéro, *l*[0] vaut 5, *l*[1] vaut 8,…
Vider une liste *l* Créer une liste vide (Scilab, python, Xcas)	*l* =[ ]	*l* =[ ]	**ClrList**	Menu *Stat* puis *DEL-A* *ou* {0}→ List 1	*l* := [ ]
Créer et remplir une liste de six 0 Créer *l* =[5²,7²,9²,11²,13²]	*l* = zeros (1,6) *Avec une boucle, ou…*	*Avec une boucle et la fonction : append*	6 **STO dim(**L_1**)** Seq(X^2,X,5,13,2) STO▸ L_1	6→Dim List 1 Dans OPTN LIST Seq(X^2,X,5,13,2) → List 1	*l* := [0$6] *ou* *l* := [0$(k=1..6)] *l*:= seq (k^2,k=5..13,2);
Ajouter un élément *a* à la fin de la liste *l*	*Si l comporte déjà n éléments :* *l* (*n*+1)=*a*	*l*.append(*a*)	*a* **STO** L_1(I) I étant le premier indice non encore utilisé	Dans le menu *List,* entrer directement l'élément *a* à la fin de la liste ! (inutilisable dans un programme)	*l* := append(*l*, *a*)
Accès à l'élément numéro k	*l* (k)	*l* [k]	L_1(k)	List1[k]	*l* [k]
Longueur d'une liste	taille(*l*)	len(*l*)	**dim** (L_1)	Dim List 1 Dans OPTN LIST	length (*l*)
Remplir une liste *l* avec dix nombres Aléatoires pris dans {1 ;2 ;3 ;4 ;5 ;6}	*l*=tirage_entier (10,1,6)	*Avec une boucle et append*	Seq(**randInt**(1,6),X,1,10,1)	Seq(1+ **Intg**(6***Rand**#),X,1,10,1) Seq : Dans OPTN LIST	*l* := [(1+rand(6))$(k=1..10)]

Graphiques – constantes - fonctions

	Quelques instructions pour les graphiques				
	Scilab	**Python 2.6 ou 3.x** ***from graphsecondev2_3 import**** télécharger le module	**TI 82-84**	**Casio 35+**	**XCAS 0.8.4**
Passer en mode grahphique / mode calcul	***automatique***	***fenetre****(xmin,xmax,ymin,ymax)* ***affiche*** *à la fin du pgm*	***DispGraph***	***DrawGraph*** Dans shift PRGM DISP Grph	***DispG*** ***DispHome***
Effacer l'écran Graphique	***Clf*** ***Attention : clear efface les fonctions !***		***ClrDraw*** *ou* ***EffDessin***	***ClrGraph*** Dans shift PRGM CLR	***ClrGraph*** ou ***efface***
Placer un point M(x,y)	***plot*** `(x,y,".")`	***point****(x,y[,couleur])*	***Pt-On****(x,y[,marque])*	***Plot*** *x,y* Dans shift Sketch(F4)	***point****(x,y)*
Tracer le segment [AB] avec A(x_A ; y_A) et B(x_B ; y_B)	***plot*** `([xA,xB],[yA,yB])` ***Attention à l'ordre!***	***segment*** *((x_A , y_A , x_B , y_B [,couleur])*	***Line****(x_A , y_A , x_B , y_B)* *ou* ***Ligne****(x_A , y_A , x_B , y_B)*	*F-**line*** x_A , y_A , x_B , y_B Dans shift Sketch(F4)	*A :=point(x_A , y_A) ;* *B:=point(x_B , y_B) ;* *segment (A,B) ;*

Tracer un cercle	`t=linspace(0,2*%pi,100);`
	`x=xo+r*cos(t);y=yo+r*sin(t);`
	`plot(x,y)`
		cercle_cr*(x,y,r[,couleur])*
		cercle_cp*(x,y,s,t[,couleur])*
			Circle*(x,y,r)*
			x et y coordonnées du centre et r le rayon
				Circle *x,y,r*
				x et y coordonnées du centre, et r le rayon
				Dans shift Sketch(F4)
					circle*(****point****(x,y),r)*
					voir autres possibilités dans l'index

		Quelques fonctions courantes - Constantes - Définir une fonction				
	Scilab	**Python 2.6** ***3.x seult.***	**TI**	**Casio**		**XCAS**
Racine carrée	sqrt	sqrt (from math import*)	touche directe	touche directe		sqrt
Puissance a^b	a^b	a**b ou pow(a,b) (from math import*)	a^b	a^b		a^b
Valeur absolue	abs	fabs (from math import*)	abs	Abs dans OPTN NUM		abs
ln, exp	log, exp	log exp (from math import*)	ln exp	ln exp		ln exp
Partie entière(*) , plus grand entier relatif inférieur ou égal au nombre considéré	floor	floor (from math import*)	Int dans MATH NUM	Intg dans OPTN / NUM		floor
Troncature(*), c'est-à-dire nombre sans sa partie décimale éventuelle	int	trunc (from math import*)	iPart dans MATH NUM	Int dans OPTN / NUM		iPart
Quotient division euclidienne	quotient (a,b)	*V 2.6* a/b *v 3.x a//b* (avec a et b entiers)	Int(A/B)	Intg(A/B)		iquo(a,b)
Reste division euclidienne	reste(a,b)	a%b , ou fmod(a,b) (from math import*)	A-B* Int(A/B)	A-B* Intg(A/B)		irem(a,b)
pi, e, i	*%pi, %e, %i*	pi, e,1*j* (from math import*)	touche directe	touche directe		pi, e, i
Définir une fonction Par exemple $f(x)=4x^3+2x+1$	function y=f(x) y=4*x^3+2*x+1 endfunction	def f(x): return 4*x**3+2*x+1	Touche Y= Y1=4*X^3+2*X+1	Menu GRAPH Y1= 4xX^3+2xX+1		f(x) :=4*x^3+2*x+1 ;
Saisir une fonction Exemple de réponse	rep=input(" f(x) vaut :"+"string") deff ('y=f(x)', "y="+rep) 4*x^3+2*x+1		Prompt Y_1 "4*X^3+2*X+1			input(f) ou saisir(f) x -> 4*x^3+2*x+1
puis, obtenir une image *f*(2) , *f*(a)	f(2) , f(a)	f(2) , f(a)	Y_1(2), Y_1(A), ATTENTION : Y_1 *se trouve dans* VARS /Y-VARS / Function	2→X : Y1 A→X : Y1 ATTENTION : *Pour obtenir* Y : VARS/GRPH/F1[Y]	Y1(2) Y1(A) Sur 35+ USB	f(2), f(a)

(*) Les fonctions Partie entière et Troncature différent sur les nombres négatifs : si x=-2,31 : partie entière : -3, troncature -2, partie décimale : 0,31, partie « fractionnaire » : 0,69