MEMENTO SCILAB

Fonctions mathématiques élémentaires	**Exemples**
abs=valeur absolue sqrt=racine carrée exp =exponentielle cos= cosinus sin= sinus	 abs(-2.3)= 2.3 sqrt(4)=2 exp(0)= 1
log = logarithme népérien ln = logarithme neperien si module lycée	log(10) = 2.3026
+, -, *(multiplication), /(division), ^(puissance) %pi, %e format(25) ;//pi	Opérations usuelles La constante π avec un certain nombre de décimale et idem pour la constante e Cela renvoie un certain nombre de décimale de π.
floor = partie entière par défaut *Facultatif :* *ceil = partie entière par excés* *fix = partie entière la plus proche de 0* *round = partie entière la plus proche*	floor(3.14) = 3 floor(3.8) = 3 ceil(3.14) = 4 ceil(3.8) = 4 fix(3.14) = 3 fix(-3.14) = -3 round(3.14) = 3 round(3.8) = 4 un nombre est **entier** si floor(x)==x un nombre est **pair** si floor(x/2)==x/2
rand() = un nombre "au hasard" $\in]0;1[$ rand(n,m) = matrice n×m de nombres tirés au hasard suivant la loi uniforme sur [0,1]	If rand () > 0.5 (proba = 50%) **tirage d'un entier** compris entre 1 et 6 : floor(6*rand()+1)
Affichage	**Exemples**
le **point-virgule** ; en fin de ligne permet de ne pas afficher le résultat de la ligne.	x = input(‘entrez un nombre : ‘) ;
disp : affiche la variable ou le vecteur ou la matrice.	disp(resultat) disp(M)
Les variables	**Exemples**
- affectation : compteur = 0 - **incrémentation** : x = x + 1 - affichage : disp(x) - la saisie par l'utilisateur : input	Cette instruction crée automatiquement la variable x dans scilab x = input(‘saisir un nombre : ’) ;
chaînes de caractères	**Exemples**
- concaténation : [...] - affichage : disp - saisie par l'utilisateur : input(‘...’, **‘s’**) - **nombre de caractères** : length - écrire une **apostrophe** : ’’	chaine = [chaine1, chaine2] disp(chaine) nom = input(‘entrez votre nom :’, ‘s’) ; longueur = length(nom) ; disp(‘vive l’’informatique’)

VECTEURS	EXEMPLES
- affectation : entre **crochets** - **longueur du vecteur** : length(V) -vecteurs lignes dont les coeffs sont régulièrement espacés : u=**linspace(**0,12,7) - affichage : on peut sélectionner une partie des composantes. -$ = end - **concaténation** : W = [U, V] - opérations : les opérations +, –, $\times\lambda$, $\div\lambda$, floor, ... agissent sur chaque composante. - pour **multiplier deux vecteurs** termes à termes : W = U .× V **ne pas oublier le point** devant le × ou le ^.	V = [] *// créer un vecteur vide* V = [10 3.14 -5 2] *// créer le vecteur [10 3,14 -5 2] (on oubli pas l'espace entre chaque nombre)* V = [3 : 2 : 11] *// créer le vecteur [3 5 7 9 11]* n = length(V) ; *// renvoie n = 5 avec l'exemple précédent.* *On découpe le segment [0,12] en 7 sous-segments // renvoie u=0.2.4.6.8.10.12.* V([2 : 4]) *// renvoie [5 7 9] avec l'exemple précédent.* V([2 : $]) *// renvoie [5 7 9 11] avec l'exemple précédent.* **pour en lever la i^{ème} composante du vecteur V** : V = [V(1 :i-1) V(i+1 :$)] W = 3*V + 1 ; W = floor(V/3) ; W = V .^ 2 ; *// élève chaque composante au carré*
MATRICES	EXEMPLES

- affectation : entre **crochets**
- retour ligne avec un **point virgule**
- matrice **nulle** : M = zeros(n,m) ; n lignes, m colonnes
- matrice **de 1** : M = ones(n,m) ;
-matrice **I_n=eye (n,n)**
- matrice de valeurs aléatoires : M = rand(n,m) ;
- **taille de la matrice** : size(M)
 	
- affichage : on peut sélectionner une partie des composantes.
 	
- coefficient d'une matrice
 	
- **concaténation** : entre crochets
W = [U, V] ou W = [U ; V]
les tailles doivent être compatibles.
- opérations : les opérations +, –, $\times\lambda$, $\div\lambda$, floor, ... agissent sur chaque composante.
- pour multiplier deux matrices termes à termes : W = M .× N
ne pas oublier le point devant le × ou le ^ sinon l'opération de multiplication est beaucoup plus compliquée.
-transposée de M : **M'**
-**rank**(A) = rang de A
-résolution d'un système X=inv(A)*B=A \ B (division à gauche) ;
	M = [] *// créer une matrice vide*
	M = [10 3.14 -5 2 ; 1 5 6 1] ;
	
	M = zeros(10,100) ;
	M = ones(5,3) ;
	//Matrice identité à n lignes,n colonnes
	M = floor(5*rand(3,10)) ; *// créer une matrice 3×10 de valeurs aléatoires entre 0 et 4.*
	[ligne, col] = size(M) ; *// ligne=3 et col=10*
	taille = size(M) ; *// taille(1)=3 et taille(2)=10*
	M([2 : 3], [5 :$])
	M(3, :) *// affiche la 3ème ligne en entier*
	M(:,2) *// affiche la 2ème colonne en entier*
	M(1,2) // renvoie le coefficient situé sur la 1ere ligne et la deuxième colonne
	
	M = [M1, M2] *// il faut que M1 et M2 aient le même nombre de lignes.*
	N = [N1 ; N2] *// il faut que N1 et N2 aient le même nombre de colonnes.*
	M = floor(V/3) ;
	
	W = M .^ 2 ; *// élève chaque composante de M au carré*
	
	Résolution de AX=B

IF-THEN-ELSE-END **IF-THEN-ELSEIF-ELSE-END**	**Exemples**

if test **then**
instruction 1
else
instruction 2
end
l'instruction 1 n'est exécutée que si test est VRAI.
si test est FAUX c'est l'instruction 2 qui est exécutée.
 	
if test **then**
instruction 1
elseif
test2 **then**
instruction 2
.....
else instruction N+1
end
 	
respectez l'indentation !
	if x > 0 then
	disp('x est positif')
	else
	disp('x est négatif ou nul')
	end
	
	pas de test après le ELSE !
	
	cas général avec différentes alternatives multiples
	if x > 0 then
	disp('x est positif')
	elseif x< 0 then
	disp('x est négatif ')
	else
	disp(' x est nul')
	end

opérateurs logiques :
- **et : &**
- **ou : |**
- **non : ~**
- **égalité : ==**
- **différent :<>**
- > >= < <=
	test pour 0 < x < 5 :
	if (0 < x) & (x < 5)
	test d'égalité pour deux scalaires :
	if (x == y)

test de scalaires : - test VRAI → T - test FAUX → F	pour (2 == 2), scilab renvoie T (true) pour (3 == 2), scilab renvoie F (false)
test de chaîne de caractères : il faut que les **deux chaînes aient la même longueur.**	if (reponse == 'oui') if (prenom == 'oscar') ne fonctionne pas si prenom n'a pas 5 lettres.
test matriciel : - un test sur une matrice renvoie une matrice de F et/ou de T. - la réponse est alors VRAIE si tous les éléments de la matrice sont des T. **il faut se méfier des tests matriciels !**	([1,2] == [1,3]) → [T F] ([1 2] ~= [1 3]) → [F T] donc considéré comme FAUX car [F T] ne contient pas que des T. Il faut écrire le test différemment car la réponse devrait être VRAIE ([1 2] est bien différent de [1 3]).

FOR-END	EXEMPLES
boucle répétée un nombre prédéterminé de fois. **For variable=debut :pas fin do** **Instructions** **end** **respectez l'indentation !** impossible de modifier i en cours de boucle.	Calcul de u_5 avec $u_{n+1} = u_n \times 5+2$ et $u_0 = 1$. u = 1 ; **for** i = 1 : 5 u = u*5 + 2 ; **end** disp(u)
un pas $\neq 1$ est possible	for i = 0 : 2 : 10 → nombres pairs de 0 à 10. for i = 5 :-1 :1 → i décroît de 5 à 1.

WHILE-END	EXEMPLES

c'est une **boucle conditionnelle** : elle se répète **tant que** le test est VRAI.
while test
instruction
end
respectez l'indentation !
s'il faut un compteur, pensez à l'initialiser et à l'incrémenter.
break permet de quitter une boucle for ou while
halt ou **pause** permet de faire une pause pendant l'execution d'un script.
	r=0 ;p=0.3 ;n=0 ;
	while (r= =0)
	u=rand() ;
	n=n+1 ;
	if (u<p) then
	r=1 ;
	end
	end
	disp(n,'n :') ;
	(simulation d'une variable aléatoire suivant la loi géométrique de parmètre p=0,3, r = résultat du lancer, u suit la loi uniforme sur [0,1], n =rang du 1er succés, l'événement [u<p] se produit avec une probabilité p.

FONCTIONS	EXEMPLES
- enregistrez-la dans votre dossier de travail (nom_fonction.sce). - la première ligne définit le nom de la fonction, les **variables d'entrée et de sortie**. - son nom ne contient **ni espace ni accent**. - **variables locales** (perdues après l'exécution de la fonction) **toutes les variables de sortie doivent recevoir une valeur avant la fin de la fonction.** pour ajouter des **commentaires** utilisez le symbole // : la suite de la ligne apparaît en vert et n'est pas pris en compte par Scilab. N'hésitez pas à en mettre beaucoup.	function reponse = test(a,b,c) if a ==b | a == c | b == c reponse = 1 ; else reponse = 0 ; end On peut la lancer avec : rep = test(1,2,5) ; *// la fonction renvoie la valeur numérique de* reponse *dans* rep. ou rep = test(x,y,z) ; *// la fonction reçoit la valeur numérique de x, y ,z dans a,b,c.* if x > a & x < b *// ce test vérifie si x est dans le bon intervalle* rep = appartient(x,a,b) *// rep=1 si x∈[a,b]*

GRAPHISME : PLOT/PLOT2D	EXEMPLES
-x=[-1 :0.1 :5] ;-y =f(x) (on peut aussi utiliser **linspace**)	x matrice ligne dont les coefficients varient de -1 à 5 par pas de 0,1 ; y matrice ligne de même taille que x.
plot2d(x,y) trace les y_i en fonction des x_i.	*xtitle permet de rajouter un titre*
xtitle(« Graphe de f, avec f(x)=xexp(-x) ») **legend** (« .... »)	*legend rajoute une légende si pls courbes*
penser à utiliser les opérations pointées **plot2d(x,y,s)** : la chaîne s = 'point, ligne, couleur' fixe les options du tracé	*: le plus simple aller dans le menu édition : propriétés de la figure, propriétés des axes.*
-clf ()	*Style =-1 permet d'avoir des croix ;* *Style = 3 permet d'avoir la courbe en vert* *Efface toutes les figures précédentes*
-Tracés de plusieurs courbes sur un même graphique :	x=[1 :0.01 :4] ;
l'argument rect=[1,0,4,3]) permet de définir la fenêtre d'affichage (x varie entre 1 et 4, alors que y varie entre 0 et 3).	y1=(x.^2)./(2*x-1) ; y2=1/2*x+1/4 ; plot2d(x,y1) ;
-Tracés de courbes dans plusieurs sous-fenêtres :	plot2d(x,y2,rect=[1,0,4,3]) ;
commande **subplot (m,n,p)** où m,n,p sont trois entiers tels que p≤ mn, divise la fenêtre graphique courante en mn sous-fenêtres de même taille (m lignes et n colonnes) et la prochaine figure sera tracée dans la p-ième sous-fenêtre.	x=[1 :0.01 :4] ;
Plot : plus facile que plot2d : plot(x,y), on aura défini $x = [0 : 0.1 : 4]; y = x.^2$ au préalable ou $y = [0 : 0.1 : 16]$.	y1=(x.^2)./(2*x-1) ;
On peut tracer plusieurs courbes avec la syntaxe suivante : plot(x1,y1,style1,x2,y2,style2, etc ...)	y2=1/2*x+1/4 ;
Tracé de suites récurrentes $u_n = nln(n)$ clf() function [y]=f(x) y=x.*log(x) endfunction	subplot(1,2,1) ;plot2d(x,y1,3) ; subplot(1,2,2) ;plot2d(x,y2,5) ;
n=[1 :1 :20] u=f(n) plot(n,u,"b+")	Tracé de suites de la forme $u_{n+1} = f(u_n)$ n=input("valeur de n ?") u=0 tab=[u] for i=1 :n u=(u-4)/(u-3) tab=[tab,u] end plot([0 :n],tab,"b+") Rmq. Notons que la fonction plot2d2 permet de faire un affichage en escalier assez parlant pour les suites.

Probabilités et statistiques	Signification et exemple
Simulation de toutes les lois usuelles Commande **grand** y_1=grand(m,n, « bin »,N,p) y_2=grand(m,n, « poi »,lambda) y_3=grand(m,n, « nor »,mu,sigma) y_4=grand(m,n, « unf »,a,b) y_5=grand(m,n, « uin »,1,n) y_6=grand(m,n, « exp »,1/lambda) y_7=grand(m,n, « geom », p) y_8=grand(m,n, « gam »,b,nu)	Génère une matrice de taille m x n constituée de nombres aléatoires distribués selon la loi spécifiée : Loi binomiale Loi de poisson Loi normale Loi uniforme à densité Loi uniforme discrète Loi exponentielle Loi géométrique Loi gamma Attention : pour la loi normale c'est l'écart type qui est donné, pour la loi exponentielle c'est l'espérance.
mean(x) : calcul la moyenne V_emp=**variance (x)** ecart_type=**st_deviation(x)** mediane=**median(x)** maximum=**max(x)**	Simulation d'une variable x suivant une loi exponentielle de paramètre 2 : lambda =2 x=grand(1,100, « exp »,1/lambda) m=mean(x) variance empirique ecart type mediane maximum
Histogrammes : commande **histplot(n,x)** où x est généré par grand, n classes de même largeur. **histplot(b,x)** Penser à utiliser **plot2d3** : pour avoir des barres. Il vaut mieux utiliser la commande **bar** qui permet de bien utiliser la largeur des colonnes.	*Trace l'histogramme de x en utilisant n classes de même largeur, l'effectif de chaque classe est normalisé par l'effectif total.* x=grand(1,100, « nor »,1,2) ; subplot(121) ; histplot(10,x) ; *Trace l'histogramme de x en utilisant les classes dont les bornes sont définies par le vecteur b ([b_1,b_2],[b_2,b_3])...)* x=grand(1,100, « nor »,1,2) ; subplot(122) ; b=[-9,-4,0,1,2,5,10] ; histplot(b,x) ; **Generer la loi binomiale :** bn= grand(1000,1,'bin',10,0.2) ; bm=tabul(bn) *;//compte sur les 1000 lancers le nombre de 0,1,2,3..., permet de récupérer le nombre d'occurrence de chaque valeur.* bar(bm(:,2),0.1) *// 2^e colonne de bm, épaisseur des barres*