

# Le calcul scientifique sous Python et Scilab

## 1. Les spécificités des langages pour le calcul scientifique

### 1.1. Python et Scilab

Le langage Python est un langage généraliste, disposant de bibliothèques puissantes en calcul numérique (scipy, numpy et matplotlib) permettant de l'utiliser dans le cadre du calcul scientifique. Plusieurs interfaces graphiques (Spyder, Eric, etc) permettent de travailler dans un environnement Python. Spyder est plus particulièrement conçu pour le calcul scientifique.


Scilab est d'abord un logiciel, proposant une interface graphique de travail et un langage de programmation. Semblable au logiciel Matlab, le logiciel Scilab est dédié au calcul scientifique pour la recherche et l'industrie. Il est moins généraliste que Python, mais dispose d'un outil de modélisation graphique (Xcos) permettant de manipuler des modèles numériques complexes, ce qui manque à Python.



### 1.2. Pourquoi un peu de C ?

Le C et ses variantes est, de très loin, le langage le plus répandu en informatique. Il est néanmoins un peu plus rigide et complexe à manipuler pour l'apprentissage de la programmation (il n'est pas interactif par exemple).

Pour la programmation des micro-contrôleurs (utilisés en TP de SI), il est incontournable. C'est pourquoi le formulaire ci-dessous propose, pour un petit nombre de fonctionnalités indispensables pour la commande des systèmes embarqués, les expressions équivalentes en C.















### 1.3. Utilisation du formulaire

Le formulaire tente de couvrir en une page l'essentiel des commandes utiles dans le cadre du programme de CPGE. Chaque commande est illustrée en une ligne, permettant de se remémorer rapidement la syntaxe des fonctions dans chaque langage. Seules les commandes présentant le symbole  sont raisonnablement à connaître pour les concours.

Le symbole  redirige vers une page proposant plus d'explications et d'exemples pour certaines commandes lorsque c'est utile. Le symbole  renvoie vers une documentation officielle de la commande.







## 2. Formulaire

### 2.1. Manipulation globale

Description	Python	Scilab	C
Importer un fichier	<code>import file</code>	 <code>exec("fichier.sce")</code> <code>exec fichier.sce</code>	
Lancer l'aide interactive	<code>help()</code>	 <code>help</code>	
Afficher l'aide associée à la commande cosinus	<code>help(cos)</code>	<code>help cos</code> <code>apropos cos</code>	
Afficher x sur la console interactive	 <code>print(x)</code>	 <code>disp(x)</code>	
Importer une bibliothèque	<code>import bibliotheque as nom</code> <code>from bibliotheque import *</code>	 Menu > Applications > Gestionnaire de module - Atoms	
Définir une fonction	 <code>def nomdelafunction(paramètres):</code> bloc d'instruction <code>return resultat</code>	<code>function resultat=nomdelafunction(paramètres)</code> bloc d'instruction <code>endfunction</code>	
Interruption d'un programme pour debug	<code>import pdb</code> <code>pdb.set_trace()</code>	   <code>pause</code>	 

### 2.2. Variables

Les variables sont dans les 3 langages, systématiquement associées à un type (et donc un espace mémoire alloué). Néanmoins, le typage est "dynamique" en Python et Scilab, c'est-à-dire que le type peut changer en fonction des besoins du calcul, tandis qu'il est statique en C (le type est choisi avant l'initialisation de la variable et ne peut plus être modifié par la suite, ce qui s'avère beaucoup plus simple et efficace pour la gestion de la mémoire).

Description	Python	Scilab	C
Obtenir le type d'une variable a	<code>type(a)</code>	  <code>type(a)</code> <code>typeof(a)</code> <code>inttype(a)</code>	 explicite dans le programme...
Changement de type	<code>int(a)</code> <code>float(a)</code> <code>str(a)</code>	  <code>int(a)</code> <code>int8(a)</code> <code>int16(a)</code> <code>float32(a)</code> <code>float64(a)</code>	

### Nombres

Nous passons sur les opérateurs classiques + - \* / qui sont identiques dans les 3 langages (attention toutefois à la division de deux entiers qui est réalisée dans les entiers en C et en python 2.7, et dans R dans Scilab et en Python 3.3...). Il est nécessaire d'importer la bibliothèque math.h en C.

Description	Python	Scilab	C
Puissance $a^b$	♥ <code>a**b</code>	<code>a^b</code>	<code>pow(a,b)</code>
Reste ou modulo	♥ <code>a%b</code>	<code>modulo(a,b)</code>	<code>a%b</code>
Division entière	♥ <code>a//b</code>	<code>floor(a/b)</code>	<code>a/b</code> avec <code>a</code> et <code>b</code> entiers
Test égalité	♥ <code>a == b</code>	<code>a == b</code>	<code>a == b</code>
Inférieur (ou égal)	♥ <code>a &lt; b ; a &lt;= b</code>	<code>a &lt; b ; a &lt;= b</code>	<code>a &lt; b ; a &lt;= b</code>
Supérieur (ou égal)	♥ <code>a &gt; b ; a &gt;= b</code>	<code>a &gt; b ; a &gt;= b</code>	<code>a&gt;b ; a&gt;=b</code>
Différent	♥ <code>a != b</code>	<code>a ~= b</code>	<code>a != b</code>
ET	♥ <code>a and b</code>	<code>a &amp; b</code>	<code>a &amp;&amp; b</code>
ET bitwise	<code>a &amp; b</code>	<code>bitand(a,b)</code>	<code>a &amp; b</code>
OU	♥ <code>a or b</code>	<code>a   b</code>	<code>a    b</code>
OU bitwise	<code>a   b</code>	<code>bitor(a,b)</code>	<code>a   b</code>
NON	♥ <code>not a</code>	<code>~a</code>	<code>!a</code>
NON bitwise	<code>~a</code>	<code>bitcmp(a,8)</code>	<code>~a</code>
OU exclusif	<code>a ^ b</code>	<code>~(a == b)</code>	<code>!(a == b)</code>
OU exclusif bitwise	<code>a ^ b</code>	<code>bitxor(a,b)</code>	<code>a ^ b</code>
décalage de 3 bits à gauche et à droite	♥ <code>a &lt;&lt; 3 ; a &gt;&gt; 3</code>	<code>a*2^3 ; floor(a/2^3)</code>	<code>a &lt;&lt; 3 ; a &gt;&gt; 3</code>
Affichage d'un nombre en binaire	♥ <code>bin(a)</code>	<code>dec2bin(a)</code>	
Affichage d'un nombre en hexadécimal	<code>hex(a)</code>	<code>dec2hex(a)</code>	

Les fonctions mathématiques standards en python peuvent provenir de différents paquets. L'utilisation de l'option `--pylab` (automatiquement chargée sous Spyder) conduit à utiliser les fonctions des modules Numpy, Scipy et Matplotlib par défaut.

Description	Python	Scilab	C
Racine carrée	♥ <code>sqrt(a)</code>	? <code>sqrt(a)</code>	<code>sqrt(a)</code>
cosinus	♥ <code>cos(a)</code>	? <code>cos(a)</code>	<code>cos(a)</code>
sinus	<code>sin(a)</code>	<code>sin(a)</code>	<code>sin(a)</code>
tangente	<code>tan(a)</code>	<code>tan(a)</code>	<code>tan(a)</code>
arcsin	<code>acos(a)</code>	? <code>acos(a)</code>	
arcsin	<code>asin(a)</code>	<code>asin(a)</code>	
arctangente sur $]-\pi/2, \pi/2[$	<code>atan(a)</code>	<code>atan(a)</code>	<code>atan(a)</code>
arctangente sur $]-\pi, \pi]$	<code>atan2(y,x)</code>	? <code>atan(y,x)</code>	<code>atan2(x,y)</code>
ch, sh, th	<code>cosh(x)</code> <code>sinh(x)</code> <code>tanh(x)</code>	? <code></code>	<code>cosh(x)</code> <code>sinh(x)</code> <code>tanh(x)</code>
Logarithme Néperien (ln)	<code>log(a)</code>	<code>log(a)</code>	<code>log(a)</code>
Logarithme en base 10 (log)	<code>log10(a)</code>	<code>log10(a)</code>	<code>log10(a)</code>
Logarithme en base $n$ quelconque	<code>log(a,n)</code>	<code>log(a)/log(n)</code>	

Exponentielle $e^a$	<code>exp(a)</code>	<code>exp(a)</code>	<code>exp(a)</code>
Arrondi entier	<code>round(a)</code> <code>fix(a)</code>	<code>round(a)</code> <code>fix(a)</code>	<code>round(a)</code>
Arrondi à l'entier supérieur	<code>ceil(a)</code>	<code>ceil(a)</code>	<code>ceil(a)</code>
Arrondi à l'entier inférieur	<code>floor(a)</code>	<code>floor(a)</code>	<code>floor(a)</code>
Générer un nombre ou une liste aléatoire	<code>random.random()</code> <code>numpy.random.sample(10)</code>	<code>rand()</code> <code>rand(1,10)</code>	<code>int nb_rand;</code> <code>srand(time(NULL));</code> <code>nb_rand=rand();</code> <code>&lt;module&gt;</code> Il peut être nécessaire d'importer time. <code>&lt;/module&gt;</code>
Constantes $\pi$ $e$	<code>pi</code> <code>e</code>	<code>%pi</code> <code>%e</code>	

Le calcul en complexes est possible dans les langages Python et Scilab.

Description	Python	Scilab	C
imaginaire $i$	<code>1j</code>	<code>%i</code>	
Nombre complexes $3+4i$	<code>3+4j</code>	<code>3+4*%i</code>	
Partie réelle de $a$	<code>a.real</code>	<code>real(a)</code>	
Partie imaginaire de $a$	<code>a.imag</code>	<code>imag(a)</code>	
Module	<code>abs(a)</code>	<code>abs(a)</code>	
Argument	<code>arctan2(a.imag,a.real)</code>	<code>atan(imag(a),real(a))</code>	
Conjugué de $a$	<code>a.conj()</code>	<code>conj(a)</code>	

## Tableaux

Le calcul numérique s'appuie intensivement sur des tableaux de valeurs, soit pour représenter des signaux, soit pour représenter des vecteurs et des matrices. Les tableaux peuvent avoir une ou plusieurs dimensions

La syntaxe de Scilab est fortement orienté vers la manipulation de tableau. Elle est en ce sens très pratique et lisible.

Python propose quant à lui trois types possibles pour manipuler des données sous forme de tableaux : les listes (à une ou plusieurs dimensions), le type "array", et le type "matrix". Des conversions sont possibles entre ces types. Il est conseillé dans le cadre du calcul numérique (pour des raisons pratiques et d'efficacité des calculs) d'utiliser le type "array". C'est ce qui est considéré par la suite.

Attention, en Python et en C, les indices des tableaux commencent à 0 tandis qu'en Scilab, ils commencent à 1.

Description	Python	Scilab	C
Vecteur ligne	♥ <code>v=array([1, 2, 3])</code>	<code>v=[1, 2, 3]</code> ou <code>[1 2 3]</code>	<code>v=[1, 2, 3] ??</code>
Vecteur colonne	♥ <code>v=array([[1],[2],[3]])</code> <code>v=array([1, 2, 3])[:,newaxis]</code> <code>v=array([1, 2, 3]).reshape(-1,1)</code>	? <code>v=[1; 2; 3]</code>	

## Calcul Scientifique sous Python et Scilab

Tableau à 2 dimensions	♥ <code>M=array([[1,2,3],[4,5,6]])</code>	🔍 <code>M=[1, 2, 3; 4, 5, 6]</code>	
Accéder à un élément	♥ <code>v[0], M[0,1]</code>	<code>v(1), M(1,2)</code>	<code>v(3) ????</code>
Accéder au dernier élément, et l'avant dernier	♥ <code>v[-1], v[-2]</code>	<code>v(\$), v(\$-1)</code>	
Dimensions d'un tableau	♥ <code>M.shape</code>	<code>size(M)</code>	
Extraire la 2ème ligne ou 2ème colonne	♥ <code>M[1,:] ou M[:,1]</code>	<code>M(2,:) ou M(:,2)</code>	
Extraire une portion de tableau (2 premières colonnes)	♥ <code>M[:,0:2]</code>	<code>M(:,1:2)</code>	
Extraire des éléments d'un tableau par leurs indices	♥ <code>M[0,(2,1)]</code>	<code>M([1],[3,2])</code>	
Séquence équirépartie d'entiers	♥ <code>range(1,11)</code>	<code>1:10</code>	
Séquence équirépartie quelconque	♥ <code>arange(0,10.1,0.1)</code>	<code>0:0.1:10</code>	
Tableau de zéros	♥ <code>zeros((2,3),float)</code>	<code>zeros(2,3)</code>	
Tableau de uns	♥ <code>ones((2,3),float)</code>	<code>ones(2,3)</code>	
Copier un tableau dans une autre variable	♥ <code>w=v.copy()</code>	<code>w=v</code>	
Multiplication élément par élément	♥ <code>v*w</code>	<code>v.*w</code>	
Maximum et minimum d'un tableau	<code>v.max(0), v.min(0)</code>	<code>max(v), min(v)</code>	
Indice <i>i</i> du maximum	<code>v.argmax(0)</code>	<code>[m,i] = max(v)</code>	

## Matrices

Description	Python	Scilab	C
Produit matriciel	♥ <code>dot(v,w)</code>	<code>v*w</code>	
Transposée	♥ <code>M.transpose()</code>	<code>M'</code>	
Résolution de système matriciel $M.X=Y$	♥ <code>linalg.solve(M,Y)</code>	<code>X=M\Y</code>	🔍
Produit scalaire de deux vecteurs	<code>vdot(v,w)</code>	<code>v'*w</code>	
Produit vectoriel	<code>cross(v,w)</code>	<code>cross(v,w)</code>	🔍
Déterminant	<code>linalg.det(M)</code>	<code>det(M)</code>	
Inverse	<code>linalg.inv(M)</code>	<code>inv(M)</code>	
Valeurs propres	<code>linalg.eig(M)[0]</code>	<code>spec(M)</code>	
Vecteurs propres	<code>linalg.eig(M)[1]</code>	<code>[v,l] = spec(M)</code>	🔍
Rang	<code>rank(M)</code>	<code>rank(M)</code>	

## Listes

Description	Python	Scilab	C
Définir une liste	♥ <code>liste1 = [M,v]</code>	<code>liste1 = list(M,v)</code>	?
Sélectionner par position	♥ <code>liste1[i]</code>	<code>liste1(i)</code>	
Supprimer un élément	<code>del liste1[i]</code>	<code>L(i)=null()</code>	

## Dictionnaires et structures

Ce sont des listes un peu plus complexes en ce que chacun des éléments qui le compose est au moins composé de 2 membres, l'un étant la clé, l'autre la valeur.

En python, le dictionnaire est embrassé par des accolades { }.

Scilab propose aussi des structures indicées (`cell()`) qui ne seront pas développées ci-dessous (car plus ou moins équivalentes à une liste).

Description	Python	Scilab	C
Définir un dictionnaire	<code>D1 = {'Scal': 5, 'Vect': array([1,2,3])}</code>	?	?
Obtenir ou changer un élément	<code>D1['Scal'] = 1 + D1['Vect'][2]</code>	<code>S1=struct('Scal', 5, 'Vect', [1,2,3])</code>	?
Supprimer un élément	<code>del D1['Scal']</code> <code>clear()</code> <code>pop(cle)</code> <code>popitem()</code>	<code>S1.Scal=null()</code>	

## Chaînes de caractères

En C il faut utiliser la bibliothèque `string.h`.

Description	Python	Scilab	C
Définir une chaîne de caractères	♥ <code>mot="Python et Scilab"</code>	<code>mot="Python et Scilab"</code>	<code>mot="Python et Scilab"</code>
Longueur d'une chaîne	♥ <code>len(mot)</code>	<code>length(mot)</code>	<code>strlen(mot)</code>
Extraire des caractères	♥ <code>mot[2:7]</code>	<code>part(mot,[1,2,11:16])</code>	
Concaténation	♥ <code>mot="python" + "/" + "Scilab"</code>	<code>mot="python" + "/" + "Scilab"</code>	<code>strcat(mot1,mot2)</code>
Remplacer une portion de chaîne	<code>mot.replace("Scilab","C")</code>	<code>strsubst(mot,"Scilab","C")</code>	
Découper une chaîne de caractères	<code>mot.split(" ")</code>	<code>strsplit(mot," ")</code>	
Supprimer des caractères de retour à la ligne	<code>mot.rstrip("\n\r")</code>		

## 2.3. Structures de contrôle

### Boucles

Description	Python	Scilab	C
Boucle FOR	♥ <code>for i in range(10):   print(i)</code>	<code>for i=1:10   disp(i); end</code>	? <code>for (i=1;i&lt;=10;i++){   printf("%d",i); }</code>
Boucle WHILE	♥ <code>i=0 while (i&lt;10):   i+=1   print(i)</code>	<code>i=0 while (i&lt;10)   i=i+1   disp(i) end</code>	? <code>int i =10; while (i&lt;10){   printf("%d",i);   i++; }</code>
Interruption d'une boucle	<code>break</code>	<code>break</code>	? <code>break</code>

### Conditions

Description	Python	Scilab	C
Condition IF	♥ <code>if (i&gt;3):   print(i) else   print("hello")</code>	<code>if (i&gt;3) then   disp(i) else   disp("hello") end</code>	? <code>if (i&gt;3){   printf("%d",i); } else{   printf("hello"); }</code>
Condition CASE		<code>select i,   case 1 then     disp("Egal à 1");   case 2 then     disp("Egal à 2");   else     disp("Aucun des deux"); end</code>	? <code>switch(i) {   case 2: {     b=b+1; }   break;   case 3: {     b=b+5; }   break;   default: {     b=0; } }</code>

### Fonctions




Description	Python	Scilab	C
Définir une fonction	♥ <code>def nomdelafunction(paramètres):   bloc d'instructions   return resultat</code>	<code>function resultat=nomdelafunction(paramètres)   bloc d'instructions endfunction</code>	? <code>function resultat=nomdelafunction(paramètres)   bloc d'instructions endfunction</code>

## 2.4. Courbes

### Courbes 2D

En Python, il est parfois nécessaire d'exécuter la commande `show()` pour afficher le graphique après l'appel à la commande `plot()`.











Description	Python	Scilab	C
Tracé d'une courbe y ou (x,y)	♥ <code>plot(y) ou plot(x,y)</code>	<code>plot(y) ou plot(x,y)</code>	? <code>plot(y) ou plot(x,y)</code>
Tracé de points (o) rouges (r) reliés par des lignes (-)	<code>plot(x,y,"-or")</code>	<code>plot(x,y,"-or")</code>	? <code>plot(x,y,"-or")</code>
Effacer le graphique	♥ <code>clf()</code>	<code>clf</code>	? <code>clf</code>

Ouvrir une nouvelle figure	<code>figure(3)</code>	<code>figure(3)</code>		
Échelle logarithmique en X	<code>semilogx(x,y)</code>	<code>plot2d("ln",x,y)</code>		
Échelle logarithmique en X et Y	<code>loglog(x,y)</code>	<code>plot2d("ll",x,y)</code>		






## Courbes 3D

## 2.5. Fichiers

### Fichiers textes

Description	Python	Scilab	C
Ouvrir un fichier texte en lecture/écriture	 <code>fic=open("fichier.txt","r")</code> <code>fic=open("fichier.txt","w")</code>	<code>fic=mopen("fichier.txt","r")</code> <code>fic=mopen("fichier.txt","w")</code>	
Fermer un fichier	 <code>fic.close()</code>	<code>close(fic)</code>	
Lire une ou plusieurs lignes	 <code>ligne=fic.readline()</code> <code>lignes=fic.readlines()</code>	<code>ligne=mgetl(f,1)</code> <code>lignes=mgetl(f)</code>	
Lire un tableau formaté	<code>a,b=loadtxt("Fichier.txt", usecols = (0,2), dtype={ 'names': ('numero', 'consigne'), 'formats': ('i2', 'f4')}, delimiter=',', unpack=True)</code>	<code>Tableau=mfscanf(-1,fic,"%d,%f,%f")</code>	
Écrire une ligne	 <code>fic.write("il fait {:f} degrees.\n"\n.format(10))</code>	<code>mfprintf(fic,"il fait %f degrees.\n",10)</code>	
Écrire un tableau formaté	<code>for i in range(len(x)) fic.write("{:d},{:f},{:f}\n"\n.format(i,x[i],y[i]))</code>	<code>mfprintf(fic,"%d,%f,%f",1:100,x,y)</code>	

### Fichiers binaires

Description	Python	Scilab	C
Sauver des variables dans un fichier binaire	<code>import pickle</code> <code>fic=open("fichier.pick","wb")</code> <code>pickle.dump(a,fic)</code> <code>pickle.dump(b,fic)</code> <code>fic.close()</code>	<code>save("fichier.dat",a,b)</code>	
Recharger des variables du fichier binaire	<code>import pickle</code> <code>fic=open("fichier.pick","rb")</code> <code>pickle.load(a,fic)</code> <code>pickle.load(b,fic)</code> <code>fic.close()</code>	<code>load("fichier.dat","a","b")</code>	
Ouvrir un fichier binaire en lecture/écriture	<code>fic=open("fichier.txt","rb")</code> <code>fic=open("fichier.txt","wb")</code>	<code>fic=mopen("fichier.txt","rb")</code> <code>fic=mopen("fichier.txt","wb")</code>	
Lire 3 octets dans un fichier binaire	<code>octets=fic.read(3)</code>	<code>octets=mget(3,"c",fic)</code>	
Écrire des octets dans un fichier binaire	<code>fic.write("PCSI")</code> <code>fic.write(int8(83))</code> <code>fic.write(float32(2.3))</code>	<code>mput(ascii("PCSI"),"c",fic)</code> <code>mput(83,"i",fic)</code> <code>mput(2.3,"f",fic)</code>	



## Images

La lecture et l'écriture d'image sous python est relativement simple par la bibliothèque `scipy.misc`, chargé automatiquement par l'option `--pylab` ou par `spyder`. Les formats supportés sont `jpg`, `bmp`, `????`. Des bibliothèques plus élaborées existent (`PIL` par exemple) mais ne sont pas abordées ici.

La lecture et l'écriture d'image sous Scilab nécessite l'installation par Atoms du module `SIVP` (image and vidéo processing). La plupart des formats sont supportés.

Les images sont ensuite manipulées dans le programme sous forme de tableau.

Description	Python	Scilab	C
Ouvrir une image	<code>im=imread("image.jpg")</code>	<code>im=imread("image.jpg")</code>	
Enregistrer une image	<code>imsave("image.jpg",im)</code>	<code>imwrite(im,"image.jpg")</code>	
Afficher une image	<code>imshow(im)</code>	<code>imshow(im)</code>	
Taille de l'image	<code>im.shape</code>	<code>size(im)</code>	

## 2.6. Calcul numérique

Description	Python	Scilab	C
Intégration numérique d'une fonction $f(x)$ de $x_0$ à $x_1$	??	<code>integrate("sin(x)", "x", x0, x1)</code>	
Intégration par la méthode des trapèzes	<code>trapz(y[, x, dx, axis])</code>	<code>inttrap(x,y)</code>	
Dérivation numérique d'une fonction $f(x)$	??	<code>derivative(f,x)</code>	
Différences finies	<code>diff(a[, n, axis])</code>	<code>numdiff(y,x)</code>	
Solution d'une équation non linéaire $f(x)=0$	??	<code>fsolve(x0,f)</code>	
Minimisation d'une fonction $f(x)$	??	<code>optim(f,x0)</code>	
Intégration numérique d'une équation différentielle ordinaire (ODE) $\frac{dx}{dt}=f(x,t)$	??	<code>ode(x0,t0,t,f)</code>	
Intégration numérique d'une équation différentielle algébrique (DAE) $f(x, \frac{dx}{dt}, t)=0$	??	<code>dae(x0,t0,t,f)</code>	

## Sommaire

1. Les spécificités des langages pour le calcul scientifique
  - 1.1. Python et Scilab
  - 1.2. Pourquoi un peu de C ?
  - 1.3. Utilisation du formulaire
2. Formulaire
  - 2.1. Manipulation globale
  - 2.2. Variables
    - Nombres*
    - Tableaux*
    - Matrices*
    - Listes*
    - Dictionnaires et structures*
    - Chaînes de caractères*
  - 2.3. Structures de contrôle
    - Boucles*
    - Conditions*
    - Fonctions*
  - 2.4. Courbes
    - Courbes 2D*
    - Courbes 3D*
  - 2.5. Fichiers
    - Fichiers textes*
    - Fichiers binaires*
    - Images*
  - 2.6. Calcul numérique