

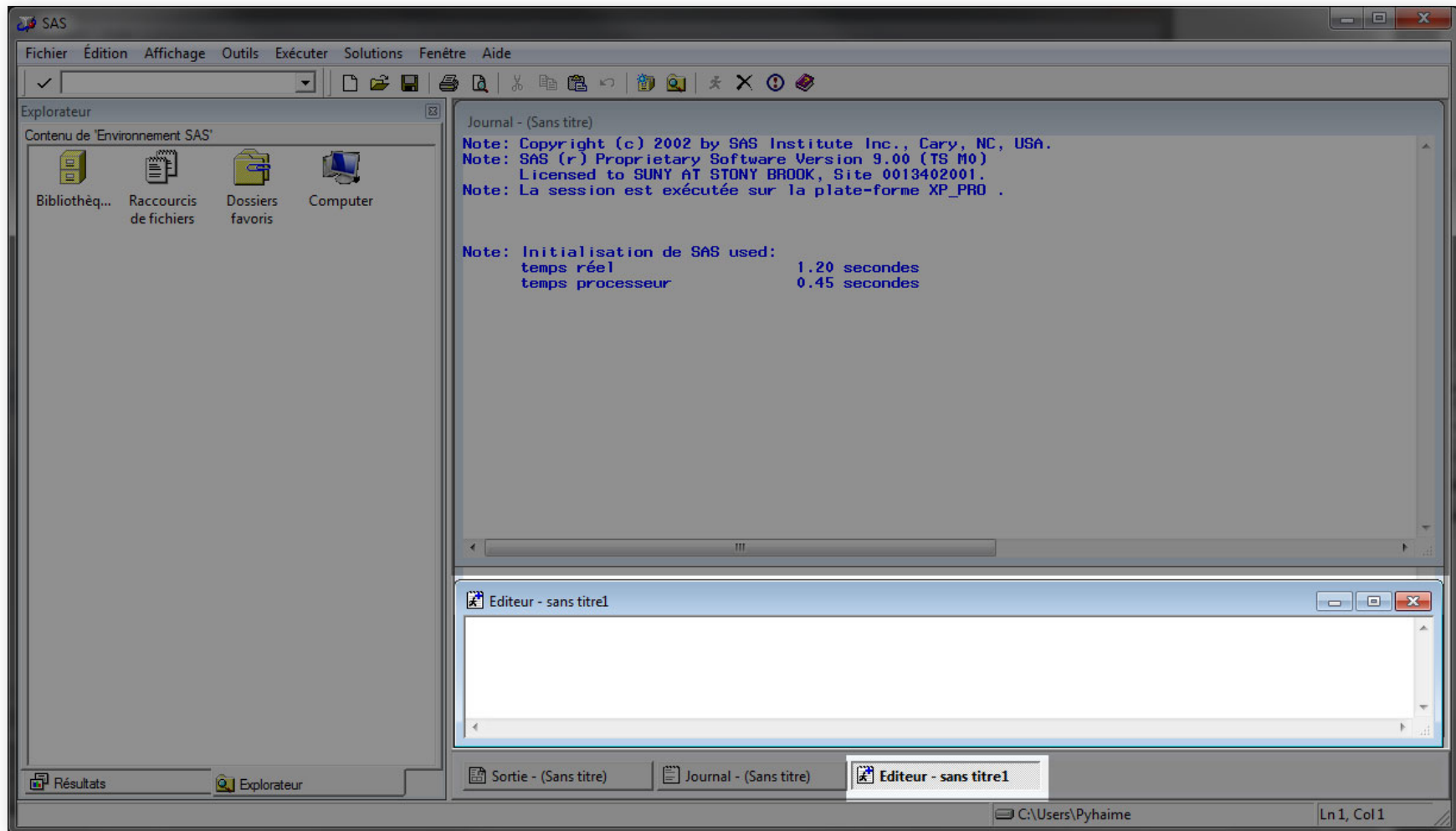
# **STATISTIQUES ET ÉTUDES ÉCONOMIQUES SOUS SAS**

## **INTRODUCTION**

# L'INTERFACE

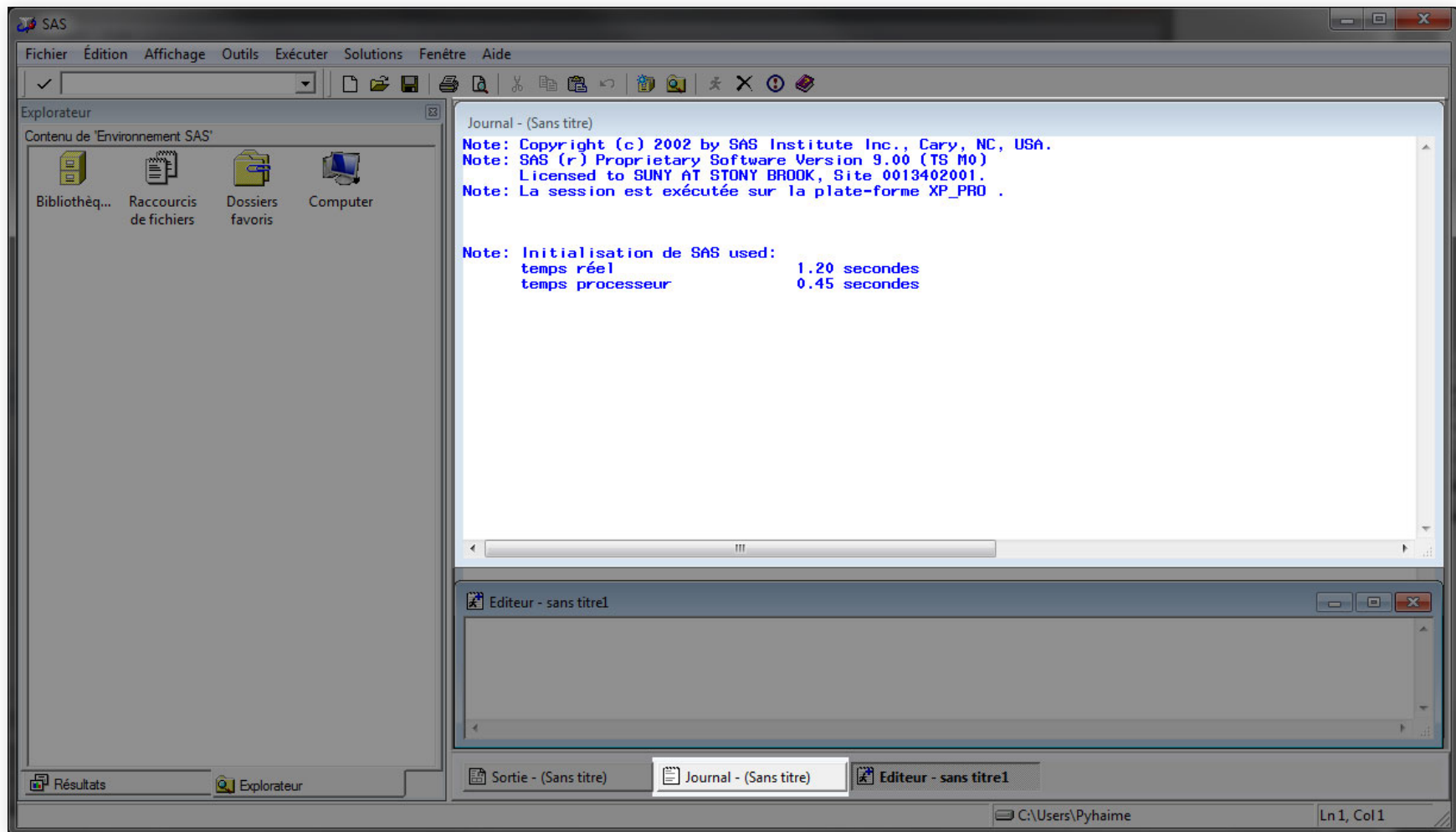
# L'ÉDITEUR

La fenêtre Editeur est la fenêtre dans laquelle les lignes de commandes sont insérées



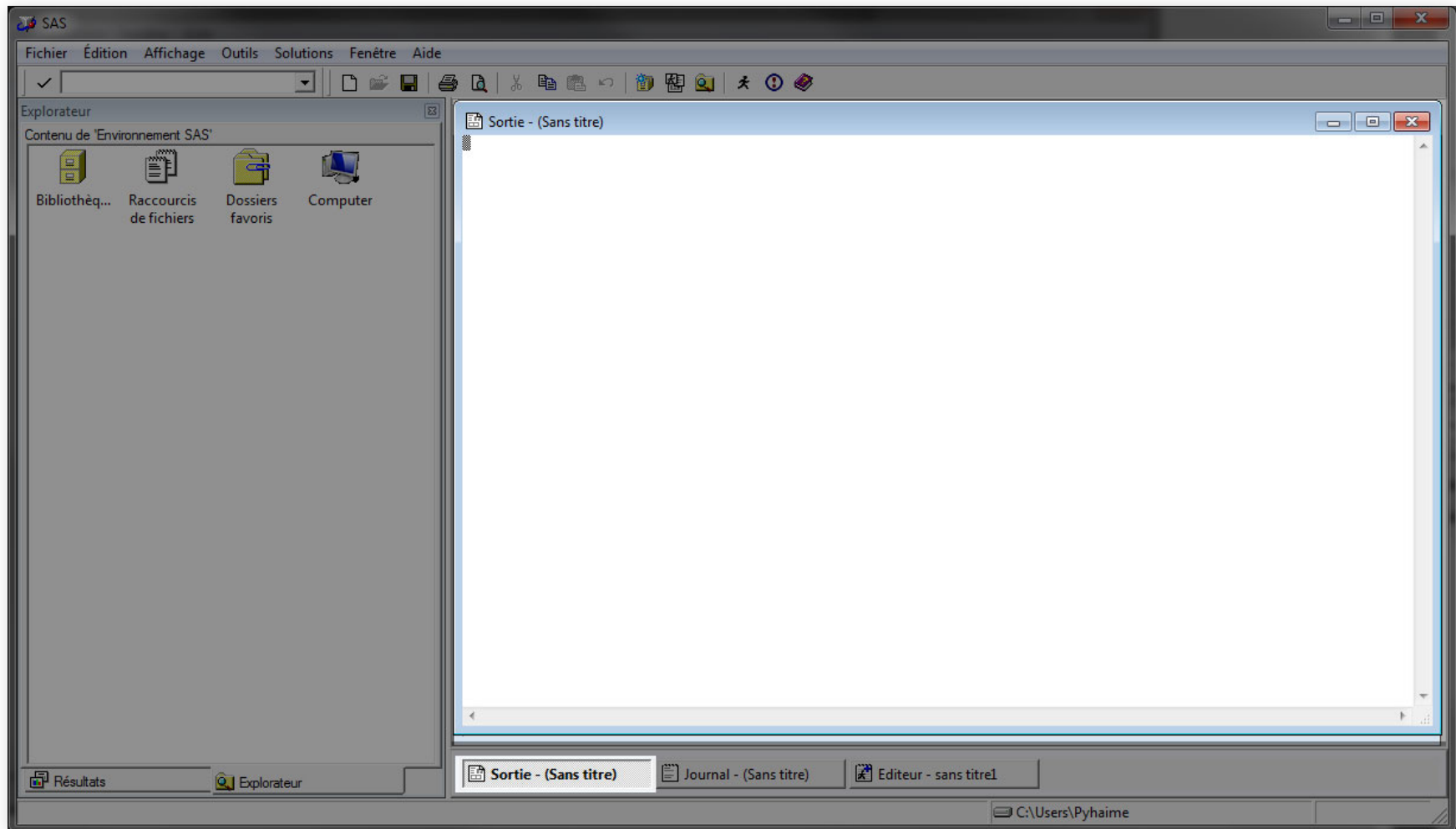
# LE JOURNAL

Le journal (ou Log) est la synthèse de l'exécution des instructions :  
Messages d'erreurs, avertissements et notes



# LA SORTIE

La fenêtre Sortie (ou OUTPUT) contient les résultats issus d'étapes PROC

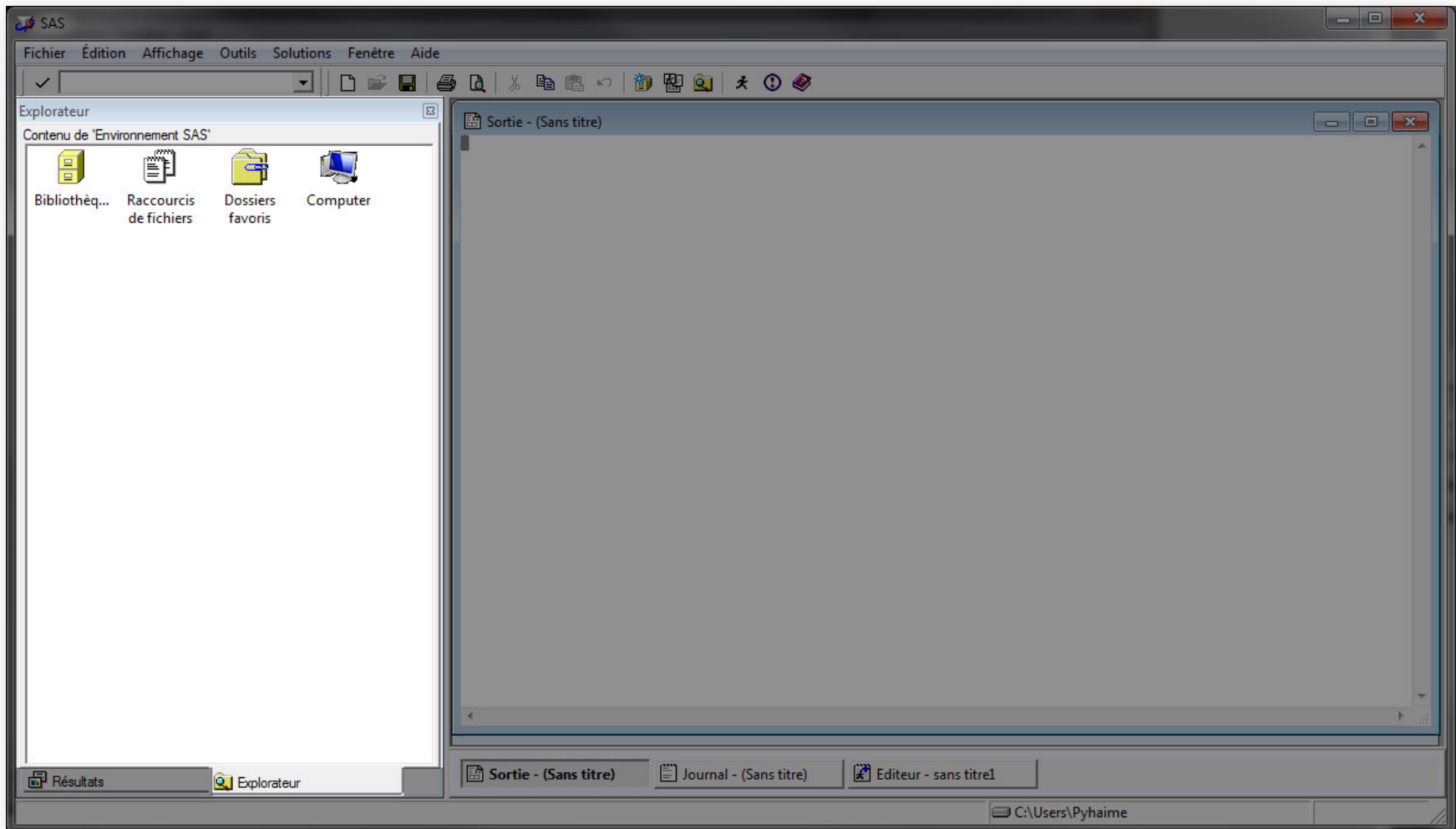


# EXPLORATEUR / RÉSULTATS

Dans la fenêtre de gauche, deux onglets sont présents

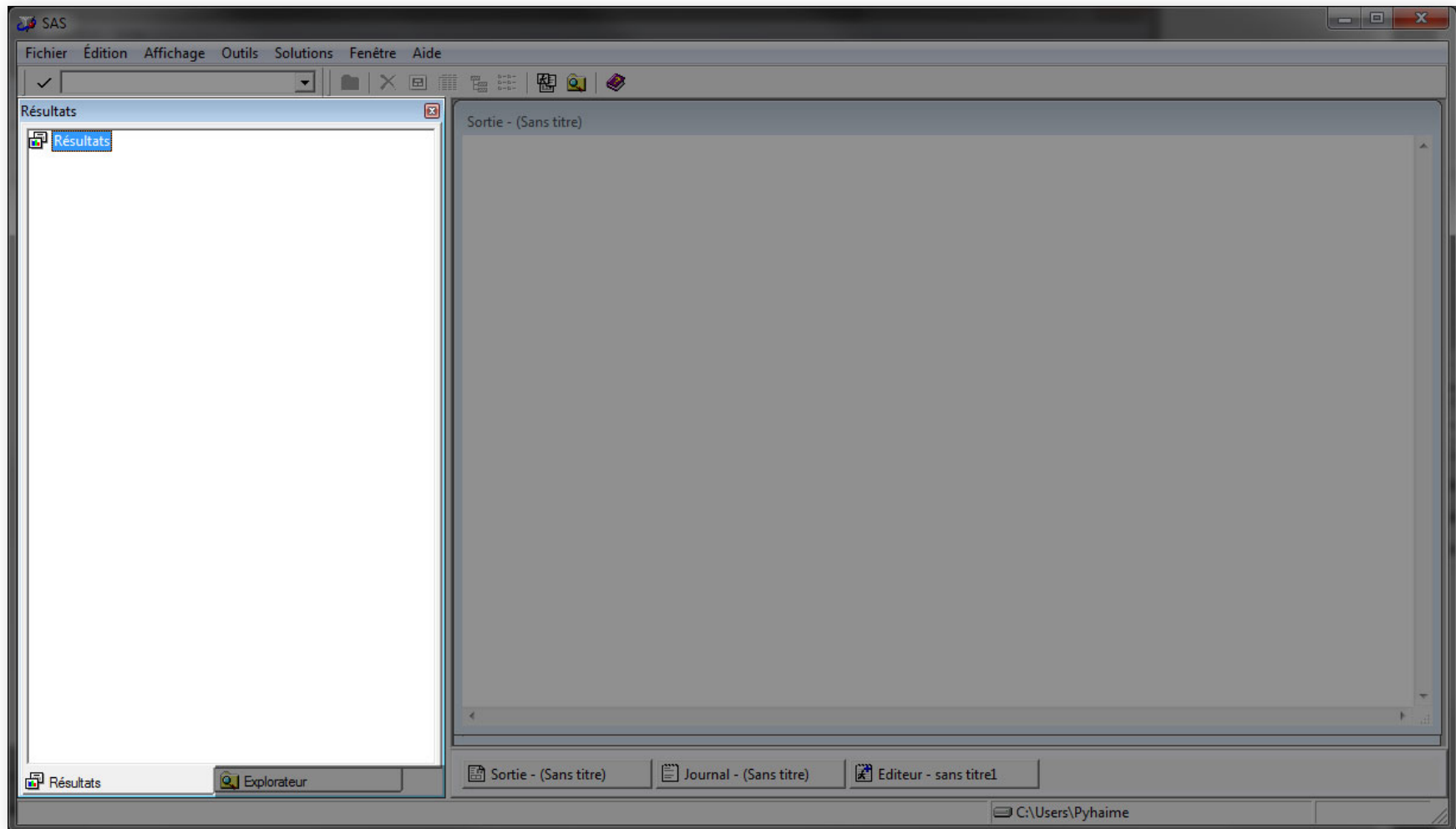
# EXPLORATEUR

L'onglet Explorateur donne notamment accès aux bibliothèques et à leur contenu



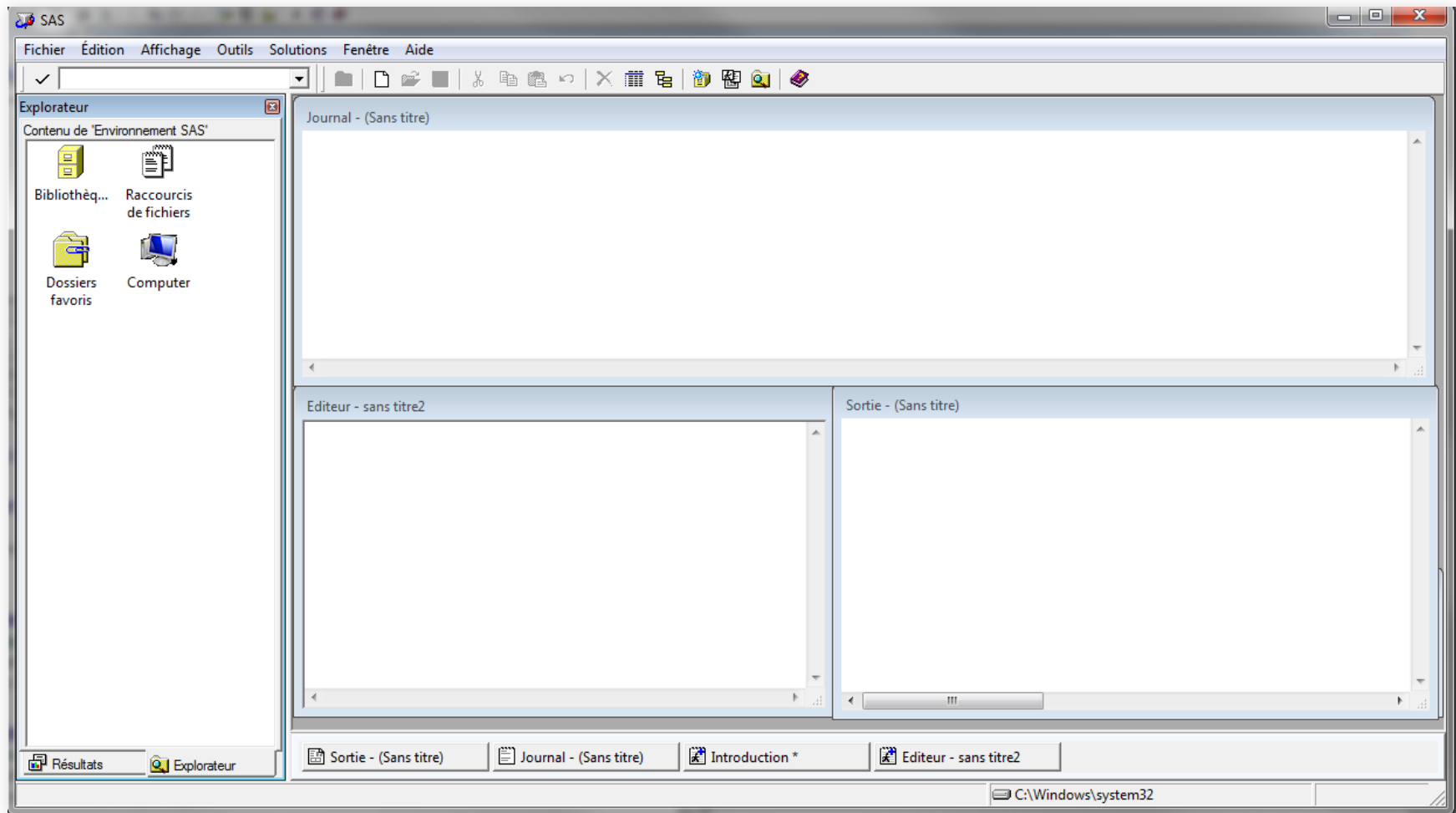
# RÉSULTATS

L'onglet Résultats donne accès à l'ensemble des résultats générés par SAS au cours de la session de travail



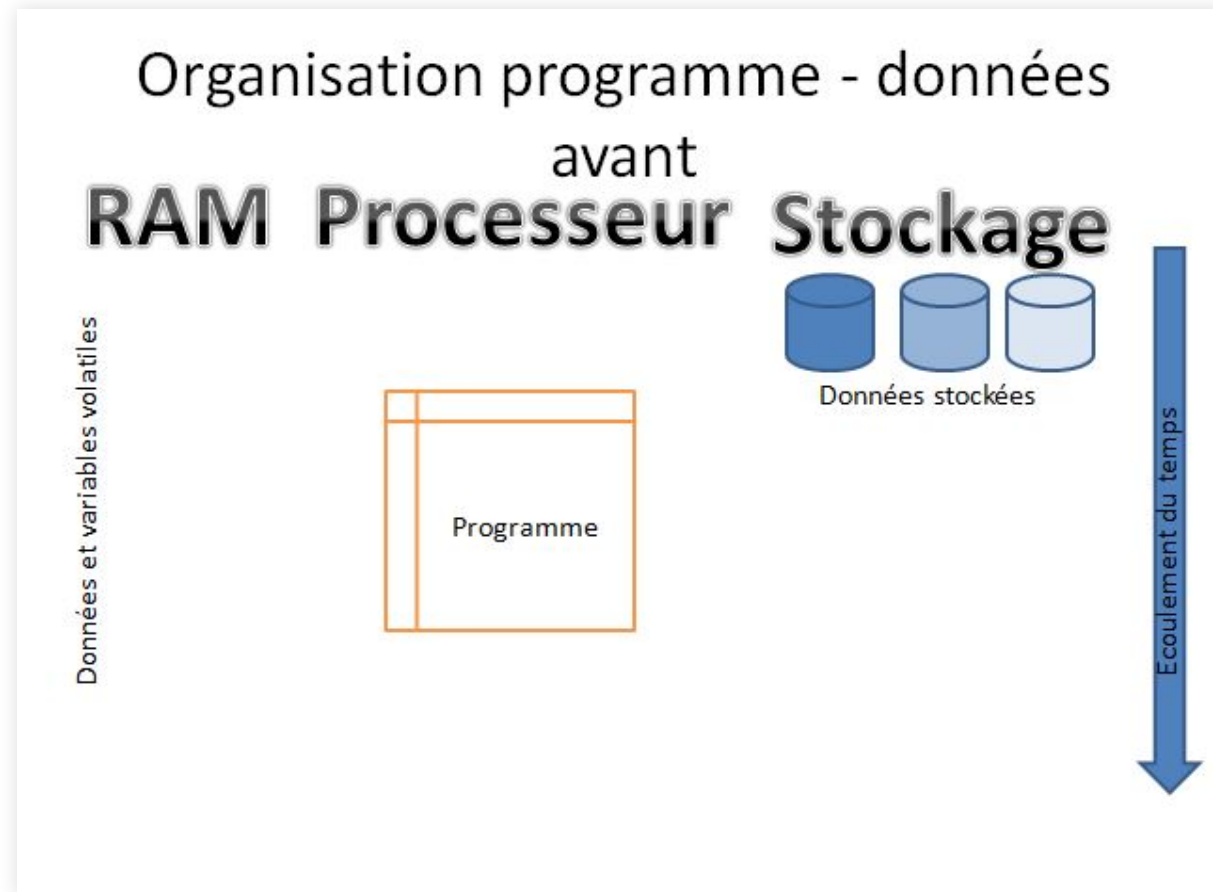


# L'INTERFACE

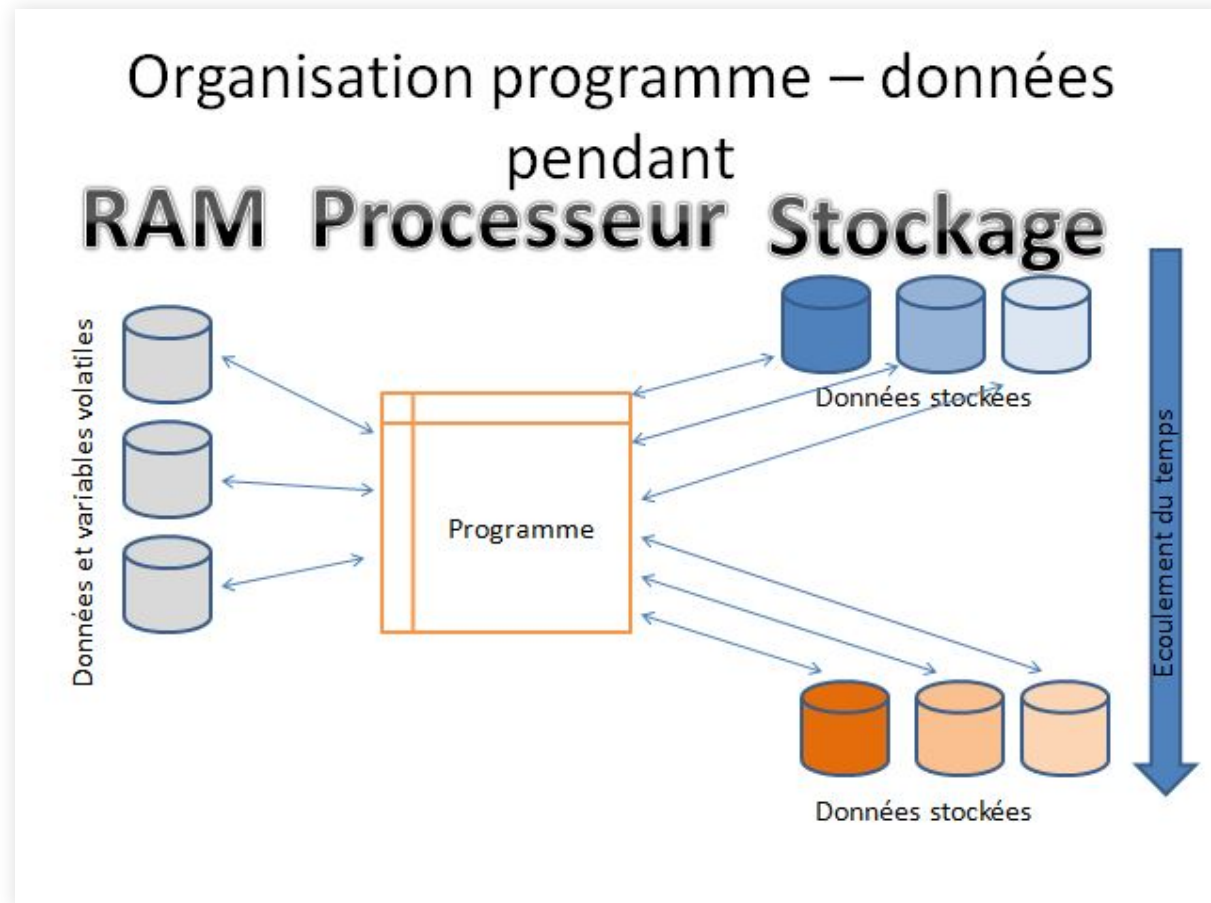


# PRINCIPES DE PROGRAMMATION

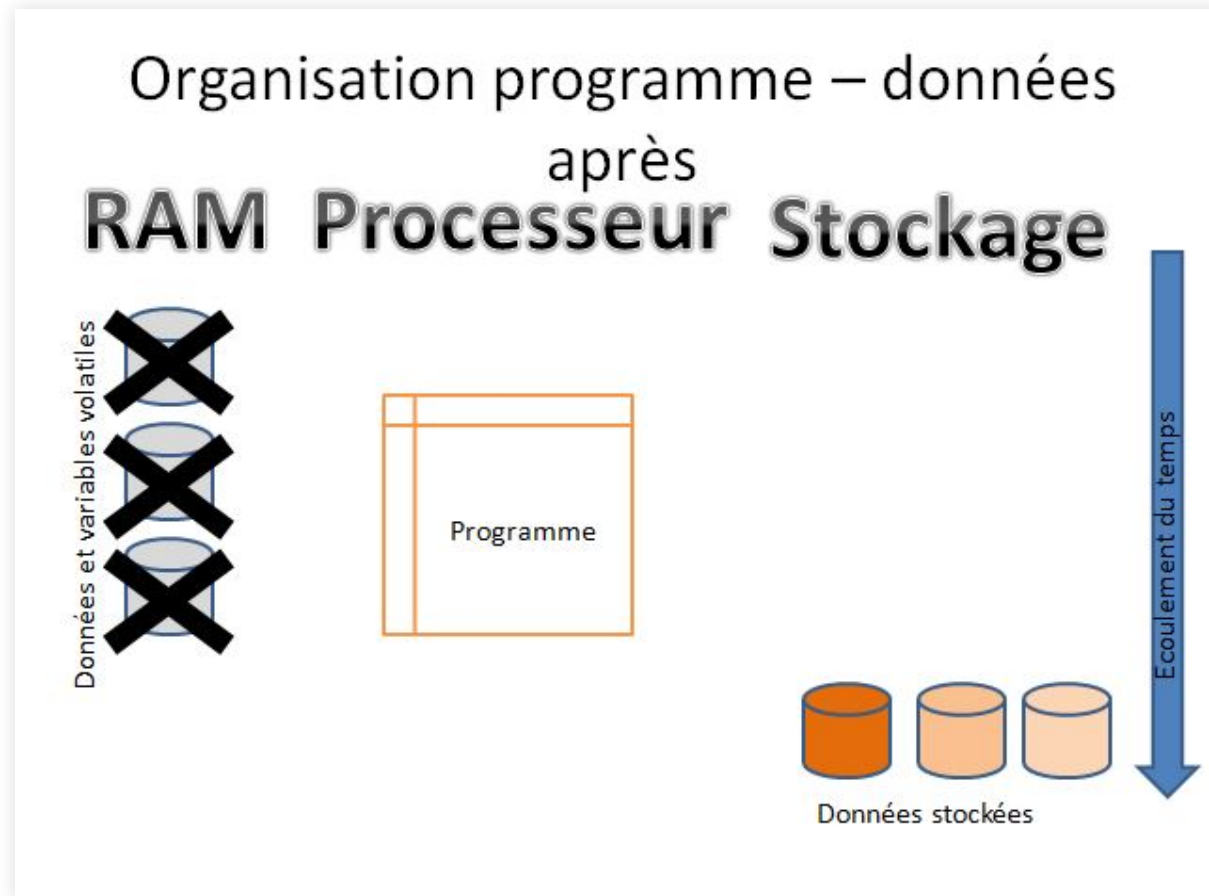
# INTERACTION PROGRAMME SAS AVEC LES DONNÉES-AVANT



# INTERACTION PROGRAMME SAS AVEC LES DONNÉES-PENDANT



# INTERACTION PROGRAMME SAS AVEC LES DONNÉES-APRÈS



# DÉCOMPOSITION D'UN PROGRAMME SAS

Un programme est une suite d'étapes SAS

Principalement de deux types d'étapes nommées

- DATA et
- PROC

# DÉCOMPOSITION D'UN PROGRAMME SAS

Chaque étape est une suite d'instructions  
qui peuvent être accompagnées de mots-clés  
et qui se terminent par un ;

# DÉCOMPOSITION D'UN PROGRAMME SAS

Les étapes se terminent par le mot clé RUN;  
ou dans le cas de certaines étapes PROC le mot clé QUIT;

QUIT; est destiné à libérer la connexion avec les objets manipulés



# ECRITURE D'UN PROGRAMME SAS

L'écriture d'un programme nécessite quelques règles élémentaires pour faciliter sa lecture et les phases de débogage

# ECRITURE D'UN PROGRAMME SAS

Il faut toujours commenter le programme

Pour insérer des commentaires, il existe deux synthaxes

```
/*Insérer des commentaires*/
```

```
*commentaire;
```

# ECRITURE D'UN PROGRAMME SAS

Ne pas hésiter à exécuter fréquemment le programme, en prenant les précautions nécessaires, afin de ne pas accumuler les bugs

# EXÉCUTION D'UN PROGRAMME SAS

L'exécution d'un programme s'effectue en appuyant sur F3 ou à l'aide du bouton 

# EXÉCUTION D'UN PROGRAMME SAS

A chaque exécution, le système complète la fenêtre Journal en générant une suite de notes, d'avertissements ou d'erreurs que nous apprendrons à décrypter

# **ANALYSE DES SORTIES**

Lorsque des tables sont générées par le programme, il convient de toujours vérifier qu'elles correspondent à nos attentes

# ANALYSE DES SORTIES

L'onglet Sortie est alimenté dans le cas de certaines étapes

Cet onglet peut vite s'avérer trop volumineux pour être  
correctement interprété

Ne pas hésiter à nettoyer cet onglet en utilisant la commande  
Ctrl + E

# MON 1ER PROGRAMME SAS : HELLO WORLD

```
data _null_ ;  
    put "Hell" "o World!";  
  
run;
```



# A VOUS DE JOUER !

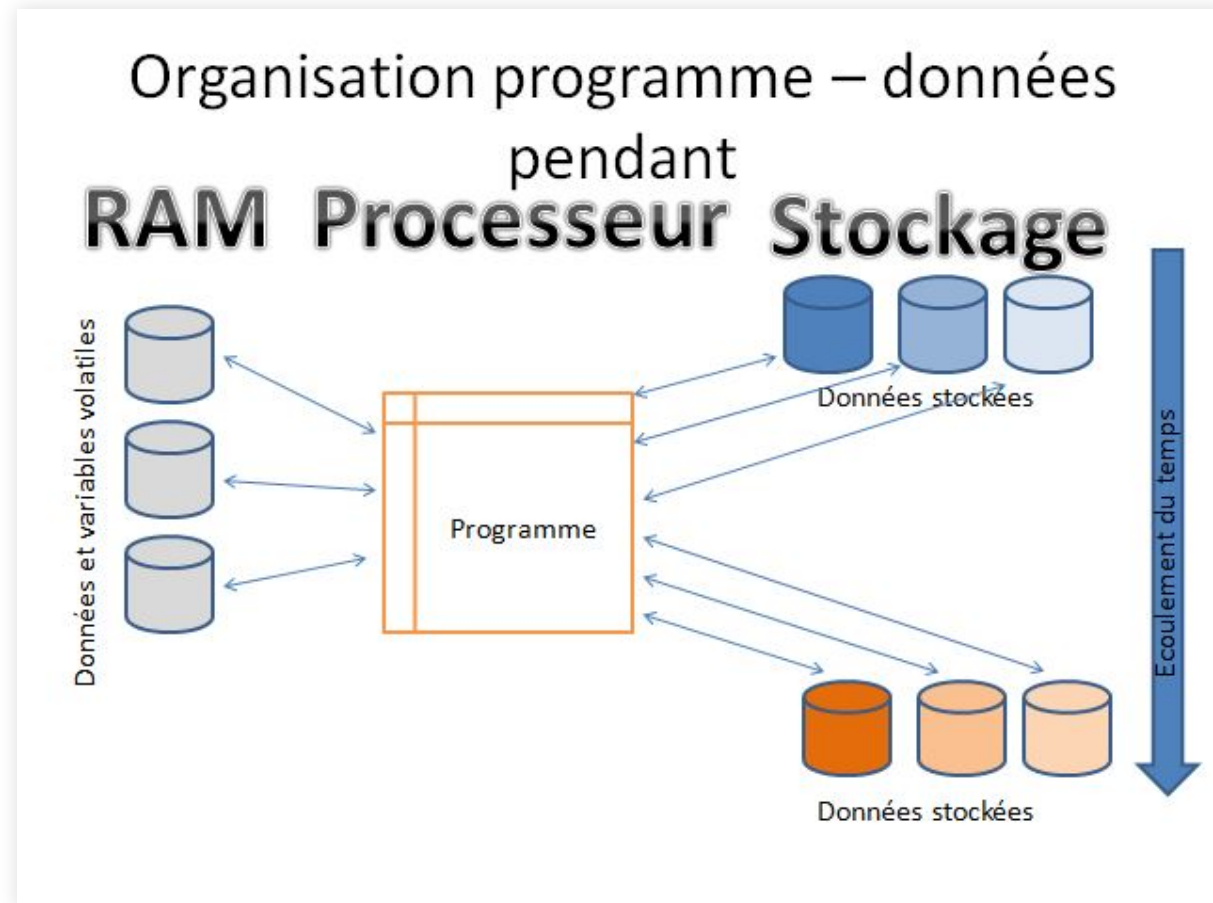
```
4 data _null_;  
5 put "Hell" "o World!";  
6 run;
```

Hello World!

NOTE: DATA statement used (Total process time):  
real time 0.01 seconds  
cpu time 0.01 seconds

# GESTION DES BIBLIOTHÈQUES

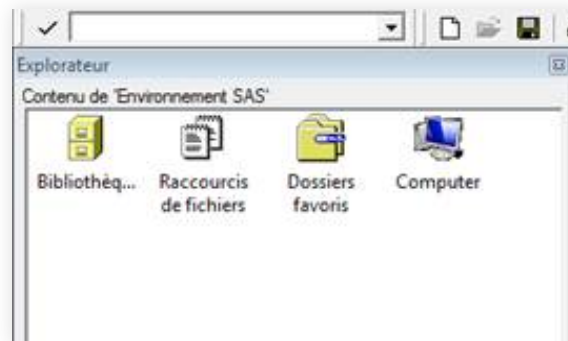
Liaison entre le programme et les stockages de bases de données



# INTRODUCTION AUX BIBLIOTHÈQUES

Le logiciel SAS dispose d'un système de bibliothèques pour la gestion de bases de données

Une bibliothèque (ou librairie) consiste en un nom virtuel attribué à un répertoire physique



C'est ici que vous stockerez les tables (appelées data) qui contiennent les variables en colonnes et les observations en lignes

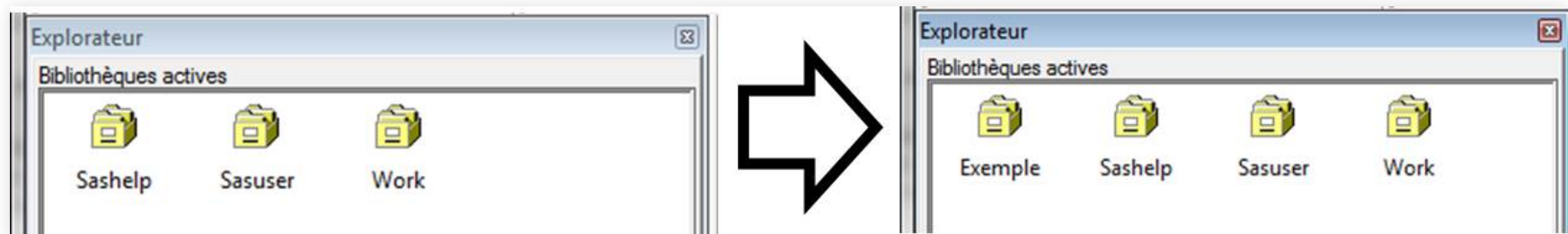
# AFFECTER UNE BIBLIOTHÈQUE

Afin d'affecter une bibliothèque, il suffit de rentrer le code ci-dessous dans la fenêtre Editeur, en respectant la syntaxe suivante :

```
Libname nom_de_ma_librairie 'Chemin vers le dossier de destination';
```

## Exemple

```
Libname exemple 'C:\Users\My Documents\SAS\';
```



# A VOUS DE JOUER !

Libname exemple 'C:\votre\_repertoire\data1';

7 Libname exemple 'C:\votre\_repertoire\data1';

NOTE: Libref EXEMPLE was successfully assigned as follows:

Engine: V9

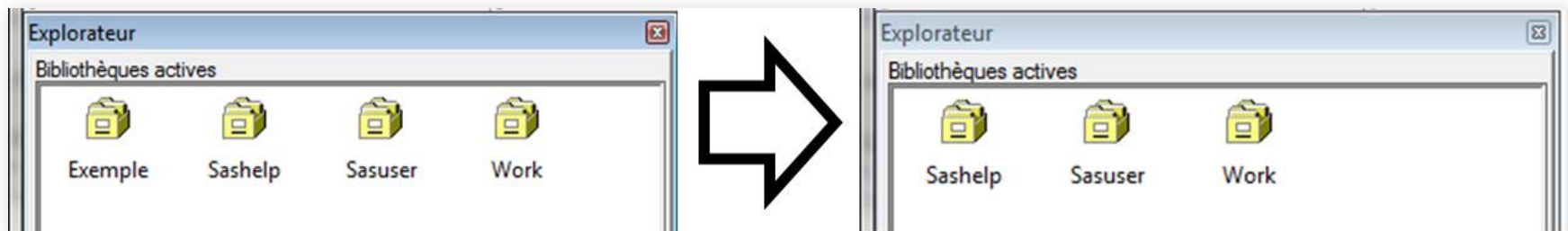
Physical Name: C:\votre\_repertoire\data1

# DÉSAFFECTER UNE BIBLIOTHÈQUE

De manière symétrique, on peut désaffecter une bibliothèque en utilisant l'instruction SAS **clear**

## Exemple

```
Libname exemple clear ;
```



# REMARQUES

Les affectations des bibliothèques se font toujours en début de programme

Par défaut, la seule bibliothèque définie est une bibliothèque temporaire appelée WORK

La WORK est automatiquement réinitialisée à la fermeture des programmes SAS, et tout son contenu est dès lors supprimé

# **ALIMENTATION DES BIBLIOTHÈQUES : L'IMPORT DE DONNÉES**



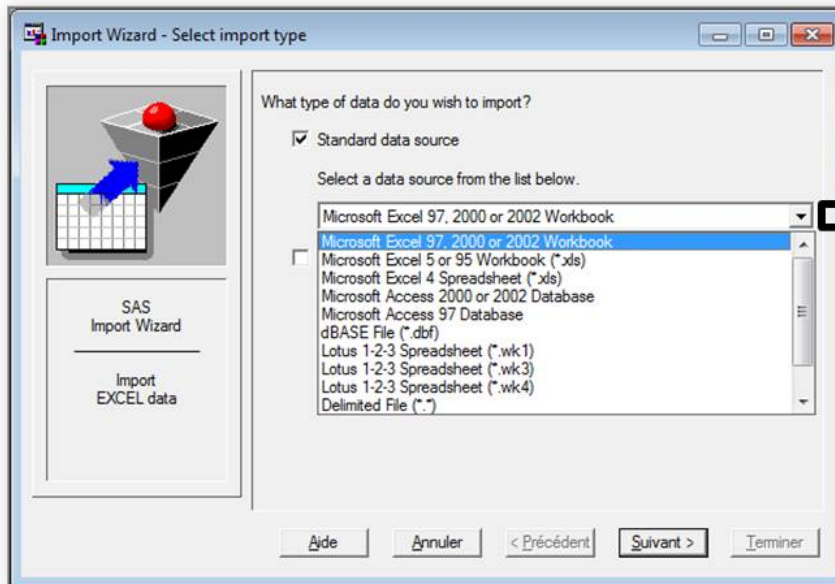
# L'IMPORT DE DONNÉES

Pour importer des tables dans une bibliothèque,  
deux solutions sont envisageables :

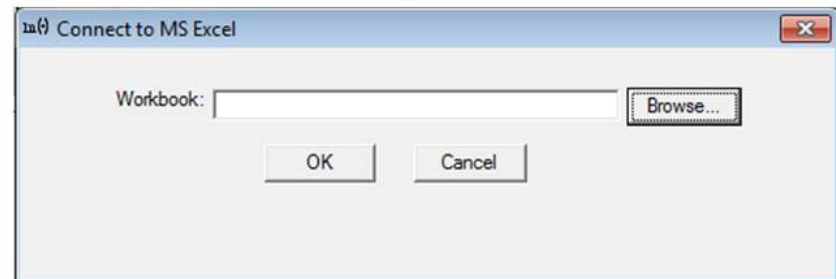
- L'assistant SAS
- La **PROC IMPORT**

# L'ASSISTANT SAS

Dans l'onglet fichier\Importer données...



Sélection du type de fichier à importer  
(xls, csv, etc...)

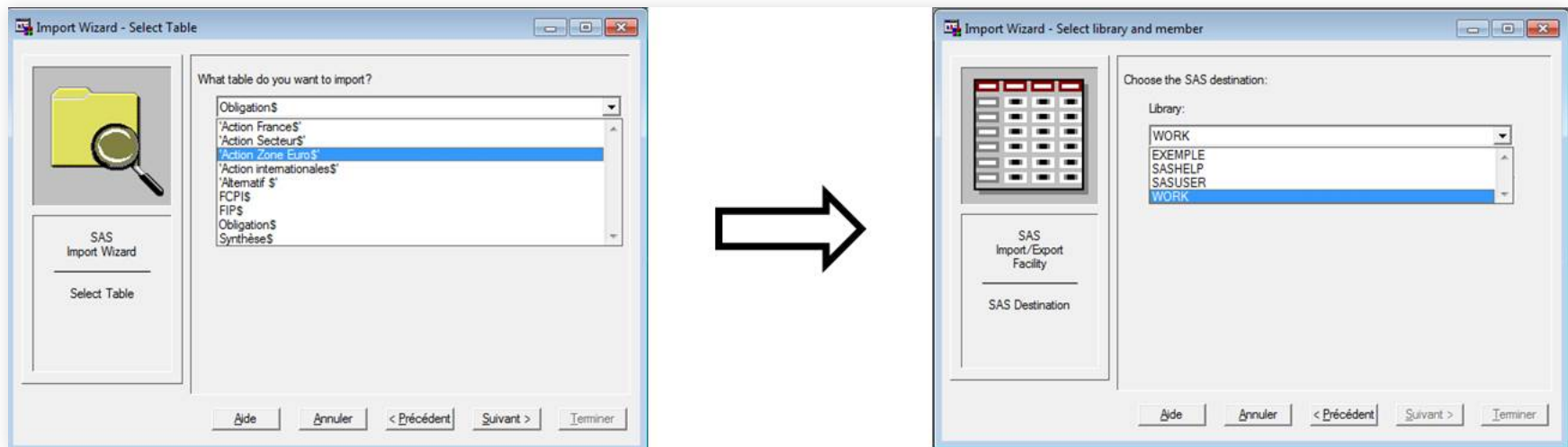


en cas de problème, résolution possible en lançant l'exécutable AccessDatabaseEngine.exe dans :  
SAS Software Depot\products\sysreqwizard\_92210\_prt\_xx\_sp0\_1\redist\ace\en ou dans :  
SAS Software Depot\products\ace\_99140\_prt\_xx\_sp0\_1\w32\native  
ou vérifier les versions de SAS vs Excel

# L'ASSISTANT SAS

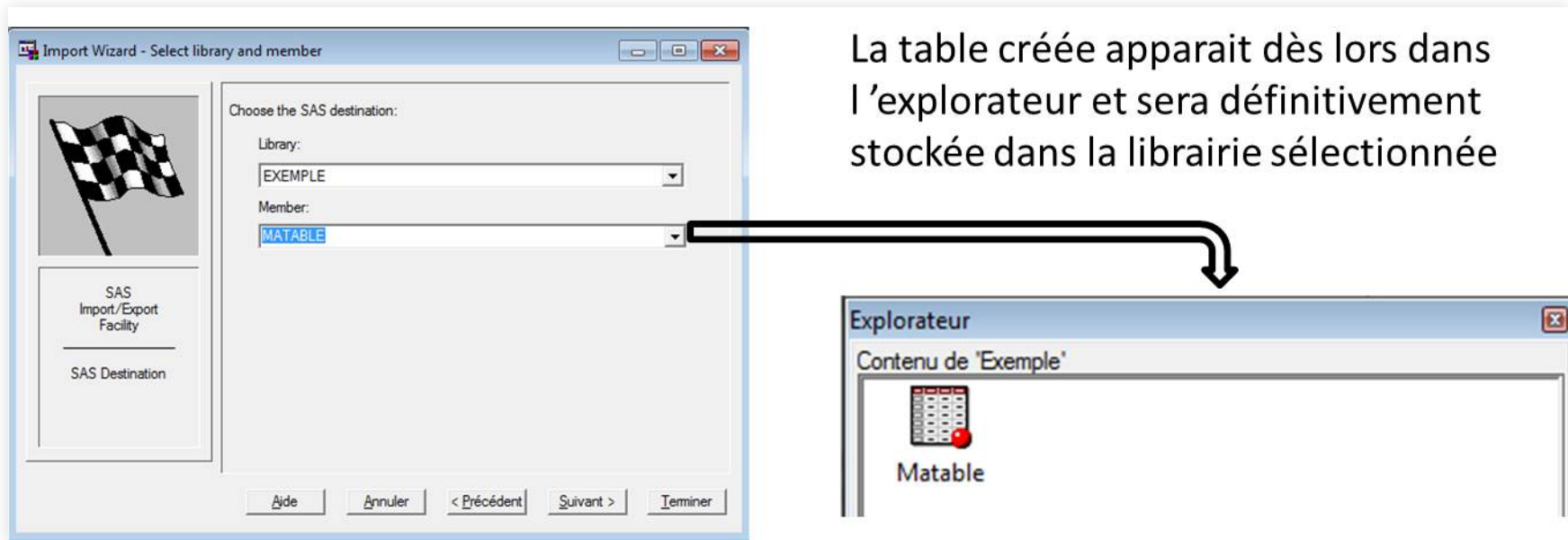
Sélectionner la feuille Excel préalablement préparée pour l'import

Puis sélectionner la librairie dans laquelle la table sera stockée sous SAS



# L'ASSISTANT SAS

Il convient ensuite de donner un nom à la table importée  
Le nom attribué devra être adapté au logiciel, il faut donc proscrire les espaces et les caractères spéciaux



# AUTRE MÉTHODE D'IMPORT DE DONNÉES

Il existe également une procédure SAS qui permet d'importer directement des tables sans utiliser l'assistant

Pour cela il faut maîtriser la proc import et respecter la syntaxe suivante

## PROC IMPORT

```
OUT= nom_de_la_librairie.nom_de_ma_table
DATAFILE= 'Chemin de mon fichier.xls'
DBMS=XLS REPLACE; /*Dans le cas d'un fichier xls*/
SHEET= 'nom_de_la_feuille_Excel_à_importer$' ;
GETNAMES=YES ; /*Conserve les noms de colonnes stockés dans la première ligne
du fichier Excel*/
RUN; /*Instruction qui exécute les instructions qui précèdent*/
```

ancienne version : DBMS=EXCEL2000 REPLACE;

# AUTRE MÉTHODE D'IMPORT DE DONNÉES

Dans l'exemple précédent, le code qu'il aurait fallu insérer dans la fenêtre Editeur aurait été le suivant:

```
PROC IMPORT  
  OUT= EXEMPLE.MATABLE  
  DATAFILE= 'C:\SAS\mon_fichier.xls'  
  DBMS=XLS REPLACE;  
  SHEET= 'ma_feuille$'  
  GETNAMES=YES ;  
RUN ;
```

# AUTRE MÉTHODE D'IMPORT DE DONNÉES

De la même manière, la table aurait été définitivement stockée  
dans la librairie sélectionnée

Le journal aurait alors retourné la note suivante

**Note: EXEMPLE.MATABLE was successfully created.**

# FORMATS ET OPÉRATEURS



# REMARQUES SUR LES FORMATS

Les formats de variables sont cruciaux et s'avèrent être à l'origine de nombreuses erreurs de compilation

Il est important d'être vigilant aux formats des variables lors de leur création ou de leur import

# PRINCIPES DE DÉFINITION DE FORMATS

Les formats SAS sont définis comme suit

< \$ >format< w >.< d >

Code	Signification
\$	Indique un format de type chaîne de caractère
format	Mot clé qui indique à SAS le format que l'on souhaite attribué
w	Indique la longueur totale de la variable incluant espaces et caractères spéciaux
.	Délimiteur entre w et d
d	Indique le nombre de décimales requises dans une variable numérique

# PRINCIPAUX FORMATS SOUS SAS

Les variables caractères ou alphanumériques sont identifiées par le symbole \$

Elles ont une longueur maximale définie lors d'un import comme la longueur maximale de la modalité associée à la variable importée

Code	Signification
\$2.	Ab
\$UPCASE2.	AB

# PRINCIPAUX FORMATS SOUS SAS

Les variables numériques sont identifiées par le nombre de chiffres qu'elles peuvent contenir et un nombre de décimales si besoin

Code	Signification
5.	12345
7.2	1234.56
NUMX7.2	1234,56

# PRINCIPAUX FORMATS SOUS SAS

Les variables dates sont les plus complexes car soumises à différentes conventions nationales

Il s'agit de cas particuliers de variables numériques

Code	Signification
DDMMYY10.	25/07/2014
MMYY5.	07/2014
YEAR4.	2014
FRADFWDX.	25 Juillet 2014
FRADFWDKX.	Vendredi 25 Juillet 2014

# INTRODUCTION AU RISQUE DE TRONCATURE

Un format mal défini peut causer des erreurs lors de la compilation d'un programme

Cependant il peut arriver que le programme parvienne à compiler mais les erreurs de définition de formats peuvent altérer les données de manière dramatique

# INTRODUCTION AU RISQUE DE TRONCATURE

Par exemple, prenons le cas d'une classe de 19 élèves telle que :

	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69	112.5
2	Alice	F	13	56.5	84
3	Barbara	F	13	65.3	98
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83
7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84
10	John	M	12	59	99.5
11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90
13	Louise	F	12	56.3	77
14	Mary	F	15	66.5	112
15	Philip	M	16	72	150
16	Robert	M	12	64.8	128
17	Ronald	M	15	67	133
18	Thomas	M	11	57.5	85
19	William	M	15	66.5	112

**DATA** exemple ;

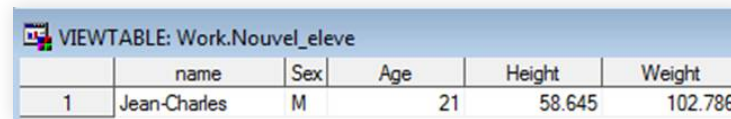
**set** sashelp.class;

**RUN** ;

# INTRODUCTION AU RISQUE DE TRONCATURE

En cours d'année, un nouvel étudiant rejoint la classe

Sa description est contenue dans la table nouvel\_eleve telle que :



	name	Sex	Age	Height	Weight
1	Jean-Charles	M	21	58.645	102.786

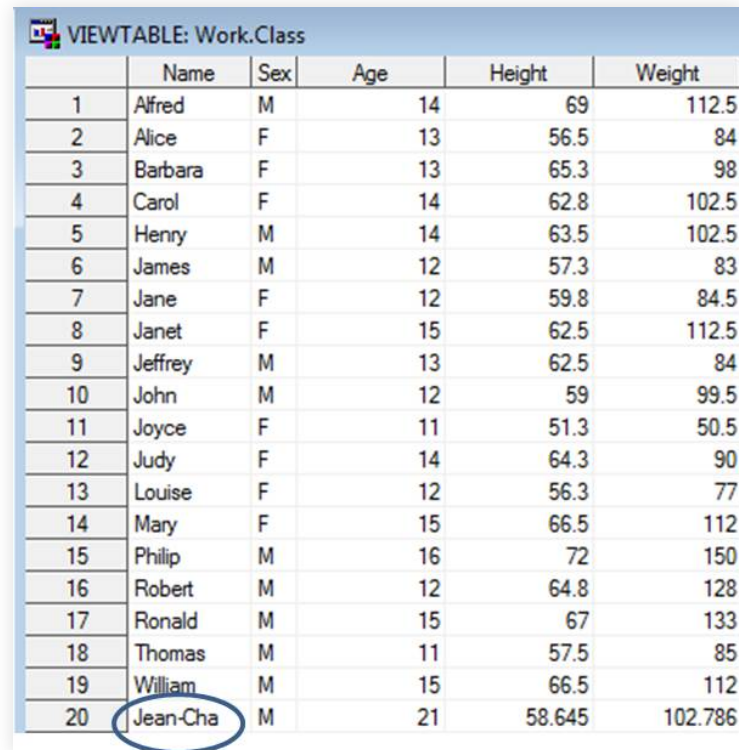
```
DATA nouvel_eleve ;  
    name = 'Jean-Charles' ;  
    Sex = 'M' ;  
    Age = 21 ;  
    Height = 58.645 ;  
    Weight = 102.786 ;  
run ;
```



# INTRODUCTION AU RISQUE DE TRONCATURE

## J'intègre la nouvel élève à la classe

La variable Name initialement était définie par défaut comme étant une chaîne de 8 caractères. Jean-Charles est une chaîne de 12 caractères elle a été tronquée au niveau du 8ème caractère



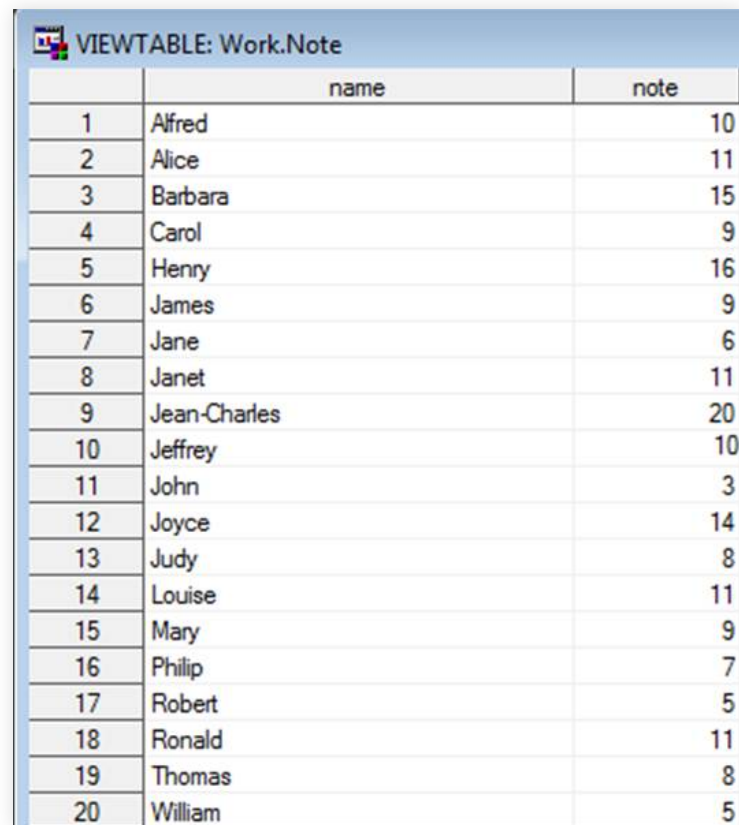
	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69	112.5
2	Alice	F	13	56.5	84
3	Barbara	F	13	65.3	98
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83
7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84
10	John	M	12	59	99.5
11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90
13	Louise	F	12	56.3	77
14	Mary	F	15	66.5	112
15	Philip	M	16	72	150
16	Robert	M	12	64.8	128
17	Ronald	M	15	67	133
18	Thomas	M	11	57.5	85
19	William	M	15	66.5	112
20	Jean-Cha	M	21	58.645	102.786

```
DATA class ;  
    set exemple nouvel_eleve ;  
RUN ;
```

# INTRODUCTION AU RISQUE DE TRONCATURE

Je veux alors implémenter les notes à l'examen stockées dans la table note

La variable Name a cette fois était définie comme une chaîne de 12 caractères maximum



	name	note
1	Alfred	10
2	Alice	11
3	Barbara	15
4	Carol	9
5	Henry	16
6	James	9
7	Jane	6
8	Janet	11
9	Jean-Charles	20
10	Jeffrey	10
11	John	3
12	Joyce	14
13	Judy	8
14	Louise	11
15	Mary	9
16	Philip	7
17	Robert	5
18	Ronald	11
19	Thomas	8
20	William	5

# INTRODUCTION AU RISQUE DE TRONCATURE

Selon la table de référence que je vais sélectionner, le programme va s'exécuter mais 2 types de résultats sont attendus

# INTRODUCTION AU RISQUE DE TRONCATURE

Si la table Class est prise en référence, avec name définie \$7.  
j'obtiens :

VIEWTABLE: Exemple.Releve			
	Name	note	resultat_du_devoir
1	Alfred	10	Alfred a obtenu un 10
2	Alice	11	Alice a obtenu un 11
3	Barbara	15	Barbara a obtenu un 15
4	Carol	9	Carol a obtenu un 9
5	Henry	16	Henry a obtenu un 16
6	James	9	James a obtenu un 9
7	Jane	6	Jane a obtenu un 6
8	Janet	11	Janet a obtenu un 11
9	Jean-Cha	20	Jean-Cha a obtenu un 20
10	Jeffrey	10	Jeffrey a obtenu un 10
11	John	3	John a obtenu un 3
12	Joyce	14	Joyce a obtenu un 14
13	Judy	8	Judy a obtenu un 8
14	Louise	11	Louise a obtenu un 11
15	Mary	9	Mary a obtenu un 9
16	Philip	7	Philip a obtenu un 7
17	Robert	5	Robert a obtenu un 5
18	Ronald	11	Ronald a obtenu un 11
19	Thomas	8	Thomas a obtenu un 8
20	William	5	William a obtenu un 5

# INTRODUCTION AU RISQUE DE TRONCATURE

Si la table Note est prise en référence, avec name définie \$12.  
j'obtiens :

VIEWTABLE: Exemple.Releve			
	Name	note	resultat_du_devoir
1	Alfred	10	Alfred a obtenu un 10
2	Alice	11	Alice a obtenu un 11
3	Barbara	15	Barbara a obtenu un 15
4	Carol	9	Carol a obtenu un 9
5	Henry	16	Henry a obtenu un 16
6	James	9	James a obtenu un 9
7	Jane	6	Jane a obtenu un 6
8	Janet	11	Janet a obtenu un 11
9	Jean-Cha	.	Jean-Cha a obtenu un .
10	Jean-Charles	20	Jean-Charles a obtenu un 20
11	Jeffrey	10	Jeffrey a obtenu un 10
12	John	3	John a obtenu un 3
13	Joyce	14	Joyce a obtenu un 14
14	Judy	8	Judy a obtenu un 8
15	Louise	11	Louise a obtenu un 11
16	Mary	9	Mary a obtenu un 9
17	Philip	7	Philip a obtenu un 7
18	Robert	5	Robert a obtenu un 5
19	Ronald	11	Ronald a obtenu un 11
20	Thomas	8	Thomas a obtenu un 8
21	William	5	William a obtenu un 5

# INTRODUCTION AU RISQUE DE TRONCATURE

Dans le premier cas, la modalité 'Jean-Charles' n'est pas possible. SAS reconnaît alors 'Jean-Cha' dans les deux tables et parvient à faire le lien entre les deux individus.

Dans le second cas, la modalité 'Jean-Charles' est une modalité possible. SAS identifie alors 2 individus, Jean-Cha présent dans la table Class, et Jean-Charles présent dans la table Note.

# INTRODUCTION AU RISQUE DE TRONCATURE

Bien que le programme ne génère pas d'erreur, la mauvaise attribution des formats a dupliqué des individus et généré des valeurs manquantes dans le pire des cas. Dans le meilleur des deux cas, ce n'est qu'une question d'affichage.

# RÉCAPITULATIF DES OPÉRATEURS

## Opérateurs de comparaisons

Code		Signification
LT	<	Inférieur strict (less than)
EQ	=	Egal (equal)
GT	>	Supérieur strict (greater than)
LE	<=	Inférieur ou égal (less or equal)
NE	^=	Différent (non equal)
GE	>=	Supérieur ou égal (greater or equal)



# RÉCAPITULATIF DES OPÉRATEURS

## Opérateurs logiques

Code		Signification
AND	&	Et
OR	!	Ou
NOT	^	Non

# RÉCAPITULATIF DES OPÉRATEURS

## Opérateurs arithmétiques

Code		Signification
+		Addition
-		Soustraction
*		Multiplication
/		Division
x <> y	Min(x,y)	Minimum de x et y
x >< y	Max(x,y)	Maximum de x et y

# RÉCAPITULATIF DES OPÉRATEURS

## Concaténation de chaînes de caractères

Code	Exemple	Signification
!! ou	Var = x  y  z	Concaténation simple
CATS	Var = CATS(x,y,z)	Concaténation en enlevant les blancs en début et fin de chaîne de caractère
CATX	Var = CATX('-',x,y,z)	Concaténation en ajoutant un séparateur entre les variables
CAT	Var = CAT(x,y,z)	Concaténation en conservant les blancs en début et fin de chaîne de caractères
CATT	Var = CATT(x,y,z)	Concaténation en enlevant les blancs uniquement en fin de chaînes de caractères

# FONCTIONS

# FONCTIONS SUR CHAINES DE CARACTÈRES

Ci-dessous sont présentées des fonctions qui prennent en arguments des chaînes de caractères (ici notée x)

Code	Signification
Length(x)	Retourne la longueur de x
Compress(x,'c')	Enlève les caractères C de la chaîne de caractères x
Repeat(x,n)	Répète n fois la chaîne de caractères x en une seule chaîne de caractères
Index(x,m)	Retourne la place du mot m dans la chaîne de caractères x
Uppcase(x)	Met x en majuscules
Lowcase(x)	Met x en minuscules
Substr(x,n,l)	Extraie de x un mot de longueur l à partir du n ième caractère
Scan(x,n,'sep')	Extraie de x le n ième mot en considérant un certain séparateur 'sep'
Tranwd(x,y,z)	Remplace les occurrences du mot y par le mot z

# FONCTIONS SUR DATES

Ci-dessous sont présentées des fonctions en relation avec les dates (ici notée d)

Code	Signification
mdy(m,j,a)	Crée une date de jour j, mois m et année n
Date()	Retourne la date courante
Datepart(d)	Extraie la partie de la date d
Day(d)	Retourne le jour de la date d
Month(d)	Retourne le mois de la date d
Year(d)	Retourne l'année de la date d
Weekday(d)	Retourne le jour de la semaine de la date d

# FONCTIONS MATHÉMATIQUES

Ci-dessous sont présentées les principales fonctions mathématiques prenant en argument une variable numérique (ici notée x)

Code	Signification
Floor(x)	Renvoie la partie entière de x
abs(x)	Valeur absolue de x
Sign(x)	Renvoie 1 si x est positif, -1 si x est négatif et 0 sinon
Round(x,n)	Renvoie l'arrondi de x avec n chiffres après la virgule
Mod(x,y)	Renvoie le reste de la division euclidienne de x par y
sqrt(x)	Renvoie la racine carrée de x
exp(x)	Renvoie l'exponentiel de x
Log(x)	Renvoie le logarithme de x

# FONCTIONS TRIGONOMÉTRIQUES

Ci-dessous sont présentées les principales fonctions trigonométriques prenant en argument une variable numérique (ici notée  $x$ )

Code	Signification
Cos( $x$ )	Renvoie le cosinus de $x$
Sin( $x$ )	Renvoie le sinus de $x$
Tan( $x$ )	Renvoie la tangente de $x$
Arcos( $x$ )	Renvoie l'arcosinus de $x$
Arsin( $x$ )	Renvoie l'arsinus de $x$
Artan( $x$ )	Renvoie l'artangente de $x$



# FONCTIONS STATISTIQUES

Ci-dessous sont présentées les principales fonctions statistiques prenant en argument une série numérique (ici notée  $x_1 \dots x_n$ )

Code	Signification
<code>n(x1... xn)</code>	Renvoie le nombre de valeurs non manquantes de la série (x1... xn)
<code>nmiss(x1... xn)</code>	Renvoie le nombre de valeurs manquantes de la série x1... xn
<code>mean(x1... xn)</code>	Renvoie la moyenne de la série x1... xn
<code>Var(x1... xn)</code>	Renvoie la variance empirique de la série x1... xn

# FONCTIONS ALÉATOIRES

Ci-dessous sont présentées les fonctions qui permettent de générer des nombres aléatoires de paramètres prédéfinis a (généralement égal à 0)

Code	Signification
Rannor(a)	Loi Normale centrée réduite
Ranuni(a)	Loi uniforme sur l'intervalle [0;1]
Ranpoi(a,l)	Loi de poisson de paramètre l
Ranbin(a,n,p)	Loi binomiale de paramètres n et p
Rantbl(a,p1,...,pn)	Loi discrète de distribution p1,...,pn

# FONCTIONS PARTICULIÈRES

Ci-dessous sont présentées deux fonctions particulièrement utiles dans SAS

Code	Signification
Lag(x)	Retourne la valeur précédente de la variable x
Lagn(x)	Renvoie la valeur n fois précédente de la variable x
Dif(x)	Retourne la différence d'ordre 1 de la variable x
Difn(x)	Renvoie la différence entre x et Lagn(x)

# QUELQUES ILLUSTRATIONS AVANT DE PASSER À L'APPLICATION

5-concatenation.sas

6-format\_num.sas

7-formatdate1.sas