Procédure Python de WPS

Guide utilisateur et référence

Version: 4.4.3

www.worldprogramming.com

Table des matières

Procédure PYTHON

La procédure PYTHON permet d'inclure dans un programme WPS en langage SAS du code écrit en langage Python.

La combinaison de Python avec le langage SAS permet d'utiliser la puissance et les performances de SAS pour le gros du traitement et de l'analyse des données, tout en tirant parti des fonctionnalités spécifiques à Python.

Introduction

La procédure Python permet d'inclure dans un programme en langage SAS du code écrit en langage Python.

En combinant les langages Python et SAS, vous pouvez :

- Utiliser le langage SAS pour exécuter la préparation et le traitement de gros volumes de données et transmettre les données traitées à Python.
- Utiliser les packages Python que vous avez développés auparavant pour l'analyse des données.
- Utiliser des packages d'analyse des données Python ou des solutions qui ne sont pas disponibles dans le langage SAS.

Les données sont transmises du langage SAS à l'environnement Python à l'aide de l'instruction `EXPORT`. Une fois les données transférées, elles sont accessibles au programme en Python sous la forme de DataFrames pandas. Une fois le programme en Python terminé, il est possible de renvoyer les données à l'environnement en langage SAS de WPS à l'aide de l'instruction `IMPORT`.

Installation et configuration

Configuration de l'environnement Python pour WPS.

Lorsque vous utilisez Python avec WPS :

- Assurez-vous que vous utilisez le même type d'installation que pour WPS – vous devez utiliser la version Windows 32 bits ou 64 bits pour les deux logiciels.
- Le package `pandas` doit être installé avec l'interpréteur Python. Pour vérifier que c'est le cas, utilisez la commande `pip list` de l'utilitaire pip en ligne de commande.

Le logiciel WPS n'inclut ni interpréteur Python, ni package pandas. Si Python n'est pas installé, vous pouvez vous procurer une copie de l'interpréteur Python auprès de *https://www.python.org* ou installer un package d'environnement Python qui inclut les modules requis.

La procédure `PYTHON` peut être utilisée avec Python 3.5.0 et versions ultérieures, et est actuellement prise en charge sur les systèmes Windows, Linux et macOS.

En revanche, les mainframes IBM ne peuvent pas l'exécuter pour l'instant.

Variables d'environnement de Python

Afin que WPS trouve et utilise l'interpréteur Python, il est nécessaire de définir la variable d'environnement *PYTHONHOME*. Cette variable doit faire référence au dossier où se trouve la principale bibliothèque Python – par exemple, `python3.dll` sur Microsoft Windows.

Flux de sortie et d'erreur standard

Le flux de sortie standard de Python (`sys.stdout`) et le flux d'erreur standard (`sys.stderr`) sont redirigés vers la sortie WPS lors de l'exécution de la procédure :

- `sys.stderr` est redirigé vers le fichier journal de WPS.
- `sys.stdout` est redirigé vers le fichier de listing de WPS lorsque vous exécutez WPS en ligne de commande, ou vers les destinations ODS spécifiées lorsque vous utilisez le Workbench.

Si vous utilisez la fonction `print()` de Python pour ajouter les sorties au fichier de listing de WPS, le nombre de caractères figurant dans la liste est déterminé par l'option système `LINESIZE` de WPS. L'option `LINESIZE` permet d'imprimer des chaînes contenant jusqu'à 256 caractères ; il convient donc d'utiliser la fonction `print()` pour des informations limitées telles que les déclarations de journal.

Si vous voulez renvoyer de gros volumes de données générées par Python vers WPS, créez un DataFrame avec le contenu voulu, et importez-le dans WPS à l'aide de l'instruction `IMPORT`.

L'exemple suivant crée une liste des fonctions disponibles dans le package pandas de Python. La liste est imprimée à l'aide de la commande `print (fnList)`, convertie en DataFrame et importée dans WPS.

```
PROC PYTHON;
SUBMIT;
import inspect
fnPandas = inspect.getmembers(pandas, inspect.isfunction)
fnList = [fn[0] for fn in fnPandas]
print (fnList)
fnTable = pandas.DataFrame({'function': fnList})
ENDSUBMIT;
IMPORT DATA=PANDAFN PYTHON=fnTable;
RUN;
```

La sortie imprimée tronque la liste des fonctions :

```
['Expr', 'Term', 'bdate_range', 'concat', 'crosstab',
 'cut', 'date_range', 'eval', 'factorize',
```

Lorsque la liste est convertie en DataFrame et importée sous forme d'ensemble de données WPS, toutes les fonctions disponibles sont indiquées.

Utiliser Python avec WPS

L'utilisation de la procédure Python dans un programme en langage SAS permet d'utiliser des packages Python spécialisés tels que `Scikit-learn` ou `Tensorflow`.

La première fois que vous invoquez la procédure Python dans un programme en langage SAS, WPS importe automatiquement les packages `pandas` et `numpy`. Vous pouvez accéder aux fonctionnalités des packages `pandas` et `numpy` dans un bloc de programme en Python imbriqué entre les instructions `SUBMIT` et `ENDSUBMIT` en faisant référence au nom complet du package, de la classe ou de la fonction. Par exemple :

```
PROC PYTHON;
  SUBMIT;
content = pandas.read_csv('file:///C:/project/sourcedata/wps.csv')
...
  ENDSUBMIT;
 RUN;
```

Vous pouvez également utiliser l'instruction `importer ... as ...` pour créer un alias du nom de package `pandas` ou `numpy`, par exemple :

```
PROC PYTHON;
  SUBMIT;
import pandas as pd
content = pd.read_csv('file:///C:/project/sourcedata/wps.csv')
...
  ENDSUBMIT;
 RUN;
```

Il est possible d'importer et d'utiliser d'autres packages Python à l'aide de blocs de code en Python. Par exemple :

```
PROC PYTHON;
  EXPORT DATA=source;
  SUBMIT;
import statsmodels.formula.api as lm
result = lm.ols(formula='x ~ y + z', data=source).fit()
...
  ENDSUBMIT;
RUN;
```

Ensuite, chaque utilisation de la procédure PYTHON dans le programme en langage SAS peut utiliser le même environnement Python. Toutes les variables globales ou packages importés utilisés dans une invocation de procédure PYTHON sont donc disponibles pour toutes les invocations ultérieures de cette procédure.

Chaque invocation de la procédure Python peut inclure plusieurs blocs de code en Python imbriqué, et peut utiliser une combinaison de code imbriqué et de programmes en Python lancés par l'instruction `EXECUTE`.

Convertir les types de données

Décrit la correspondance entre les formats du langage SAS et les types de données Pandas.

Cette section décrit la correspondance entre les formats de WPS et les types de données Pandas. WPS comporte de nombreux formats qui affectent la sortie et l'affichage des données. Lorsque vous écrivez des données dans un DataFrame pandas à l'aide de l'instruction `EXPORT` de la procédure Python, les données formatées sont converties dans le type de données pandas ou numpy équivalent.

De nombreux formats n'affectent que la mise en page des sorties de données, telle que l'ajout de symboles de devise ou l'utilisation de virgules comme séparateurs décimaux, et ces formats n'ont aucun effet lors de l'écriture de données.

Données non formatées, de SAS à Python

Les données non formatées sont converties en DataFrame pandas comme suit :

Format WPS	Type de données Python	Remarques
Nombre non formaté	`float64`	
Chaîne de caractères non formatée	`object`	La longueur maximale d'objet pour une variable n'est pas définie dans le cadre des métadonnées de l'ensemble de données.

Données numériques formatées, de SAS à Python

Les principaux formats numériques sont convertis en DataFrame pandas comme suit :

Format WPS	Type de données Python	Remarques
w.d	Float64	
BEST. et BEST*w*.	float64	
FLOAT*w.d*	float64	

Chaînes de caractères formatées, de SAS à Python

Les principaux formats de chaînes de caractères sont convertis en DataFrame pandas comme suit :

Format WPS	Type de données Python	Remarques
$*w*. $CHAR*w*. $F*w*.	object	La longueur maximale d'objet pour une variable n'est pas définie dans le cadre des métadonnées de l'ensemble de données.

Dates et heures formatées, de SAS à Python

Les principaux formats de date, d'heure et de date-heure sont convertis en DataFrame pandas comme suit :

Format WPS	Type de données Python	Remarques
DATE*w*.	datetime64[ns]	Type de date-heure Numpy.
DDMMYY*w*. et toutes les variantes (telles que DDMMYYB*w*., MMDDYYS*w*., et YYMM*w*.)	datetime64[ns]	Type de date-heure Numpy.
DTDATE*w*. et toutes les variantes (telles que DTMONYY*w*. et DTWKDATX*w*.	datetime64[ns]	Type de date-heure Numpy.
TIME*w*., HOUR*w*., HHMM*w*. et tous les formats d'heure similaires.	float64	
JULIAN*w*. et tous les formats de date similaires.	datetime64[ns]	Type de date-heure Numpy.

Des types de données pandas Python aux ensemble de données du langage SAS

Les ensembles de données du langage SAS ne peuvent comporter que des données de type numérique et chaîne de caractères. Il est possible d'appliquer des formats aux données dans l'ensemble de données pour qu'elles représentent mieux les données source Python. L'importation d'un DataFrame pandas à l'aide de l'instruction `IMPORT` convertit les données comme suit :

Type de données Pandas	Format WPS	Remarques
Object	Caractère	La longueur maximale des objets est calculée avant l'importation.
int64	Numérique	
bool	Numérique	*True* est converti en 1 ; *False* est converti en zéro (0).
Float64	Numérique	
datetime64[ns]	Numérique	Formaté comme `DATETIME19.`

Certaines valeurs Python ne peuvent pas être représentées de la même manière dans un ensemble de données SAS. Le tableau suivant montre comment sont converties ces valeurs :

Valeur de données Pandas	Valeur WPS	Remarques
chaîne nulle (`''`)	`' '`	Valeur de caractère manquant.
`True`	`1`	Valeur numérique
`False`	`0` (zéro)	Valeur numérique
`NaN` Défini par `float('nan')`	`.`	Valeur numérique manquante
Infini Défini par `float('inf')` Défini par `float('inf')`	`.`	Valeur numérique manquante

Importer des modules Python personnalisés

Comment utiliser vos propres packages et modules dans Python.

Pour utiliser des packages personnalisés, il faut que Python puisse y accéder. Le chemin des emplacements contenant les packages personnalisés est spécifié par la variable Python *sys.path*. La variable *sys.path* construit une liste d'emplacements où rechercher les packages en utilisant les variables d'environnement *PYTHONHOME* et *PYTHONPATH* ; vous pouvez également spécifier les emplacements en modifiant la valeur de la variable pendant l'exécution d'un programme.

- `sys.path` contient la liste des dossiers où rechercher des packages. La variable est élaborée à partir des valeurs comprises dans *PYTHONHOME* et *PYTHONPATH*.

- *PYTHONHOME* spécifie l'emplacement de l'interpréteur Python et des bibliothèques standard, notamment les packages installés dans *site-packages* à l'aide d'un gestionnaire de packages tel que *PIP*. Cette variable doit être spécifiée pour permettre à WPS de communiquer avec Python.
- *PYTHONPATH* contient la liste des dossiers où rechercher des packages. Cette liste est placée avant la liste de recherche définie dans `sys.path`.

Le premier élément de `sys.path` est soit le répertoire contenant le programme en Python, soit une chaîne vide, interprétée par Python comme le répertoire en cours. S'il est exécuté depuis WPS, `sys.path[0]` contient le premier chemin de recherche spécifié dans *PYTHONPATH* ou dans *PYTHONHOME*. Si votre programme fait références à des packages du répertoire en cours, vous devez modifier `sys.path` quand `PROC PYTHON` est en cours d'exécution. Voir la section Modifier la variable *sys.path* (p. 9)

Définissez *PYTHONPATH* avant de lancer WPS

La variable *PYTHONPATH* peut être définie comme une variable système, et utilisée par toutes les applications de votre dispositif qui exécutent Python.

Si vous avez plusieurs installations de Python, par exemple Python 2 et Python 3, et que vous configurez *PYTHONPATH* avec une variable système, le programme peut tenter d'importer la mauvaise version d'un paquet. Dans ce cas, il convient de définir la variable en tant que partie du programme en langage SAS exécuté dans WPS.

Définir *PYTHONPATH* dans un pogramme en langage SAS

Il est possible de définir *PYTHONPATH* dans un programme en langage SAS à l'aide de l'option système `SET`. Par exemple, pour utiliser les packages stockés dans le dossier `C:\temp\python`, vous pouvez ajouter l'instruction suivante au programme avant l'instruction `PROC PYTHON` :

```
OPTIONS SET = PYTHONPATH 'C:\temp\python';
```

Si la valeur spécifiée pour *PYTHONHOME* est `C:\python3`, le contenu de la variable Python de `sys.path` est :

```
['C:\\temp\\python', 'C:\\python3\\python37.zip',
 'C:\\python3\\DLLs', 'C:\\python3\\lib', 'C:\\python3',
 'C:\\python3\\lib\\site-packages']
```

Modifier la variable *sys.path*

Vous pouvez modifier la variable *sys.path* de manière programmatique en ajoutant la ligne suivante à un programme en Python :

```
import os, sys
sys.path.append(os.getcwd())
```

Si un programme en Python lancé par l'instruction `EXECUTE` inclut d'autres fichiers Python, ajoutez la ligne ci-dessus en tant que programme imbriqué avant le programme exécuté pour permettre à WPS de trouver les fichiers inclus :

```
PROC PYTHON;
  SUBMIT;
import os, sys
sys.path.append(os.getcwd())
  ENDSUBMIT;
  EXECUTE 'programs/dataCollect.py';
RUN;
```

Utilisation des graphiques créés par Python

Tous les graphiques générés à l'aide des fonctionnalités de Python sont capturés, et peuvent être inclus dans les sorties ODS standard de la session WPS.

WPS crée une variable temporaire, *wpsgloc*, qui pointe vers un dossier temporaire utilisé pour stocker les graphiques à inclure dans les sorties ODS. Vous devez ajouter cette variable au nom du fichier d'image chaque fois que vous créez une image. Vous pouvez y parvenir, par exemple, en utilisant la fonction Python `os.path.join()` :

```
PROC PYTHON;
SUBMIT;
import os
import matplotlib.pyplot as plt
...
plt.savefig(os.path.join(wpsgloc, 'my_image.png'))
ENDSUBMIT;
RUN;
```

Il n'est pas possible de modifier la variable *wpsgloc* dans le programme en langage SAS. Elle peut avoir l'une des valeurs suivantes :

- La bibliothèque `WORK` quand le programme est exécuté dans le Workbench, par exemple :

  ```
  C:\Users\user-id\AppData\Local\Temp\WPS Temporary Data
  ```

- Le répertoire de travail quand le programme est exécuté en ligne de commande dans WPS Analytics.

```
ODS PDF FILE='scatter_plot.pdf';
PROC PYTHON;
  EXPORT DATA=STATS PYTHON=df;
  SUBMIT;
import os
import matplotlib.pyplot as plt
df.plot(kind='scatter', x='i', y='j')
plt.savefig(os.path.join(wpsgloc, 'scatter.png'))
  ENDSUBMIT;
RUN;
ODS PDF CLOSE;
```

wpsgloc est un emplacement temporaire pour les images de graphiques utilisées dans le Workbench.

Si le programme est exécuté en ligne de commande, les graphiques sont créés dans le répertoire en cours. Les sorties HTML font référence aux graphiques, les sorties PDF les intègrent.

Exemple

Montre comment utiliser un ensemble de données en langage SAS dans `PROC PYTHON` pour créer un diagramme en nuage de points.

L'exemple suivant crée un ensemble de données dans une étape `DATA` du langage SAS, puis utilise l'instruction `EXPORT` pour passer cet ensemble de données à l'environnement Python. L'ensemble de données est converti en un DataFrame pandas lors de l'exportation, et le DataFrame est utilisé pour créer un diagramme en nuage de points à l'aide de `Matplotlib`.

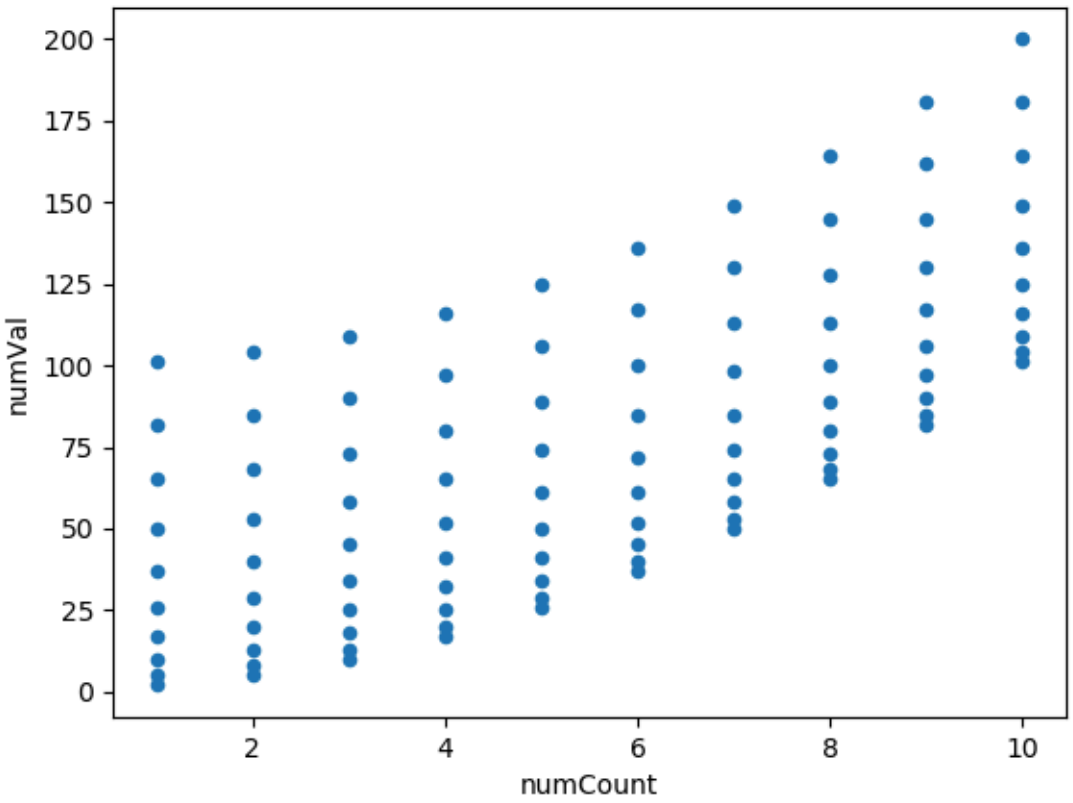
Une destination de sortie fichier PDF est créée à l'aide du système de livraison de sortie (ODS) du langage SAS. Si vous ajoutez PDF aux destinations de sortie, le fichier d'image de diagramme de nuage de points renvoyé est inclus dans la sortie PDF. Le PDF est enregistré et vous pouvez consulter la sortie dans une visionneuse de PDF.

Cet exemple nécessite les packages Python suivants :

- `SciPy`
- `Matplotlib`

```
ODS PDF FILE='scatter_plot.pdf';
DATA stats (drop = count);
  DO count=1 TO 10;
    DO numCount=1 TO 10;
      numVal = (numCount*numCount)+(count*count);
      OUTPUT;
    END;
  END;
RUN;

PROC PYTHON;
  EXPORT DATA=STATS PYTHON=df;
  SUBMIT;
import os
import matplotlib.pyplot as plt
df.plot(kind='scatter', x='numCount', y='numVal')
plt.savefig(os.path.join(wpsgloc, 'scatter.png'))
  ENDSUBMIT;
RUN;
ODS PDF CLOSE;
```

Ceci crée le diagramme de nuage de points suivant dans la sortie ODS PDF :

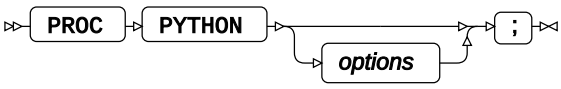


Référence de la procédure Python

Décrit la syntaxe et les options de `PROC PYTHON` et de ses instructions.

PROC PYTHON

Invoque l'environnement Python qui permet d'exécuter des programmes en Python imbriqués ou externes.



Il est possible de mettre les ensembles de données créés dans WPS à la disposition du programme en Python à l'aide de l'instruction `EXPORT`. Inversement, tous les ensembles de données exportés depuis le programme en Python peuvent être chargés dans WPS à l'aide de l'instruction `IMPORT`.

Les programmes en Python peuvent être inclus au sein des programmes WPS dans une procédure Python, ou bien figurer dans des fichiers distincts.

- Pour exécuter un programme en Python inclus au sein d'un programme WPS, utilisez les instructions `SUBMIT` et `ENDSUBMIT`.
- Pour les programmes en Python stockés dans des fichiers externes, utilisez l'instruction `EXECUTE`.

Pour quitter l'environnement Python, il suffit d'utiliser une instruction `RUN`.

Lorsque l'environnement Python est invoqué, WPS charge automatiquement les packages `pandas` et `numpy`. Vous pouvez utiliser ces packages soit en citant le nom complet du package dans le code, soit en utilisant l'instruction `import ... as ...` comme alias du nom de paquet.

Options

Les options suivantes sont disponibles avec l'instruction `PROC PYTHON`.

KEEP
Spécifie que l'environnement Python en cours persiste une fois la procédure terminée.

Si cette option est spécifiée, l'environnement Python en cours reste actif à la fin de la procédure en cours, et est utilisé lors de l'invocation suivante de la procédure Python dans le même programme. Si cette invocation ne spécifie pas `KEEP`, l'environnement est fermé à la fin de la procédure.

Le comportement par défaut consiste à fermer l'environnement Python à la fin de la procédure. Lorsque vous spécifiez `KEEP`, l'environnement Python en cours est conservé, notamment les modules chargés lors de l'exécution d'un programme en Python, et est utilisé à l'invocation suivante de la procédure Python.

Vous pouvez spécifier l'option système `PYTHONKEEP` afin d'utiliser le même environnement Python tout au long de l'exécution du programme en langage SAS.

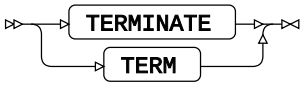
LIB

Spécifie l'emplacement de la bibliothèque pour les ensembles de données utilisés dans la procédure Python. L'emplacement par défaut est la bibliothèque `WORK`.

L'emplacement *référence-bibliothèque* est utilisé si *nom-bibliothèque* n'est pas spécifié dans le chemin de l'option `DATA` de l'instruction `EXPORT` ou `IMPORT`.

TERMINATE

Spécifie qu'il faut fermer l'environnement Python en cours une fois la procédure terminée.



Si vous spécifiez `TERMINATE`, l'environnement Python en cours s'arrête, même si l'option système `PYTHONKEEP` a été spécifiée. Lors de l'invocation suivante de `PROC PYTHON` dans le même programme, seuls les packages `pandas` et `numpy` par défaut sont chargés.

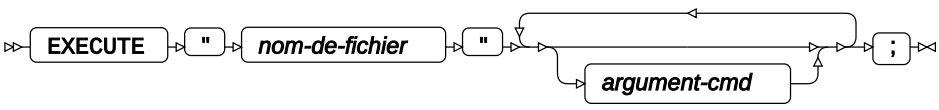
Exemple

L'exemple suivant montre comment invoquer la procédure `PYTHON` pour imprimer `hello world` dans la sortie ODS.

```
PROC PYTHON;
SUBMIT;
print ('Hello World')
ENDSUBMIT;
RUN;
```

EXECUTE

Exécute un programme en Python stocké dans un fichier distinct.



L'instruction `EXECUTE` est l'alias de l'instruction `SUBMIT`. Elle permet d'exécuter dans WPS Analytics le code Python situé dans un autre fichier.

Options EXECUTE

Les options suivantes sont disponibles avec l'instruction `EXECUTE`

nom-de-fichier

Chaîne entre guillemets contenant le chemin d'accès du fichier de programme Python. *nom-de-fichier* peut être un nom de chemin absolu ou relatif.

Dans le Workbench, le chemin des noms de chemin relatifs est l'espace de travail. Par exemple, pour exécuter un fichier nommé `monprogramme.py` depuis un projet nommé `python`, le chemin relatif est `/python/monprogramme.py`.

argument-cmd

Spécifie un argument de ligne de commande passé au programme en Python. L'accès aux arguments dans les programmes en Python se fait depuis `sys.argv`.

Tous les arguments de ligne de commande sont passés au programme en Python sous forme de chaînes. Les arguments numériques doivent être entre guillemets, et le programme en Python doit convertir les arguments au type numérique requis.

Exemple simple

Dans cet exemple, un programme en Python stocké dans un fichier externe est exécuté dans la procédure `PYTHON`. Le fichier est spécifié par son chemin absolu :

```
PROC PYTHON;
  EXECUTE 'C:\temp\printDS.py';
RUN;
```

Exemple – Passer un argument à un programme en Python

Dans cet exemple, un programme externe en Python `multiple.py` est exécuté, et renvoie le carré d'une valeur spécifiée. La valeur est spécifiée dans une variable.

```
import sys

def multiply (value):
  return value*value

print(multiply(int(sys.argv[0])))
```

`multiple.py` est exécuté par la procédure `PYTHON` dans un programme en langage SAS, et la valeur à multiplier est passée au programme en Python sous forme d'argument de l'instruction `EXECUTE`. L'argument numérique *6* est passé sous forme de chaîne de caractères. Le programme en Python utilise la fonction intégrée `int()` pour renvoyer un entier avant de calculer le carré.

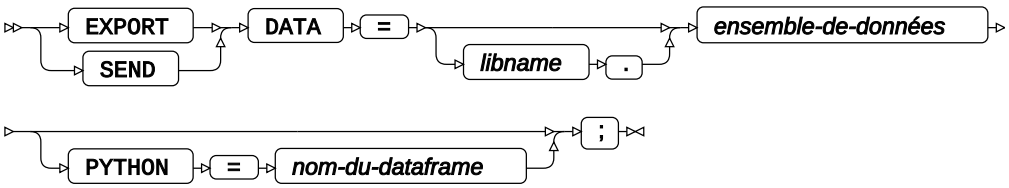
```
PROC PYTHON;
  EXECUTE 'C:/temp/multiple.py' '6';
RUN;
```

Le résultat dans la sortie ODS est le suivant :

```
36
```

EXPORT

Convertit un ensemble de données pour langage SAS en `DataFrame pandas`.



La préparation de l'ensemble de données doit être terminée avant l'exportation des données vers Python. Ceci permet d'utiliser les fonctionnalités de traitement d'ensemble de données du langage SAS pour créer un ensemble de données à exporter ne contenant que les données pertinentes pour le programme en Python.

Options d'exportation

Les options suivantes sont disponibles avec l'instruction `EXPORT`

DATA

Spécifie l'emplacement et le nom de l'ensemble de données SAS à convertir.

L'emplacement de la bibliothèque emplacement peut être spécifié soit en tant que *nom de bibliothèque* de l'option `DATA`, ou en tant qu'option `LIB` de l'instruction `PROC PYTHON`.

- Si vous spécifiez le *nom de bibliothèque*, cet emplacement est toujours utilisé.
- Si l'option `LIB` de l'instruction `PROC PYTHON` est spécifiée, et que le *nom de bibliothèque* n'est pas spécifié, l'emplacement de l'option `LIB` est utilisé.
- Si vous n'avez spécifié ni le *nom de bibliothèque*, ni l'option `LIB` de l'instruction `PROC PYTHON`, l'emplacement `WORK` est utilisé.

PYTHON

Spécifie le nom du `DataFrame` pandas tel qu'il figure dans le programme en Python.

La casse du nom du `DataFrame` pandas dans le code en Python est prise en compte, et la casse de la variable *nom-du-dataframe* spécifiée doit correspondre.

Si cette option n'est pas spécifiée, la valeur par défaut de *nom-du-dataframe* est le nom de l'*ensemble de données* spécifié dans l'option. Si vous utilisez la valeur *ensemble-de-données* par défaut dans un programme en Python imbriqué, le nom de la variable doit utilisez la même casse que dans l'option `DATA`.

Exemple – Exporter un ensemble de données pour langage SAS en DataFrame pandas

L'exemple suivant crée un ensemble de données dans une étape `DATA` du langage SAS. L'ensemble de données est alors exporté vers un `DataFrame` pandas et tous les types de colonne sont indiqués.

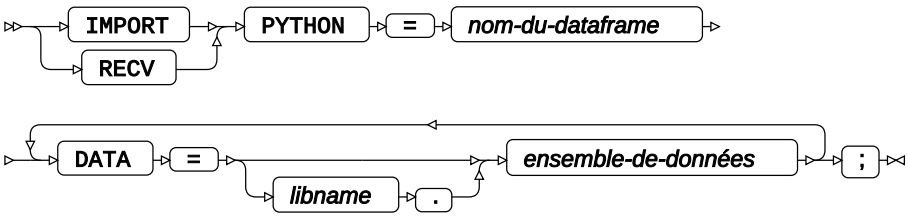
```
DATA TESTDATA;
INFILE CARDS DLM='#';
INPUT num char $;
CARDS;
1 # Hello
2 # World
;
RUN;

PROC PYTHON;
EXPORT DATA=TESTDATA PYTHON=dframe;
SUBMIT;
print (dframe)
ENDSUBMIT;
RUN;
```

Produit la sortie ODS suivante :

```
   num      char
0  1.0      Hello
1  2.0      World
```

IMPORT

Permet de convertir un `DataFrame` pandas en un ensemble de données pour langage SAS et d'y faire référence dans un programme en langage SAS.



Options d'importation

Les options suivantes sont disponibles avec l'instruction `IMPORT`

DATA

Spécifie l'emplacement et le nom de l'ensemble de données dans l'environnement de langage SAS de WPS Analytics.

L'emplacement de la bibliothèque emplacement peut être spécifié soit en tant que *nom de bibliothèque* de l'option `DATA`, ou en tant qu'option `LIB` de l'instruction `PROC PYTHON`.

- Si vous spécifiez le *nom de bibliothèque*, cet emplacement est toujours utilisé.
- Si l'option `LIB` de l'instruction `PROC PYTHON` est spécifiée, et que le *nom de bibliothèque* n'est pas spécifié, l'emplacement de l'option `LIB` est utilisé.
- Si vous n'avez spécifié ni le *nom de bibliothèque*, ni l'option `LIB` de l'instruction `PROC PYTHON`, l'emplacement `WORK` est utilisé.

PYTHON

Spécifie le nom du `DataFrame` pandas tel qu'il figure dans l'environnement Python. Doit être spécifié.

La casse de la variable *nom-du-dataframe* est prise en compte, et la casse utilisée pour le DataFrame pandas dans le programme en Python doit correspondre.

SUBMIT

Spécifie le début d'un programme Python imbriqué.

Le terme « programme imbriqué » désigne une partie de la procédure Python dans un programme en langage SAS. L'instruction `SUBMIT` signale le début du programme, et l'instruction `ENDSUBMIT` signale sa fin.

Le programme en langage Python doit commencer sur une nouvelle ligne après l'instruction `SUBMIT`, et l'instruction `ENDSUBMIT` doit figurer seule sur sa ligne. La première ligne du code de programme en ne doit pas commencer par un espace, et le reste des instructions du bloc doivent respecter les règles de retrait normales de Python, par exemple :

```
PROC PYTHON;
  SUBMIT;
fruits = ['apple', 'banana', 'cherry', 'damson', 'elderberry', 'fig']
for fruit in fruits:
    print(fruit)
  ENDSUBMIT;
RUN;
```

Une même procédure Python peut contenir plusieurs programmes en Python imbriqués. Chaque programme en Python est exécuté lorsqu'il se présente. Les variables définies dans un programme imbriqué peuvent être réutilisées dans les programmes imbriqués suivants de la même procédure Python.

ENDSUBMIT

Spécifie la fin d'un programme Python imbriqué.

L'instruction doit figurer en début de ligne après le programme en langage Python.

Notices légales

Marques

WPS et World Programming sont des marques commerciales ou des marques déposées de World Programming Limited dans l'Union européenne et dans d'autres pays. Le sigle (r) ou ® indique l'enregistrement au niveau de l'Union européenne (« marque communautaire »).

SAS et tous les autres noms de produits et de services de SAS Institute Inc. sont des marques déposées ou des marques commerciales de SAS Institute Inc. aux États-Unis et dans d'autres pays. ® indique que la marque est déposée aux États-Unis.

Toutes les autres marques commerciales sont la propriété de leurs détenteurs respectifs.

Notices générales

World Programming Limited n'est associé d'aucune manière à SAS Institute Inc.

WPS n'est pas le système SAS.

Les expressions « SAS » et « langage SAS » utilisées dans ce document font référence au langage de programmation SAS qui est souvent désigné par ces termes.

Les expressions « programme », « programme SAS » et « programme en langage SAS » utilisées dans ce document font référence aux programmes écrits en langage SAS. Ils peuvent également être appelés « scripts », « scripts SAS » ou « scripts en langage SAS ».

Les expressions « IML » et « langage IML », « syntaxe IML » et « Interactive Matrix Language » utilisées dans ce document font référence au langage de programmation informatique qui est souvent désigné par ces termes.

WPS inclut du logiciel développé par des tiers. Vous trouverez plus d'informations dans le fichier THANKS ou acknowledgements-fr.txt inclus dans l'installation de WPS.