ChemScript 12.0

About this document

This document is the "ChemScript" section of the manual *Chem & Bio Office® Chem& Bio3D, Finder & Bio Viz* and is made available as an excerpt for fast downloading.

To read the manual in its entirety or to download other sections, see the desktop support site at www.cambridgesoft.com.

Contents

14

ChemScript

ChemScript is the cheminformatics Software Development Kit (SDK), a library of the “chemical intelligence” programming algorithms that are prevalent throughout CambridgeSoft products. As a software developer, you can create your own scripts for your particular business needs.

All example script files in the ChemScript library are available in Python and .NET. If you are familiar with either of these languages, you will find these scripts easy to understand. However, if you are new to either Python or .NET, we suggest that you look at some of the Web sites and books listed at the end of this chapter.

Why use ChemScript?

ChemScript adds considerable versatility to how you manage your chemical data. Using ChemScript, you can modify, view, and transfer your data from one place to another using your own custom rules.[1]

For example, you can:

- Process chemical data in novel ways.
- Integrate cheminformatics applications.
- Develop new cheminformatics applications or Web services.

1. ChemScript lets you convert up to 10,000 data records per day. For greater capacity, you will need ChemScript Ultra. For information, contact CambridgeSoft.

Here are just a few common uses for ChemScript:

Salt splitting and stripping. You can identify and remove salt fragments from a structure drawing and register the pure compound.

Canonical codes. Generate canonical codes for a set of structures and use the codes to find duplicates in your data.

File format conversion. Convert structure or reaction files from one format to another.

Generate properties. Execute Struct=Name or generate physical property features found in Chem & Bio Draw.

Common scaffold orientation. Enforce standard orientations of structures based on the established orientation of a common substructure.

2D Structure Diagram Generation (SDG) and Cleanup. Generate new 2D structures from connection tables without coordinates and clean up existing 2D structure using the Chem & Bio Draw algorithms.

About Python

Although ChemScript is available in Python and .NET, we will use Python throughout this guide to explain ChemScript. Python is a non-proprietary and widely used programming language. It provides clear syntax, object-oriented programming, dynamic data typing, and high performance across a broad range of platforms.

Here are some of Python's features:

Human Readable Code. Easy to understand and maintain.

Powerful Syntax; Simple to Use. You can write concise, useful programs using a few lines of code.

Functional Programming. Full language support for functions, control flow, and iteration.

High Level. Language support for container objects and utility classes.

Widely Adopted. A large user base continually provides new additions and contributions.

Object Oriented. Full support for classes, inheritance, and polymorphism.

Connected. Bindings and interfaces for most common languages and technologies.

The Python community has developed a rich set of extensions to Python that are freely available at the Python Web site. These extensions provide database connectivity, server-side Web functionality, numeric processing, language interop, GUI features, and just about anything else that is needed for rapid software development.

How ChemScript works

The most fundamental purpose for ChemScript is to read data from one source, modify the data using a script, and write the modified data to another location. Where the data is retrieved from or written to can be almost any database, file(s), or application.

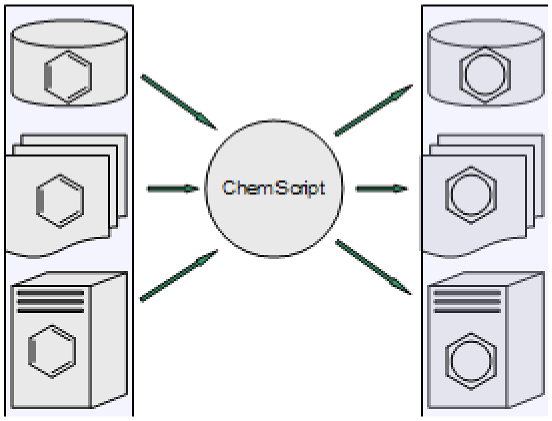

Figure 14.1 A ChemScript script can retrieve data from one source, modify the data, and write it to another location.

How the data is modified is determined entirely by the script. The script can delete data, calculate or add new data, or edit existing data. The data can be either text, structures, or both.

Since ChemScript scripts are the same as any other Python scripts or .NET program, you can execute them using either the Windows command line or any development environment.

Getting Started

For editing ChemScript files, we recommend that you use one of the many programming tools that are available. The Python installation provides one such tool, IDLE, installed as part of the ChemBioOffice 2010 suite. To learn more about IDLE, see the online Help guide in the IDLE main menu.

By default, ChemScript 12.0, IDLE, and Python 2.5 are installed on your local computer when you install Chem & Bio Office 2010. After the installation, we suggest that you go through the simple exercises in this section to familiarize yourself with ChemScript and IDLE.

STARTING IDLE

From the Start menu, go to **All Programs**>**Python 2.5**>**IDLE (Python GUI)**. Python is installed correctly if a Python shell is opened in IDLE and the header indicates a version of Python (Ex. 2.5), and a version of IDLE (Ex. 1.2). A prompt will also appear: “>>>”

LOADING THE CHEMSCRIPT LIBRARY

At the command prompt, type the line below and press enter:

```
from ChemScript12 import *
```

NOTE: The command is case-sensitive.

The ChemScript library is loaded. If a “Welcome to CS ChemScript” message appears, followed by a command prompt, then ChemScript is installed correctly.

CHEMSCRIPT HELP

You can read a description of any ChemScript class within IDLE. For example, enter the line below to return Help for the ChemScript `Atom` class.

```
help(Atom)
```

The help will begin with a message such as:

> “Help on class Atom in module ChemScript12:”

ENTER SMILES DATA

Type the following line and press **Enter**:

```
myMol = Mol.LoadData('C1CCCCC1C')
```

This message appears:

```
Open molecule successfully: chemi-
cal/x-smiles
```

REPORTING A CHEMICAL NAME

At the command prompt type the line below and press **Enter**:

```
myMol.chemicalName()
```

The line `methylcyclohexane` is returned.

COUNTING ATOMS

At the command prompt type the following line and press **Enter**:

```
myMol.CountAtoms()
```

The command returns the atom count for the structure defined with the chemical/x-smiles format for 'C1CCCCC1C', which is 7.

EXITING IDLE

At the command prompt type the line below and press **Enter**:

```
exit()
```

Confirm any prompts to complete the exit command. Python IDLE exits.

Getting Started Guide

ChemScript 12.0 includes a Getting Started guide to help you begin developing and using your own scripts. To open the guide, go to **Start**>**All Programs**>**ChemBioOffice2010** >**ChemScript 12.0**>**Getting Started**.

The document includes notes on the ChemScript objects and functionality, Python, and an overview of examples located on the file system.

Editing Scripts

Using IDLE or another development environment, you can either edit any of the scripts provided with ChemBioOffice or create one of your own. Regardless of how you develop a script, it must include these commands:

```
from sys import *
from os import *
```

```
   from os.path import *
   from ChemScript12 import *
```

The first command imports the python system. The second and third commands import the operating system modules. The last command imports all the ChemScript functions. After you include these command lines, how you develop the rest of your script is entirely up to you.

Introducing the ChemScript API

ChemScript 12.0 includes a ChemScript API reference guide. You can find the guide at **Start**>**All Programs**>**ChemBioOffice 2010**>**ChemScript 12.0**>**API Reference.** It provides links and information for the ChemScript object classes.

The ChemScript object model comprises two fundamental levels of functionality, described below.

ChemScript Object Classes

At the top level, the API consists of four object classes:

Atom. Chemical element, charge, bonds to neighboring atoms, drawing coordinates, 3D coordinates (if available), stereochemistry, etc.

Bond. Bonded atoms, bond order, etc.

Molecule (Mol). A chemical connection table, which can represent one or more molecular fragments. This class also includes file I/O capabilities and other advanced chemistry functionality such as stereochemistry.

Reaction (Rxn). A chemical reaction with one or more steps.

Functions and Algorithms

The secondary level consists of the core set of high-level features that you can modify to meet your specific business needs. Some examples are described below.

Template Based Normalization. Enforce standard representations of functional group structures in chemical data.

Template Based Product Generation. Automatic generation of products from a set of reactants and a generically defined reaction. For example, reactions like those between amines and carboxylates.

Substructure Identification and Mapping.
Atom-by-atom comparison of a molecule with a substructure. Positive matching provides an atom-by-atom map of the substructure atoms to those in the molecule.

Salt Stripping. Remove salts from a reaction based on a pre-defined list of salt fragments.

Structure Orientation. Enforce standard orientation of structures based on the established orientation of a common scaffold.

2D Structure Generation and Cleanup. Use Chem & Bio Draw-based algorithms to generate structure from scratch or after modifying chemical data using a program.

Canonical Codes. Generate unique identifying codes from a chemical structure.

File Format Conversion. Read and write file data using all CambridgeSoft supported file formats (CDX, CDXML, MOL, CHM, SKC, SMILES, etc.).

Chemical Name and Structure Conversion[1].
Use the Chem & Bio Draw Struct=Name feature to generate structures from chemical names and names from their structures.

1. Premium functionality that may be licensed from CambridgeSoft.

Molecular Mechanics. Optimize molecular structures using the MM2 force-field.

The ChemScript API online

CambridgeSoft provides the API online. You can find the API at sdk.cambridgesoft.com.

Tutorials

We provide several sample scripts to illustrate how you can develop your own custom code to meet your business needs. Many of the scripts we use are in the ChemScript samples directory. By default, this directory is where Chem & Bio Office 2010 is installed:

```
C:\Program Files\Cambridge-
Soft\ChemOffice2010\ChemScript
12\Samples
```

For the sake of brevity, we won't repeat the scripts in this manual or try to teach the Python language. However, we briefly describe what you can do with the code examples so that you can modify and expand upon them for your own use. As you read the tutorials, you are encouraged to view the code in IDLE and edit it as desired to see how each example works. For more on IDLE, see "Using ChemScript" on page 5.

Example 1: Automated Structure Clean up

This sample script cleans up the structures in multiple Chem & Bio Draw files all at the same time. It uses the same cleanup function used in Chem & Bio Draw. You can find the script at `Example.001/script.py`. The script reads the CDX structure files from a source directory, applies the cleanup feature to each structure, and write the modified files to an output directory. The original files remain unchanged.

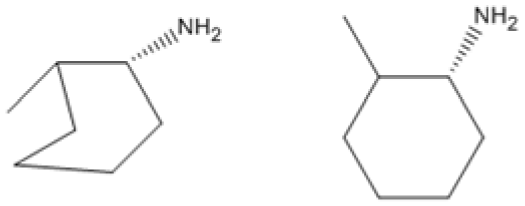

Figure 14.1 The structure cleanup script reads a structure file (left) and creates a new, cleaned up structure file (right).

Although this example uses the CDXML format, other formats such as MDL MOL may also be applied. You can also force ChemScript to use specific file formats for reading and writing data.

Example 2: Create an SD file

The ChemScript examples include a script that illustrates how you can create an SD file from existing CDXML files. You can find the script at `Example.002/script.py`. We begin with a list of CDXML files that each contain a chemical structure. The list of files is hard-coded into the ChemScript script. When executed, the script uses the `SDFileWriter` method to create an SD file that includes all the structures.

Example 3: Create a list of CDXML files

This example illustrates how to read an SD file and write a list of CDXML files. The source files is at `Example.003/script.py`.

Example 4: Filter an SD file

This example uses the `atomByAtomSearch` method to demonstrate a simple application of the "atom-by-atom" substructure search in ChemScript. The program reads an SD file and filters structures into one of two output SD files, structures that contain a phenyl group and structures that don't. It also illustrates how you can read chemical data formatted as a SMILES string. See `Example.004/script.py`.

Example 5: Computing Canonical Codes

This example script checks whether any structures appear in both of two SD files based on the structures' canonical codes. The output is a new SD file with the duplicate structures excluded. See `Example.005/script.py`.

This example first computes the canonical code for each structure. Since the canonical code does not vary with different representations of the same chemical structure, it can be used to determine whether two structures are chemically equivalent.

This example also introduces the Python Dictionary, which is an associative array. The dictionary maps a key to a value. The dictionary is used to determine whether a canonical code has been previously encountered.

This example uses an alternate looping construct to read an SD file.

CAUTION

Canonical codes should never be permanently stored because their representation can change among different versions of ChemScript.

Example 6: Simple salt stripping

The program reads an SD File, identifies and removes salt components (if any are present), and outputs two SD files. The output structure file contains the original structures without the salt component, and the output salt file contains the salt components that were stripped, along with a reference to the original structure.

NOTE: This example uses a default set of salts that CambridgeSoft provides. However, you can also define a customized salt table that enables you to designate which chemicals are considered salts.

Example 7: Structure Overlay

This script introduces the ChemScript structure overlay feature. It uses a scaffold structure file to superimpose two chemically similar structures. The script first examines the structures in an SD file that contain a common scaffold substructure. It then aligns these structures so that they have the same orientation with respect to the scaffold. See `Example.006/script.py`.

NOTE: The overlay functionality can also be used to align three dimensional structures.

Example 8: Reaction Transformation

This example demonstrates reaction transformation. This means that you can draw a reaction that defines a transformation of a molecule and then apply that transformation to a set of structure files.

All the files necessary for this tutorial are in the `Example.007` directory. The `transform1.cdxml` file provides the reaction that defines the transformationFigure 14.2.

The files in the Input directory contain all the structures that will be transformed.

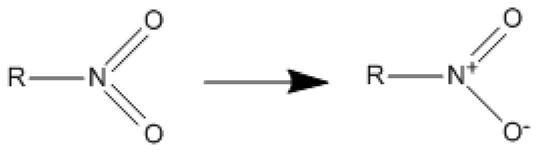

Figure 14.2 The transform file defines how the transformation is applied to the source files.

The script searches the input files for structures that contain a nitro group, shown as a reactant in the transformation file. If a structure is found, the script transforms the nitro group to the form shown in the product and copies the entire structure to a new file. One example is shown in the figure below.

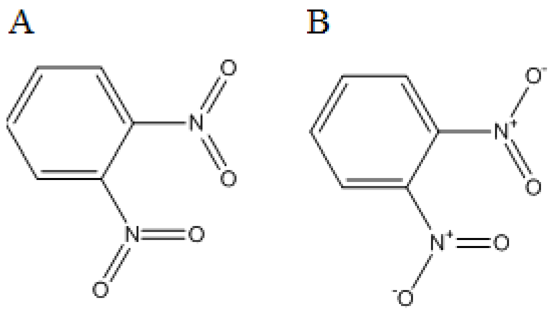

Figure 14.3 A) before the transformation is applied; B) after transformation.

Files that don't meet the search criteria are ignored.

Useful References

There are numerous resources available for learning Python and .NET. Just a few of the many books and Web sites are listed below.

Books

Python

In addition, there are many books about the Python language that are available.

- *Beginning Python: From Novice to Professional* by Magnus Lie Hetland.
- *Dive into Python* by Mark Pilgrim.
- *Learning Python* by Mark Lutz & David Ascher. This is a beginner/intermediate learning manual and reference.
- *Python in a Nutshell* by Alex Martelli. This book is a brief introduction and good reference to Python.

.NET

We recommend these books to learn more about .NET:

- *C# in a Nutshell* by Peter Drayton, Ben Albahari, and Ted Neward.
- *Pro C# with .NET 3.0* by Andrew Troelsen.
- *C# Essentials* by Ben Albahari, Peter Drayton, and Brad Merrill.
- *C# 3.0 Cookbook* by Jay Hilyard and Stephen Teilhet.

Web Sites

Python

You can find more information on Python at: http://www.python.org. This is the official python programming language site.

.NET

For information on .NET, see http://msdn.microsoft.com

Index