Chemical Price Research

Release 0.1

Baptiste Saliou

Jan 16, 2024

USER MANUAL:

Date: Jan 16, 2024 **Version**: 0.1.0

This Python library enables the search for molecule prices from three distinct integrators: Molport, ChemSpace, and MCule, using their corresponding SMILES (Simplified Molecular Input Line Entry System) notation.

This guide explains the functions of this library.

CHAPTER

ONE

INSTALLATION

1.1 Installing an official release

ChemPrice can be installed via PyPi.

1.1.1 Use pip

An alternative method is to install is using pip:

```
~$ pip install chemprice
```

1.2 Verify Installation

You can verify that ChemPrice was installed on your local computer by running:

```
~$ pip show chemprice
Name: chemprice
...
```

If instead of what is shown above your output is:

```
WARNING: Package(s) not found: chemprice
```

ChemPrice was not installed correctly or your system cannot find the path to it. If ChemPrice is installed correctly you can also test the package by running:

```
~$ pip install pytest
~$ pytest chemprice/tests/
```

CHAPTER

TWO

HOW TO USE CHEMPRICE

ChemPrice is a software designed to simplify the gathering of pricing data from more than 100 suppliers by integrating with ChemSpace, Mcule, and Molport platforms. It ensures uniformity in pricing units, providing comprehensive and optimized pricing details for specified compounds.

2.1 Getting started

First, let's create a list of molecules in SMILES notation to be searched:

```
smiles_list = ["CC(=O)NC1=CC=C(C=C1)O", "CC(C)CC1=CC=C(C=C1)C(C)C(=O)O",
↪"O=C(C)Oc1ccccc1C(=O)O"]
```

Next, create an instance from the PriceCollector class. Using this instance, we'll be able to connect to the various integrators and then launch a search on the list of SMILES entered.

```
from chemprice import PriceCollector

pc = PriceCollector()
```

2.2 Requesting an API key

To access integrators' data, we need to be able to connect to their API.

If you don't have an API key yet, you can request one via the following links : Molport, ChemSpace and MCule.

2.3 Enter the API key for each integrator

Now that the `PriceCollector` class has been created, we need to connect to one or more integrators via an API key.

Connection to Molport via API key (the following credentials are examples from Molport API documentations, and users need to obtain their personal credentials from Molport platform):

```
pc.setMolportApiKey("880d8343-8ui2-418c-9g7a-68b4e2e78c8b")
```

In the case of Molport, it's also possible to log in with a login and password. ChemSpace and Mcule support only API keys.

```
pc.setMolportUsername("john.spade")
pc.setMolportPassword("fasdga34a3")
```

To check the status of each key, run the :

```
pc.status()
```

Possible Outputs

```
# Username/Password and API Key are Set:
Status: Molport: both credentials are set.

# Only Username/Password or API Key is Set:
Status: Molport: credential is set.

# No Credential is Set:
Status: Molport: no credential is set.
```

In these examples, we're only talking about the Molport connection; for ChemSpace and MCule, the approach is the same. You need to use the `setChemSpaceApiKey()` and `setMCuleApiKey()` functions, such as :

```
pc.setChemSpaceApiKey(<chemspace_api_key>)
pc.setMCuleApiKey(<mcule_api_key>)
```

2.4 Price Search

Before starting the price search, check the validity of the api keys entered.

```
pc.check()
```

Possible Outputs:

```
# API Key is Set and correct:
Check: Molport api key is correct.

# API Key is Set but not correct:
Check: Molport api key is incorrect.
```

If the credentials checked are correct, then it's possible to run the method `collect()` to obtain the price information found on the molecule. The prices are given in USD according to the units and quantity entered by the vendor. The units of measurement for quantities are categorized into three families: moles, grams, and liters.

```
all_prices = pc.collect(smiles_list)
```

The output will be a dataframe containing all price information of the molecules in the search list.

Input Smiles	Source	Supplier Name	Purity	Amount	Measure	Price_USD
CC(=O)NC1=CC=C(C=C1)O	Molport	"ChemDiv, Inc."	>90	100	mg	407.1
CC(=O)NC1=CC=C(C=C1)O	Molport	MedChemExpress Europe	98.83	10	g	112.8
CC(=O)NC1=CC=C(C=C1)O	Molport	TargetMol Chemicals	100.0	500	mg	50.0

With the `selectBest()` function, we can select the best prices for each molecule. In fact, for each unit of measurement (mol, gram, and liter) the results are calculated separately to find the best quantity/price ratio.

```
pc.selectBest(all_prices)
```

The output will be a dataframe containing only the best quantity/price ratio for each molecule.

Input Smiles	Source	Supplier Name	Purity	Amount	Measure	Price_USD	USD/g	USD/mol
CC(=O)NC1=CC=C(C=C1)O	Molport	Cayman Europe	>=98	500	g	407.1	0.22	
O=C(C)Oc1ccccc1C(=O)O	Molport	Cayman Europe	>=90	500	g	112.8	0.1606	
O=C(C)Oc1ccccc1C(=O)O	Molport	Life Chemicals Inc.	>90	20	micromol	50.0		3950000.0000000005

CHAPTER

THREE

DEVELOPMENT ENVIRONMENT

The development environment is an installation of ChemPrice on your local computer which can be used for testing existing features or developing new ones in order to contribute to the library.

Then clone your forked GitHub repository of ChemPrice on your local computer using either HTTPS:

```
~$ git clone https://github.com/<your-username>/ChemPrice.git
```

Or using SSH:

```
~$ git clone git@github.com:<your-username>/ChemPrice.git
```

Then from the terminal navigate to the ChemPrice repository you just created. You can now install ChemPrices in editable mode. Editable mode will allow your code changes to be propagated through the library code without having to reinstall.

```
~/<PATH-TO-CLONE>/ChemPrice$ pip install -e .
```

You are now ready to develop ChemPrice!

3.1 Testing

To run the unit tests for ChemPrice use this command:

```
~$ pytest chemprice/tests/
```

CHAPTER

FOUR

API DOCUMENTATION

ChemPrice principal class is `PriceCollector`.

4.1 chemprice.PriceCollector

class chemprice.**PriceCollector**

A class for collecting and checking API credentials, as well as data collection and filtering.

setMolportUsername(*username*)

Sets the Molport username.

Parameters
username – The Molport username to set.

setMolportPassword(*password*)

Sets the Molport password.

Parameters
password – The Molport password to set.

setMolportApiKey(*api_key*)

Sets the Molport api key.

Parameters
api_key – The Molport api key to set.

setChemSpaceApiKey(*api_key*)

Sets the ChemSpace api key.

Parameters
api_key – The ChemSpace api key to set.

setMCuleApiKey(*api_key*)

Sets the MCule api key.

Parameters
api_key – The MCule api key to set.

status()

Displays the set status of various API keys.

check(*Molport=True*, *ChemSpace=True*, *MCule=True*)

Checks the validity of API credentials.

Parameters

- **Molport** – Whether to check Molport credentials.
- **ChemSpace** – Whether to check ChemSpace credentials.
- **MCule** – Whether to check MCule credentials.

Returns
A value indicating whether the checks were successful.

collect(*smiles_list*, *progress_output=None*)

Collects data using API credentials.

Parameters

- **smiles_list** – List of SMILES representations.
- **progress_output** – Progress output (optional).

Returns
A DataFrame containing collected data.

selectBest(*dataframe*)

Filters and selects the best data from a DataFrame.

Parameters
dataframe – The input DataFrame.

Returns
The filtered DataFrame.

4.2 Utils

utils.**molport_collect_prices**(*instance*, *molecule_ids*)

Collects price data for molecules from Molport API.

Parameters

- **instance** (PriceCollector) – The PriceCollector instance containing API credentials.
- **molecule_ids** (*pandas.DataFrame*) – DataFrame containing molecule IDs and SMILES.

Returns
DataFrame containing collected price data.

Return type
pandas.DataFrame

utils.**chemspace_collect_prices**(*instance*, *smiles_list*)

Collects price data for molecules from ChemSpace API.

Parameters

- **instance** (PriceCollector) – The PriceCollector instance containing API credentials.
- **smiles_list** (*list*) – list containing molecule SMILES.

Returns
DataFrame containing collected price data.

Return type
pandas.DataFrame

utils.**mcule_collect_prices**(*instance*, *package_ids*)

Collects price data for molecules from MCule API.

Parameters

- **instance** (PriceCollector) – The PriceCollector instance containing API credentials.
- **package_ids** (*list*) – list containing molecule package IDs.

Returns

DataFrame containing collected price data.

Return type

pandas.DataFrame

INDEX