Usage

Boolean variables in PySAT are represented as natural identifiers, e.g. numbers from $\mathbb{N}_{>0}$. A *literal* in PySAT is assumed to be an integer, e.g. `-1` represents a literal $\neg x_1$ while 5 represents a literal x_5. A *clause* is a list of literals, e.g. `[-3, -2]` is a clause $(\neg x_3 \vee \neg x_2)$.

The following is a trivial example of PySAT usage:

```
>>> from pysat.solvers import Glucose3
>>>
>>> g = Glucose3()
>>> g.add_clause([-1, 2])
>>> g.add_clause([-2, 3])
>>> print(g.solve())
>>> print(g.get_model())
...
True
[-1, -2, -3]
```

Another example shows how to extract *unsatisfiable cores* from a SAT solver given an unsatisfiable set of clauses:

```
>>> from pysat.solvers import Minisat22
>>>
>>> with Minisat22(bootstrap_with=[[-1, 2], [-2, 3]]) as m:
...     print(m.solve(assumptions=[1, -3]))
...     print(m.get_core())
...
False
[-3, 1]
```

Finally, the following example gives an idea of how one can extract a *proof* (supported by Glucose3, Glucose4, and Lingeling only):

```
>>> from pysat.formula import CNF
>>> from pysat.solvers import Lingeling
>>>
>>> formula = CNF()
>>> formula.append([-1, 2])
>>> formula.append([1, -2])
>>> formula.append([-1, -2])
>>> formula.append([1, 2])
>>>
>>> with Lingeling(bootstrap_with=formula.clauses, with_proof=True) as l:
...     if l.solve() == False:
...         print(l.get_proof())
...
['2 0', '1 0', '0']
```

PySAT usage is detailed in the provided examples. For instance, one can find simple PySAT-based implementations of

- Fu&Malik algorithm for MaxSAT [15]
- RC2/OLLITI algorithm for MaxSAT [19] [20]
- CLD-like algorithm for MCS extraction and enumeration [17]
- LBX-like algorithm for MCS extraction and enumeration [18]
- Deletion-based MUS extraction [16]

[15]Zhaohui Fu, Sharad Malik. *On Solving the Partial MAX-SAT Problem*. SAT 2006. pp. 252-265

[16]Joao Marques Silva. *Minimal Unsatisfiability: Models, Algorithms and Applications*. ISMVL 2010. pp. 9-14

[17]Joao Marques-Silva, Federico Heras, Mikolas Janota, Alessandro Previti, Anton Belov. *On Computing Minimal Correction Subsets*. IJCAI 2013. pp. 615-622

[18]Carlos Mencia, Alessandro Previti, Joao Marques-Silva. *Literal-Based MCS Extraction*. IJCAI 2015. pp. 1973-1979

[19]António Morgado, Carmine Dodaro, Joao Marques-Silva. *Core-Guided MaxSAT with Soft Cardinality Constraints*. CP 2014. pp. 564-573

[20]António Morgado, Alexey Ignatiev, Joao Marques-Silva. *MSCG: Robust Core-Guided MaxSAT Solving. System Description*. JSAT 2015. vol. 9, pp. 129-134

The examples are installed with PySAT as a subpackage and, thus, they can be accessed internally in Python:

```
>>> from pysat.formula import CNF
>>> from pysat.examples.lbx import LBX
>>>
>>> formula = CNF(from_file='input.cnf')
>>> mcsls = LBX(formula.weighted())
>>>
>>> for mcs in mcsls.enumerate():
...     print(mcs)
```

Alternatively, they can be used as standalone executables, e.g. like this:

```
$ lbx.py -e all -d -s g4 -v another-input.wcnf
```