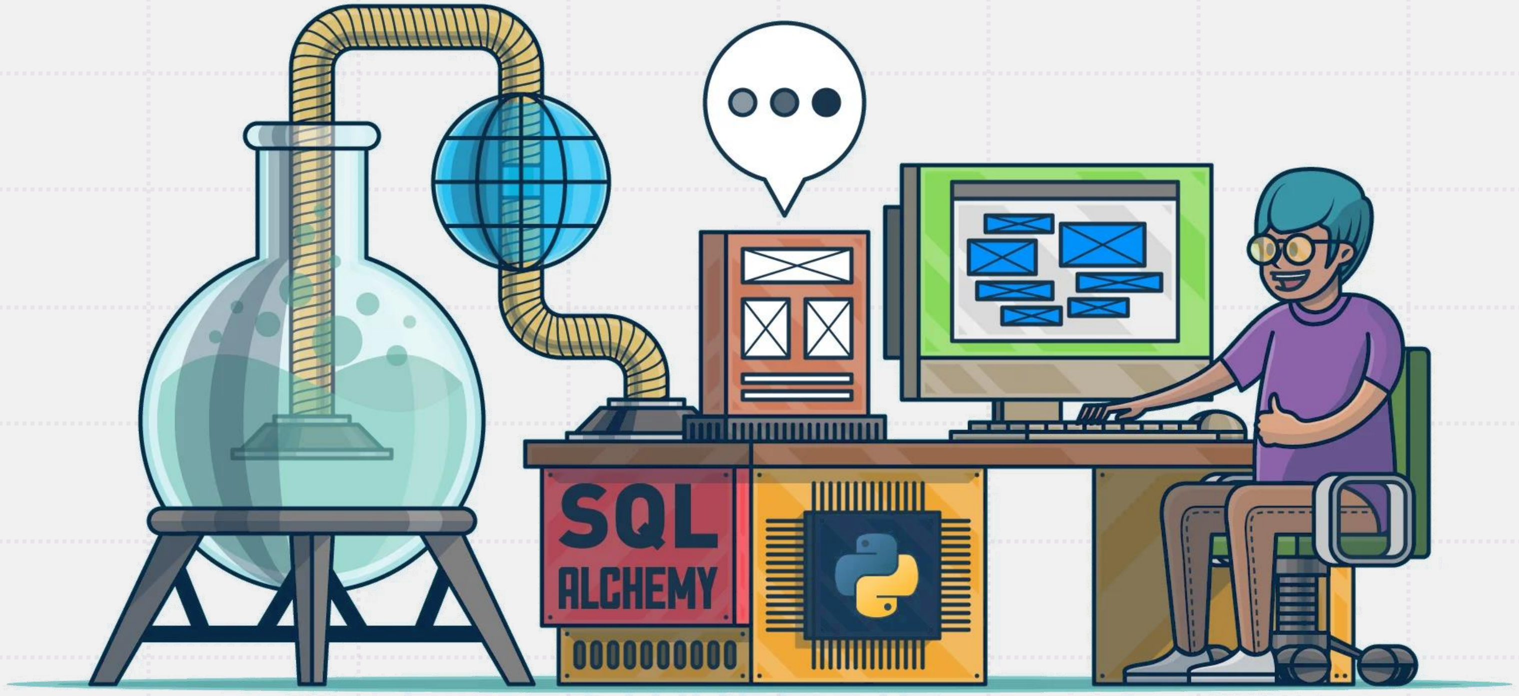
Programowanie baz danych

sqlalchemy 2



Real Python

Instrukcje DML

Insutrkcje DML

Instrukcje DML są w SQLAlchemy Expression Language reprezentowane przez klasy:

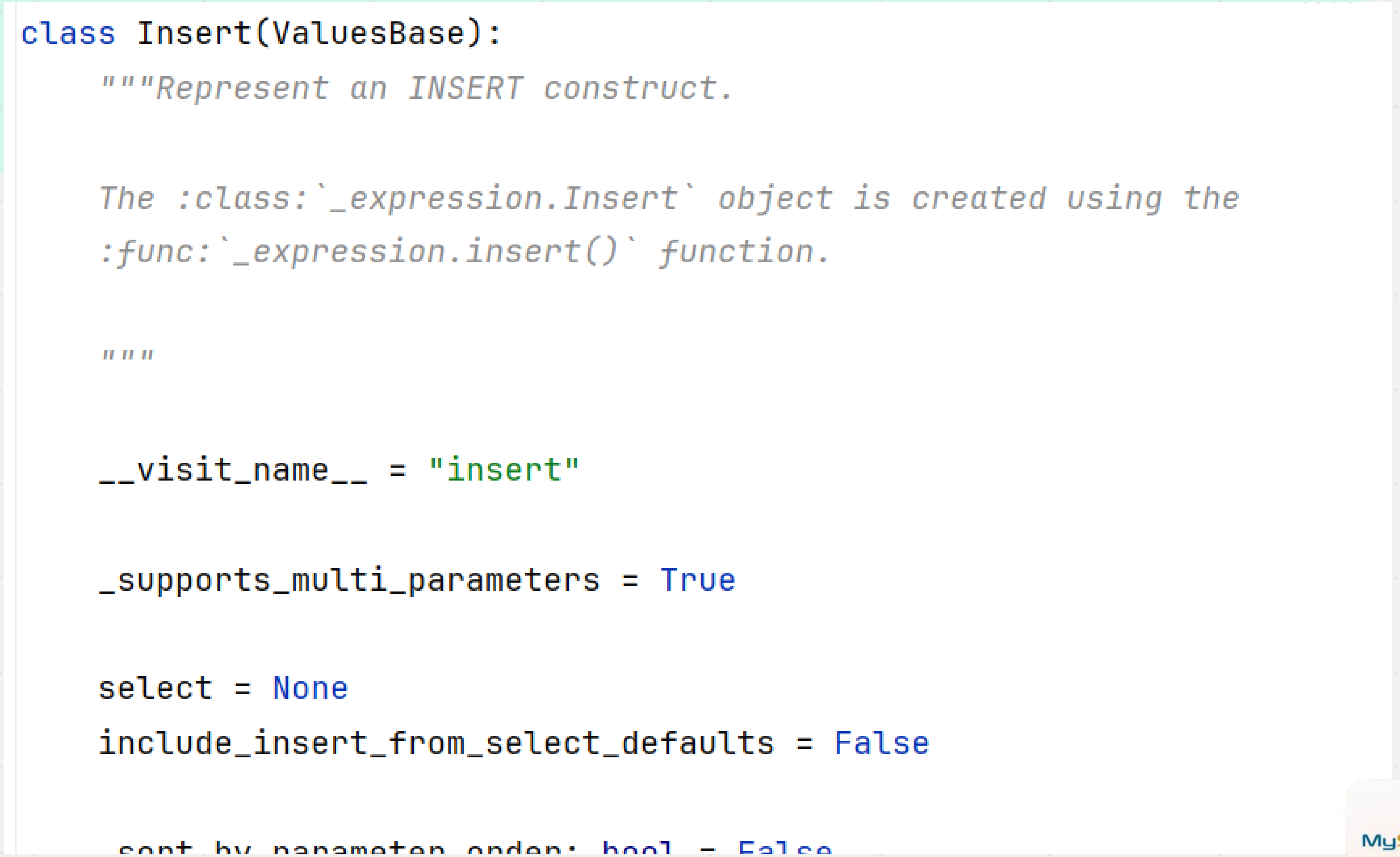
1. Insert

2. Update

3. Delete

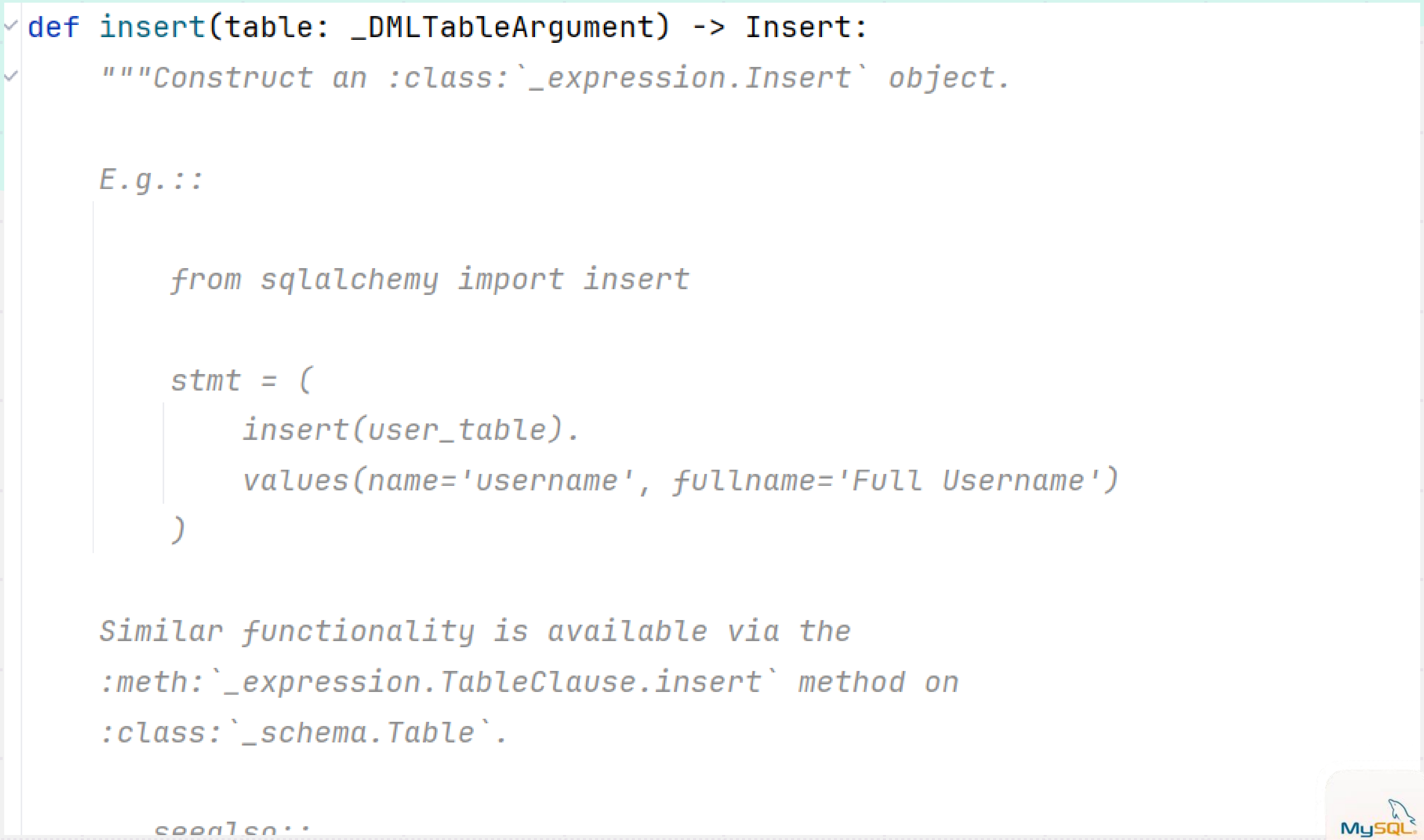
Wszystkie te klasy implementują tzw. fluent interface

1. Klasa Insert



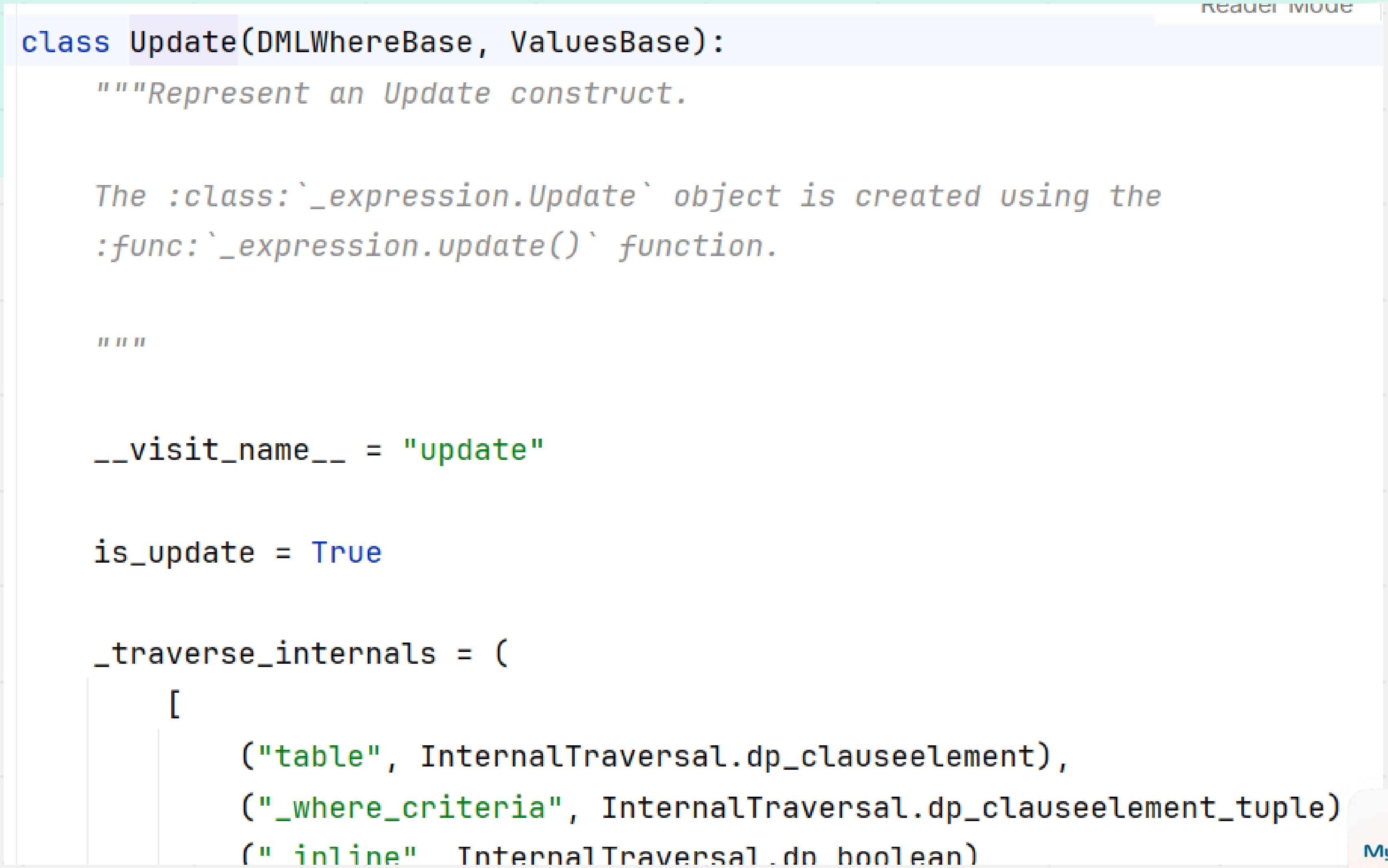
```
class Insert(ValuesBase):
    """Represent an INSERT construct.

    The :class:`_expression.Insert` object is created using the
    :func:`_expression.insert()` function.

    """

    __visit_name__ = "insert"

    _supports_multi_parameters = True

    select = None
    include_insert_from_select_defaults = False

    _sort_by_parameter_order: bool = False
```

*Preferowanym sposobem inicjalizacji obiektu klasy **Insert** jest funkcja **insert**.*



```
def insert(table: _DMLTableArgument) -> Insert:
    """Construct an :class:`_expression.Insert` object.

    E.g.::

        from sqlalchemy import insert

        stmt = (
            insert(user_table).
            values(name='username', fullname='Full Username')
        )

    Similar functionality is available via the
    :meth:`_expression.TableClause.insert` method on
    :class:`_schema.Table`.

      seealso::
```

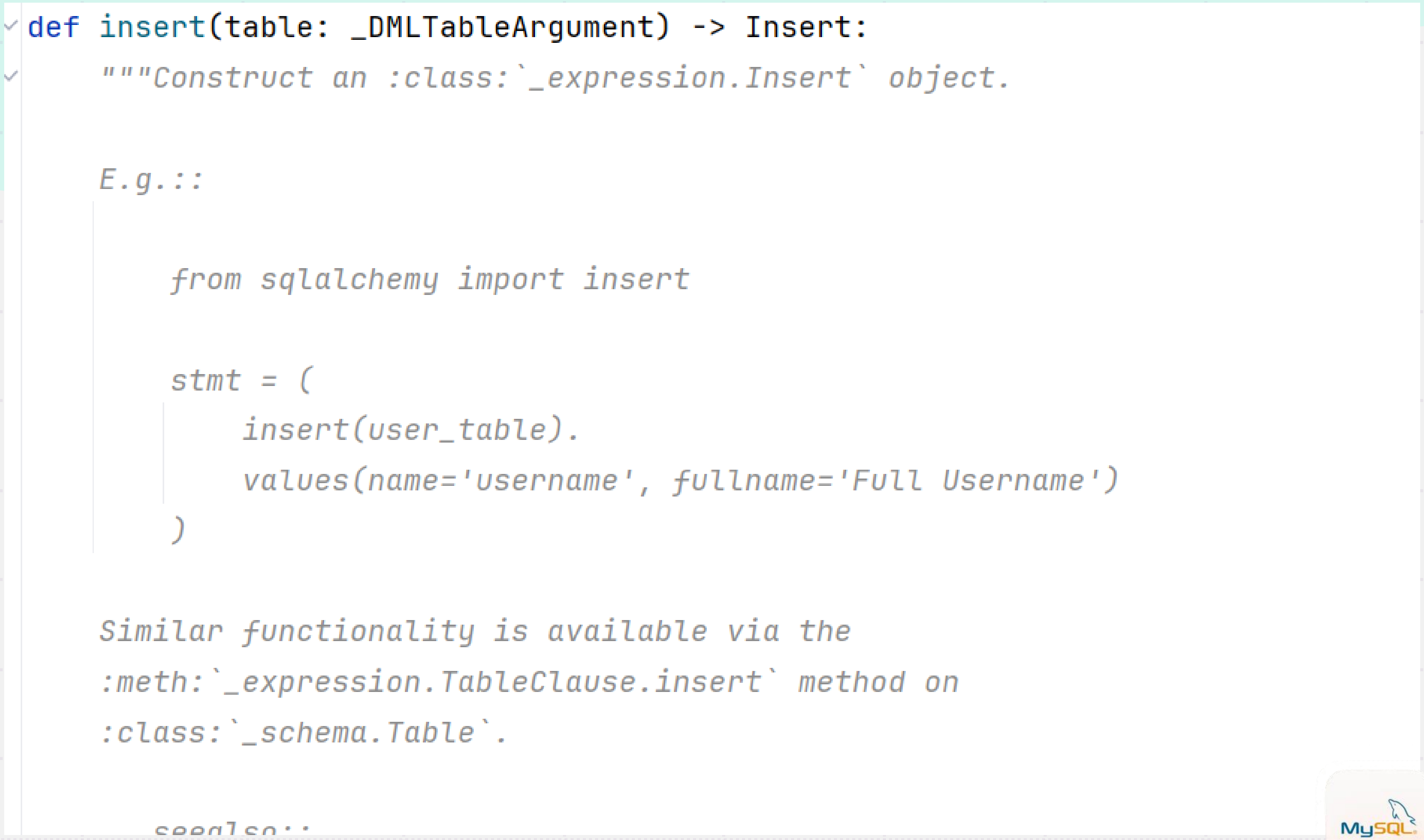
1. Klasa Update



Reader Mode

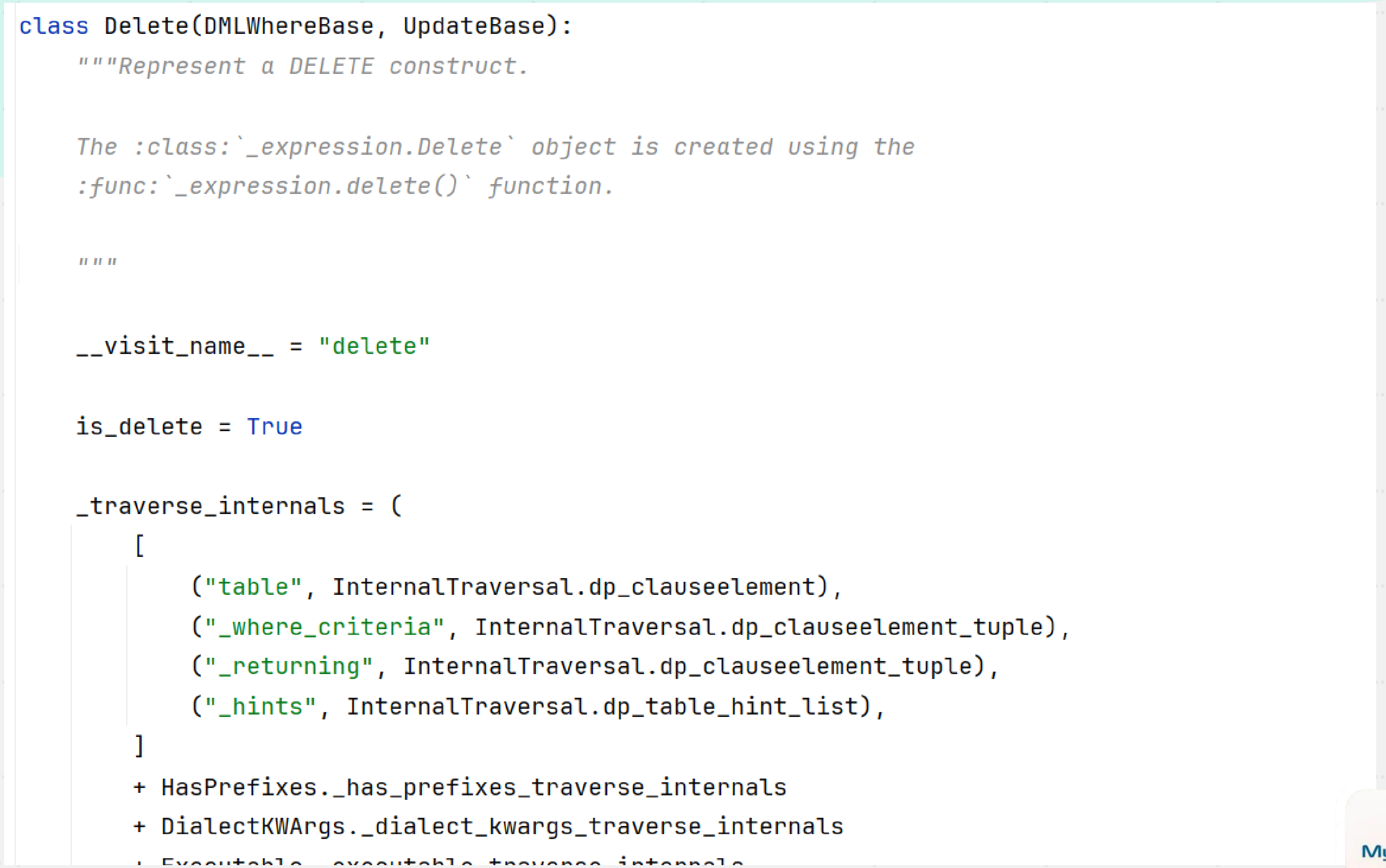
```
class Update(DMLWhereBase, ValuesBase):
    """Represent an Update construct.

    The :class:`_expression.Update` object is created using the
    :func:`_expression.update()` function.

    """

    __visit_name__ = "update"

    is_update = True

    _traverse_internals = (
        [
            ("table", InternalTraversal.dp_clauseelement),
            ("_where_criteria", InternalTraversal.dp_clauseelement_tuple),
            ("_inline", InternalTraversal.dp_boolean),
```

Preferowanym sposobem inicjalizacji obiektu klasy ***Update*** *jest funkcja* ***update****.*



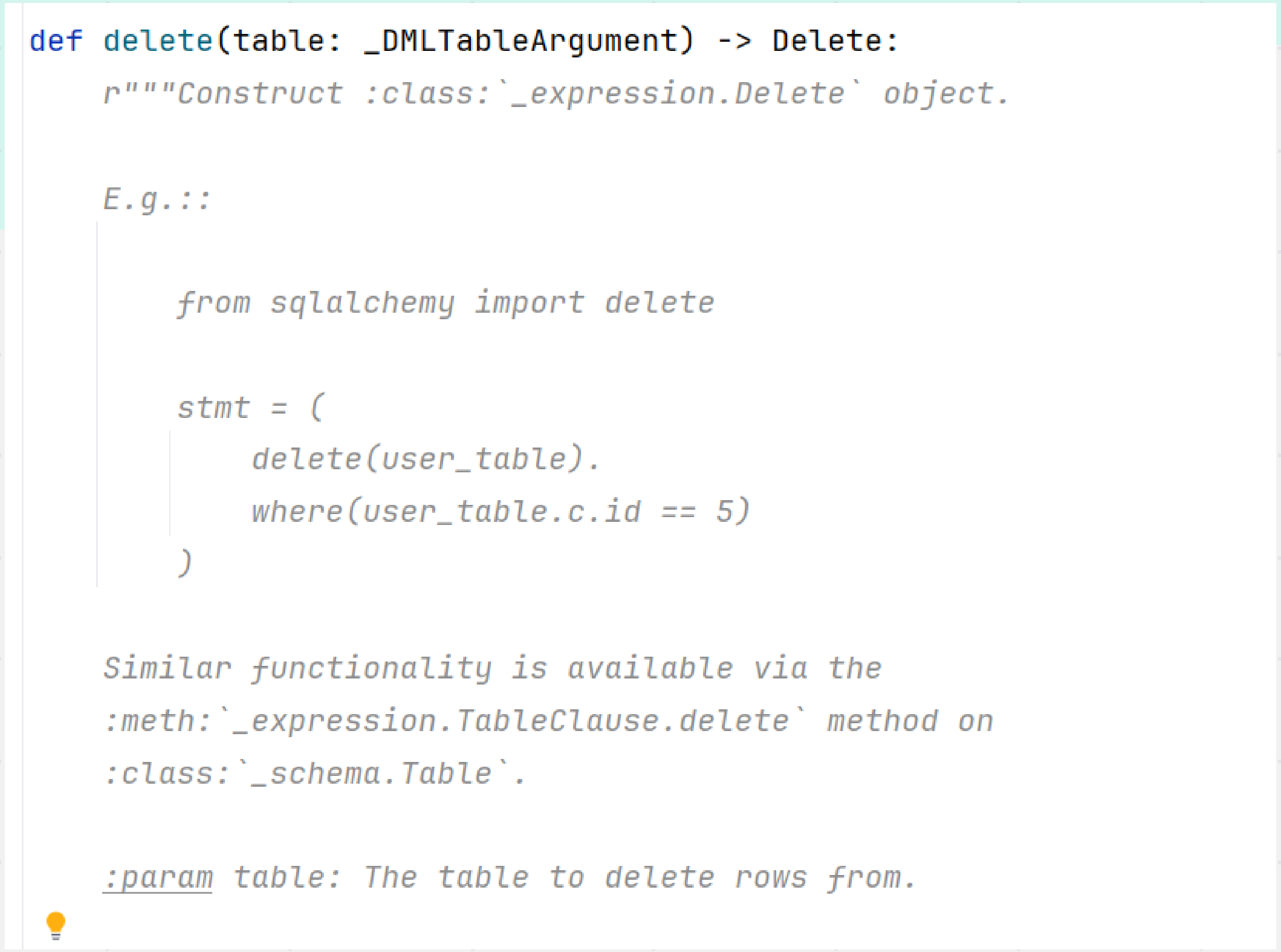
```
def insert(table: _DMLTableArgument) -> Insert:
    """Construct an :class:`_expression.Insert` object.

    E.g.::

        from sqlalchemy import insert

        stmt = (
            insert(user_table).
            values(name='username', fullname='Full Username')
        )

    Similar functionality is available via the
    :meth:`_expression.TableClause.insert` method on
    :class:`_schema.Table`.

      seealso::
```

3. Klasa Delete



```
class Delete(DMLWhereBase, UpdateBase):
    """Represent a DELETE construct.

    The :class:`_expression.Delete` object is created using the
    :func:`_expression.delete()` function.

    """

    __visit_name__ = "delete"

    is_delete = True

    _traverse_internals = (
        [
            ("table", InternalTraversal.dp_clauseelement),
            ("_where_criteria", InternalTraversal.dp_clauseelement_tuple),
            ("_returning", InternalTraversal.dp_clauseelement_tuple),
            ("_hints", InternalTraversal.dp_table_hint_list),
        ]
        + HasPrefixes._has_prefixes_traverse_internals
        + DialectKWArgs._dialect_kwargs_traverse_internals
        + Executable._executable_traverse_internals
```

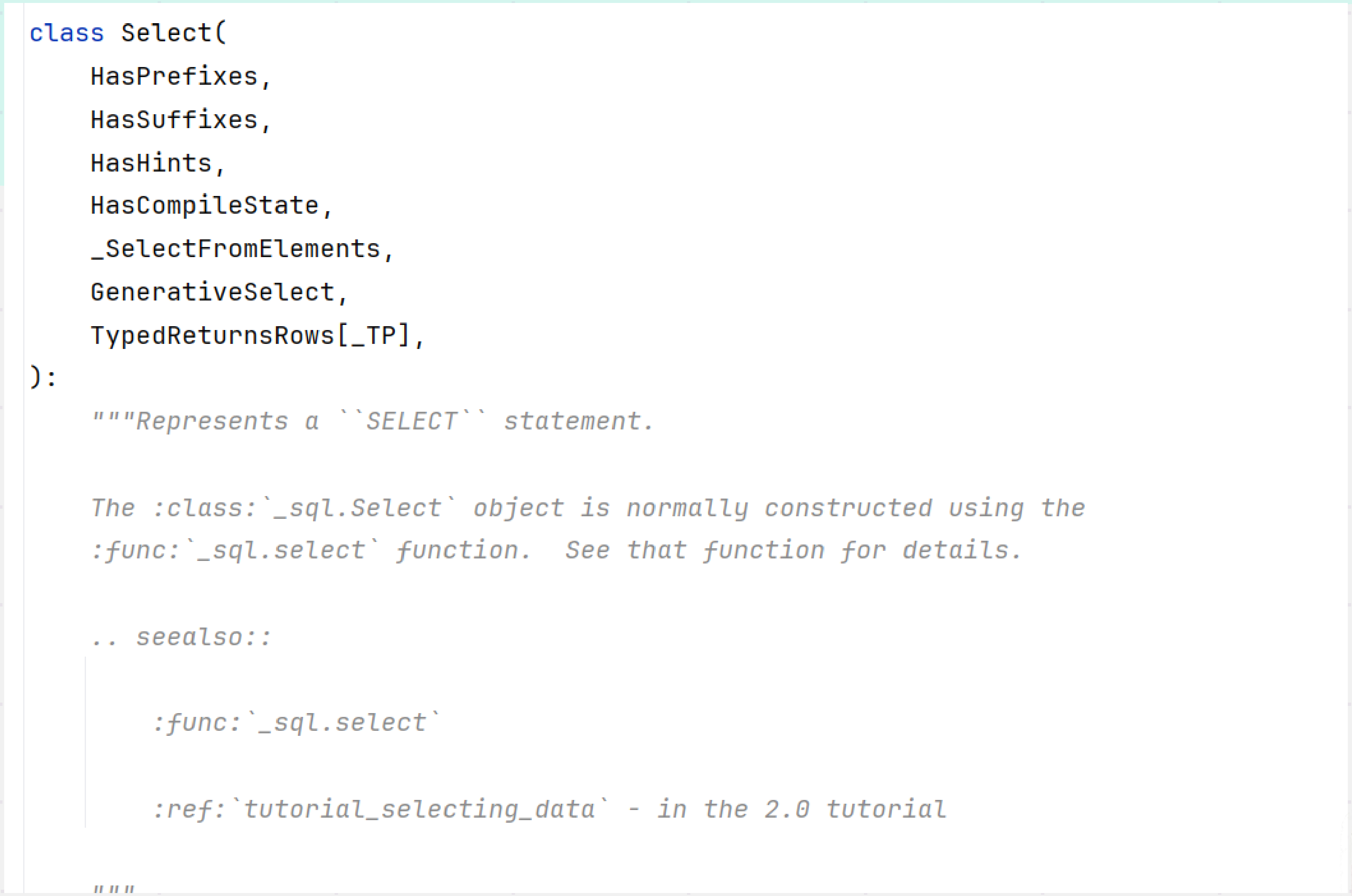
Preferowanym sposobem inicjalizacji obiektu klasy ***Delete*** *jest funkcja* ***delete****.*



```
def delete(table: _DMLTableArgument) -> Delete:
    r"""Construct :class:`_expression.Delete` object.

    E.g.::

        from sqlalchemy import delete

        stmt = (
            delete(user_table).
            where(user_table.c.id == 5)
        )

    Similar functionality is available via the
    :meth:`_expression.TableClause.delete` method on
    :class:`_schema.Table`.

    :param table: The table to delete rows from.
```

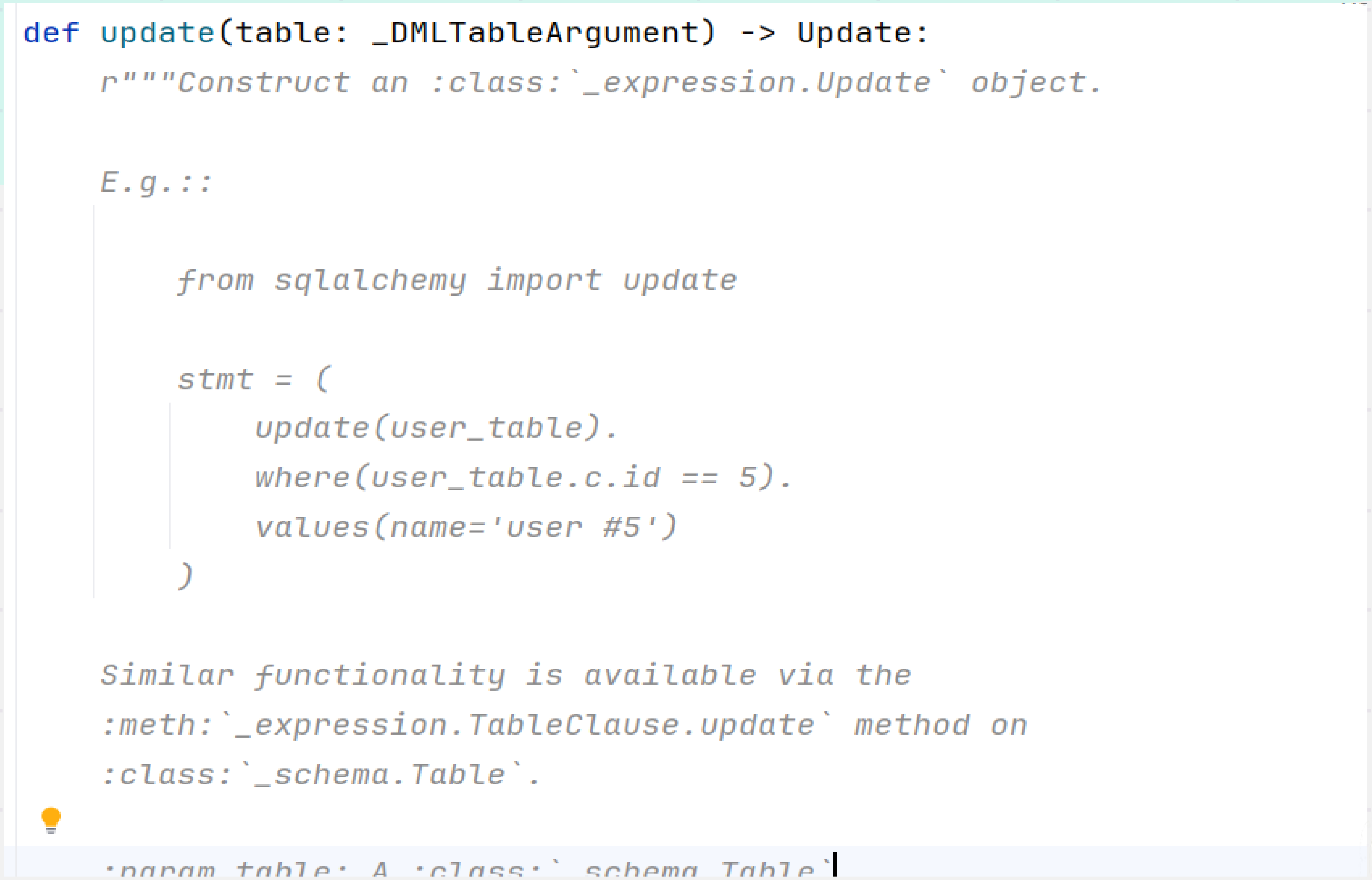
Instrukcje DQL

Klasa Select



```
class Select(
    HasPrefixes,
    HasSuffixes,
    HasHints,
    HasCompileState,
    _SelectFromElements,
    GenerativeSelect,
    TypedReturnsRows[_TP],
):
    """Represents a ``SELECT`` statement.

    The :class:`_sql.Select` object is normally constructed using the
    :func:`_sql.select` function.  See that function for details.

    .. seealso::

        :func:`_sql.select`

        :ref:`tutorial_selecting_data` - in the 2.0 tutorial

    """
```

*Preferowanym sposobem inicjalizacji obiektu klasy **Select** jest funkcja **select**.*



```
def update(table: _DMLTableArgument) -> Update:
    r"""Construct an :class:`_expression.Update` object.

    E.g.::

        from sqlalchemy import update

        stmt = (
            update(user_table).
            where(user_table.c.id == 5).
            values(name='user #5')
        )

    Similar functionality is available via the
    :meth:`_expression.TableClause.update` method on
    :class:`_schema.Table`.

    :param table: A :class:`_schema.Table`
```

ORM

MySQL + SQLAlchemy

SQLAlchemy object-relational mapping

SQLAlchemy ORM jest zbudowanym na SQLAlchemy Core modułem dostarczającym możliwość mapowania obiektów na relacje (czyli klas na tabele w bazie danych, a obiektów tych klas na rekordy w tabelkach)

Mapper

Session