psycopg2 (version 2.6.1 (dt dec mx pq3 ext lo64))

index
/usr/lib/python2.7/dist-packages/psycopg2/__init__.py

```
A Python driver for PostgreSQL

psycopg is a PostgreSQL_ database adapter for the Python_ programming
language. This is version 2, a complete rewrite of the original code to
provide new-style classes for connection and cursor objects and other sweet
candies. Like the original, psycopg 2 was written with the aim of being very
small and fast, and stable as a rock.

Homepage: http://initd.org/projects/psycopg2

.. _PostgreSQL: http://www.postgresql.org/
.. _Python: http://www.python.org/

:Groups:
  * `Connections creation`: connect
  * `Value objects constructors`: Binary, Date, DateFromTicks, Time,
    TimeFromTicks, Timestamp, TimestampFromTicks
```

Package Contents

_json	errorcodes	pool	tz
_psycopg	extensions	psycopg1	
_range	extras	**tests** (package)	

Classes

exceptions.StandardError(exceptions.Exception)
- Error
  - DatabaseError
    - DataError
    - IntegrityError
    - InternalError
    - NotSupportedError
    - OperationalError
    - ProgrammingError
  - InterfaceError
- Warning

class **DataError**(DatabaseError)

Error related to problems with the processed data.

Method resolution order:
- DataError
- DatabaseError
- Error
- exceptions.StandardError
- exceptions.Exception
- exceptions.BaseException
- __builtin__.object

Data descriptors inherited from DatabaseError:

__weakref__
list of weak references to the object (if defined)

Methods inherited from Error:

__init__(...)
x.__init__(...) initializes x; see help(type(x)) for signature

__reduce__(...)

__setstate__(...)

Data descriptors inherited from Error:

cursor
The cursor that raised the exception, if available, else None

diag
A Diagnostics object to get further information about the error

pgcode
The error code returned by the backend, if available, else None

pgerror
The error message returned by the backend, if available, else None

---

Data and other attributes inherited from Error:

__new__ = <built-in method __new__ of type object>
T.__new__(S, ...) -> a new object with type S, a subtype of T

Methods inherited from exceptions.BaseException:

__delattr__(...)
x.__delattr__('name') <==> del x.name

__getattribute__(...)
x.__getattribute__('name') <==> x.name

__getitem__(...)
x.__getitem__(y) <==> x[y]

__getslice__(...)
x.__getslice__(i, j) <==> x[i:j]

Use of negative indices is not supported.

__repr__(...)
x.__repr__() <==> repr(x)

__setattr__(...)
x.__setattr__('name', value) <==> x.name = value

__str__(...)
x.__str__() <==> str(x)

__unicode__(...)

Data descriptors inherited from exceptions.BaseException:

__dict__

args

message

class **DatabaseError**(Error)
Error related to the database engine.

Method resolution order:
DatabaseError
Error
exceptions.StandardError
exceptions.Exception
exceptions.BaseException
__builtin__.object

Data descriptors defined here:

__weakref__
list of weak references to the object (if defined)

Methods inherited from Error:

__init__(...)
x.__init__(...) initializes x; see help(type(x)) for signature

__reduce__(...)

__setstate__(...)

Data descriptors inherited from Error:

cursor
The cursor that raised the exception, if available, else None

diag
A Diagnostics object to get further information about the error

pgcode
The error code returned by the backend, if available, else None

pgerror
The error message returned by the backend, if available, else None

---

Data and other attributes inherited from Error:

__new__ = <built-in method __new__ of type object>
T.__new__(S, ...) -> a new object with type S, a subtype of T

Methods inherited from exceptions.BaseException:

__delattr__(...)
x.__delattr__('name') <==> del x.name

__getattribute__(...)
x.__getattribute__('name') <==> x.name

__getitem__(...)
x.__getitem__(y) <==> x[y]

__getslice__(...)
x.__getslice__(i, j) <==> x[i:j]

Use of negative indices is not supported.

__repr__(...)
x.__repr__() <==> repr(x)

__setattr__(...)
x.__setattr__('name', value) <==> x.name = value

__str__(...)
x.__str__() <==> str(x)

__unicode__(...)

Data descriptors inherited from exceptions.BaseException:

__dict__

args

message

class **Error**(exceptions.StandardError)
Base class for error exceptions.

Method resolution order:
Error
exceptions.StandardError
exceptions.Exception
exceptions.BaseException
__builtin__.object

Methods defined here:

__init__(...)
x.__init__(...) initializes x; see help(type(x)) for signature

__reduce__(...)

__setstate__(...)

Data descriptors defined here:

cursor
The cursor that raised the exception, if available, else None

diag
A Diagnostics object to get further information about the error

pgcode
The error code returned by the backend, if available, else None

pgerror
The error message returned by the backend, if available, else None

---

Data and other attributes defined here:

__new__ = <built-in method __new__ of type object>
T.__new__(S, ...) -> a new object with type S, a subtype of T

Methods inherited from exceptions.BaseException:

__delattr__(...)
x.__delattr__('name') <==> del x.name

__getattribute__(...)
x.__getattribute__('name') <==> x.name

__getitem__(...)
x.__getitem__(y) <==> x[y]

__getslice__(...)
x.__getslice__(i, j) <==> x[i:j]

Use of negative indices is not supported.

__repr__(...)
x.__repr__() <==> repr(x)

__setattr__(...)
x.__setattr__('name', value) <==> x.name = value

__str__(...)
x.__str__() <==> str(x)

__unicode__(...)

Data descriptors inherited from exceptions.BaseException:

__dict__

args

message

class **IntegrityError**(DatabaseError)
Error related to database integrity.

Method resolution order:
IntegrityError
DatabaseError
Error
exceptions.StandardError
exceptions.Exception
exceptions.BaseException
__builtin__.object

Data descriptors inherited from DatabaseError:

__weakref__
list of weak references to the object (if defined)

Methods inherited from Error:

__init__(...)
x.__init__(...) initializes x; see help(type(x)) for signature

__reduce__(...)

__setstate__(...)

Data descriptors inherited from Error:

cursor
The cursor that raised the exception, if available, else None

diag
A Diagnostics object to get further information about the error

pgcode
The error code returned by the backend, if available, else None

pgerror
The error message returned by the backend, if available, else None

---

Data and other attributes inherited from Error:

__new__ = <built-in method __new__ of type object>
T.__new__(S, ...) -> a new object with type S, a subtype of T

Methods inherited from exceptions.BaseException:

__delattr__(...)
x.__delattr__('name') <==> del x.name

__getattribute__(...)
x.__getattribute__('name') <==> x.name

__getitem__(...)
x.__getitem__(y) <==> x[y]

__getslice__(...)
x.__getslice__(i, j) <==> x[i:j]

Use of negative indices is not supported.

__repr__(...)
x.__repr__() <==> repr(x)

__setattr__(...)
x.__setattr__('name', value) <==> x.name = value

__str__(...)
x.__str__() <==> str(x)

__unicode__(...)

Data descriptors inherited from exceptions.BaseException:

__dict__

args

message

class **InterfaceError**(Error)
Error related to the database interface.

Method resolution order:
InterfaceError
Error
exceptions.StandardError
exceptions.Exception
exceptions.BaseException
__builtin__.object

Data descriptors defined here:

__weakref__
list of weak references to the object (if defined)

Methods inherited from Error:

__init__(...)
x.__init__(...) initializes x; see help(type(x)) for signature

__reduce__(...)

__setstate__(...)

Data descriptors inherited from Error:

cursor
The cursor that raised the exception, if available, else None

diag
A Diagnostics object to get further information about the error

pgcode
The error code returned by the backend, if available, else None

pgerror
The error message returned by the backend, if available, else None

---

Data and other attributes inherited from Error:

__new__ = <built-in method __new__ of type object>
T.__new__(S, ...) -> a new object with type S, a subtype of T

Methods inherited from exceptions.BaseException:

__delattr__(...)
x.__delattr__('name') <==> del x.name

__getattribute__(...)
x.__getattribute__('name') <==> x.name

__getitem__(...)
x.__getitem__(y) <==> x[y]

__getslice__(...)
x.__getslice__(i, j) <==> x[i:j]

Use of negative indices is not supported.

__repr__(...)
x.__repr__() <==> repr(x)

__setattr__(...)
x.__setattr__('name', value) <==> x.name = value

__str__(...)
x.__str__() <==> str(x)

__unicode__(...)

Data descriptors inherited from exceptions.BaseException:

__dict__

args

message

class **InternalError**(DatabaseError)
The database encountered an internal error.

Method resolution order:
InternalError
DatabaseError
Error
exceptions.StandardError
exceptions.Exception
exceptions.BaseException
__builtin__.object

Data descriptors inherited from DatabaseError:

__weakref__
list of weak references to the object (if defined)

Methods inherited from Error:

__init__(...)
x.__init__(...) initializes x; see help(type(x)) for signature

__reduce__(...)

__setstate__(...)

Data descriptors inherited from Error:

cursor
The cursor that raised the exception, if available, else None

diag
A Diagnostics object to get further information about the error

pgcode
The error code returned by the backend, if available, else None

pgerror
The error message returned by the backend, if available, else None

---

Data and other attributes inherited from Error:

__new__ = <built-in method __new__ of type object>
T.__new__(S, ...) -> a new object with type S, a subtype of T

Methods inherited from exceptions.BaseException:

__delattr__(...)
x.__delattr__('name') <==> del x.name

__getattribute__(...)
x.__getattribute__('name') <==> x.name

__getitem__(...)
x.__getitem__(y) <==> x[y]

__getslice__(...)
x.__getslice__(i, j) <==> x[i:j]

Use of negative indices is not supported.

__repr__(...)
x.__repr__() <==> repr(x)

__setattr__(...)
x.__setattr__('name', value) <==> x.name = value

__str__(...)
x.__str__() <==> str(x)

__unicode__(...)

Data descriptors inherited from exceptions.BaseException:

__dict__

args

message

class **NotSupportedError**(DatabaseError)
A method or database API was used which is not supported by the database.

Method resolution order:
NotSupportedError
DatabaseError
Error
exceptions.StandardError
exceptions.Exception
exceptions.BaseException
__builtin__.object

Data descriptors inherited from DatabaseError:

__weakref__
list of weak references to the object (if defined)

Methods inherited from Error:

__init__(...)
x.__init__(...) initializes x; see help(type(x)) for signature

__reduce__(...)

__setstate__(...)

Data descriptors inherited from Error:

cursor
The cursor that raised the exception, if available, else None

diag
A Diagnostics object to get further information about the error

pgcode
The error code returned by the backend, if available, else None

pgerror
The error message returned by the backend, if available, else None

---

Data and other attributes inherited from Error:

__new__ = <built-in method __new__ of type object>
T.__new__(S, ...) -> a new object with type S, a subtype of T

Methods inherited from exceptions.BaseException:

__delattr__(...)
x.__delattr__('name') <==> del x.name

__getattribute__(...)
x.__getattribute__('name') <==> x.name

__getitem__(...)
x.__getitem__(y) <==> x[y]

__getslice__(...)
x.__getslice__(i, j) <==> x[i:j]

Use of negative indices is not supported.

__repr__(...)
x.__repr__() <==> repr(x)

__setattr__(...)
x.__setattr__('name', value) <==> x.name = value

__str__(...)
x.__str__() <==> str(x)

__unicode__(...)

Data descriptors inherited from exceptions.BaseException:

__dict__

args

message

class **OperationalError**(DatabaseError)

Error related to database operation (disconnect, memory allocation etc).

Method resolution order:
OperationalError
DatabaseError
Error
exceptions.StandardError
exceptions.Exception
exceptions.BaseException
__builtin__.object

Data descriptors inherited from DatabaseError:

__weakref__
list of weak references to the object (if defined)

Methods inherited from Error:

__init__(...)
x.__init__(...) initializes x; see help(type(x)) for signature

__reduce__(...)

__setstate__(...)

Data descriptors inherited from Error:

cursor
The cursor that raised the exception, if available, else None

diag
A Diagnostics object to get further information about the error

pgcode
The error code returned by the backend, if available, else None

pgerror
The error message returned by the backend, if available, else None

---

Data and other attributes inherited from Error:

__new__ = <built-in method __new__ of type object>
T.__new__(S, ...) -> a new object with type S, a subtype of T

Methods inherited from exceptions.BaseException:

__delattr__(...)
x.__delattr__('name') <==> del x.name

__getattribute__(...)
x.__getattribute__('name') <==> x.name

__getitem__(...)
x.__getitem__(y) <==> x[y]

__getslice__(...)
x.__getslice__(i, j) <==> x[i:j]

Use of negative indices is not supported.

__repr__(...)
x.__repr__() <==> repr(x)

__setattr__(...)
x.__setattr__('name', value) <==> x.name = value

__str__(...)
x.__str__() <==> str(x)

__unicode__(...)

Data descriptors inherited from exceptions.BaseException:

__dict__

args

message

class **ProgrammingError**(DatabaseError)

```
Error related to database programming (SQL error, table not found etc).
```

Method resolution order:

- ProgrammingError
- DatabaseError
- Error
- exceptions.StandardError
- exceptions.Exception
- exceptions.BaseException
- __builtin__.object

Data descriptors inherited from DatabaseError:

__weakref__

```
list of weak references to the object (if defined)
```

Methods inherited from Error:

__init__(...)

```
x.__init__(...) initializes x; see help(type(x)) for signature
```

__reduce__(...)

__setstate__(...)

Data descriptors inherited from Error:

cursor

```
The cursor that raised the exception, if available, else None
```

diag

```
A Diagnostics object to get further information about the error
```

pgcode

```
The error code returned by the backend, if available, else None
```

pgerror

```
The error message returned by the backend, if available, else None
```

Data and other attributes inherited from Error:

__new__ = <built-in method __new__ of type object>

```
T.__new__(S, ...) -> a new object with type S, a subtype of T
```

Methods inherited from exceptions.BaseException:

__delattr__(...)

```
x.__delattr__('name') <==> del x.name
```

__getattribute__(...)

```
x.__getattribute__('name') <==> x.name
```

__getitem__(...)

```
x.__getitem__(y) <==> x[y]
```

__getslice__(...)
x.__getslice__(i, j) <==> x[i:j]

Use of negative indices is not supported.

__repr__(...)
x.__repr__() <==> repr(x)

__setattr__(...)
x.__setattr__('name', value) <==> x.name = value

__str__(...)
x.__str__() <==> str(x)

__unicode__(...)

Data descriptors inherited from exceptions.BaseException:

__dict__

args

message

class **Warning**(exceptions.StandardError)
A database warning.

Method resolution order:
Warning
exceptions.StandardError
exceptions.Exception
exceptions.BaseException
__builtin__.object

Data descriptors defined here:

__weakref__
list of weak references to the object (if defined)

Methods inherited from exceptions.StandardError:

__init__(...)
x.__init__(...) initializes x; see help(type(x)) for signature

---

Data and other attributes inherited from exceptions.StandardError:

__new__ = <built-in method __new__ of type object>
T.__new__(S, ...) -> a new object with type S, a subtype of T

Methods inherited from exceptions.BaseException:

__delattr__(...)
x.__delattr__('name') <==> del x.name

__getattribute__(...)
x.__getattribute__('name') <==> x.name

__getitem__(...)
x.__getitem__(y) <==> x[y]

__getslice__(...)
x.__getslice__(i, j) <==> x[i:j]

Use of negative indices is not supported.

__reduce__(...)

__repr__(...)
x.__repr__() <==> repr(x)

__setattr__(...)
x.__setattr__('name', value) <==> x.name = value

__setstate__(...)

__str__(...)
x.__str__() <==> str(x)

__unicode__(...)

Data descriptors inherited from exceptions.BaseException:

__dict__

args

message

Functions

Date(...)
Date(year, month, day) -> new date

Build an object holding a date value.

DateFromTicks(...)
DateFromTicks(ticks) -> new date

Build an object holding a date value from the given ticks value.

Ticks are the number of seconds since the epoch; see the documentation of the standard Python time module for details).

Time(...)
Time(hour, minutes, seconds, tzinfo=None) -> new time

Build an object holding a time value.

TimeFromTicks(...)
TimeFromTicks(ticks) -> new time

Build an object holding a time value from the given ticks value.

Ticks are the number of seconds since the epoch; see the documentation of the standard Python time module for details).

Timestamp(...)
Timestamp(year, month, day, hour, minutes, seconds, tzinfo=None) -> new timestamp

Build an object holding a timestamp value.

TimestampFromTicks(...)
TimestampFromTicks(ticks) -> new timestamp

Build an object holding a timestamp value from the given ticks value.

Ticks are the number of seconds since the epoch; see the documentation of the standard Python time module for details).

connect(dsn=None, database=None, user=None, password=None, host=None, port=None, connection_factory=None, cursor_factory=None, async=False, **kwargs)
Create a new database connection.

The connection parameters can be specified either as a string:

    conn = psycopg2.connect("dbname=test user=postgres password=secret")

or using a set of keyword arguments:

    conn = psycopg2.connect(database="test", user="postgres", password="secret")

The basic connection parameters are:

- *dbname*: the database name (only in dsn string)

```
- *database*: the database name (only as keyword argument)
- *user*: user name used to authenticate
- *password*: password used to authenticate
- *host*: database host address (defaults to UNIX socket if not provided)
- *port*: connection port number (defaults to 5432 if not provided)

Using the *connection_factory* parameter a different class or connections
factory can be specified. It should be a callable object taking a dsn
argument.

Using the *cursor_factory* parameter, a new default cursor factory will be
used by cursor().

Using *async*=True an asynchronous connection will be created.

Any other keyword parameter will be passed to the underlying client
library: the list of supported parameters depends on the library version.
```

Data

BINARY = <psycopg2._psycopg.type 'BINARY'>
DATETIME = <psycopg2._psycopg.type 'DATETIME'>
NUMBER = <psycopg2._psycopg.type 'NUMBER'>
ROWID = <psycopg2._psycopg.type 'ROWID'>
STRING = <psycopg2._psycopg.type 'STRING'>
__version__ = '2.6.1 (dt dec mx pq3 ext lo64)'
apilevel = '2.0'
paramstyle = 'pyformat'
threadsafety = 2