# py-ulid

*Release 1.0.2*

**Feb 24, 2020**

# Contents:


The py-ulid library is a *minimal*, self-contained implementation of the ULID (Universally Unique Lexicographically Sortable Identifier) specification in Python. For more information, please refer to the official specification.

# CHAPTER 1

# Installation

You can install the py-ulid library from PyPi

```
pip install py-ulid
```

The py-ulid library can be used in any version of python >= 3.5 and does not require *any* additional packages or modules.

CHAPTER 2

# How to use

To generate a ULID, simple run the generate() function

```
from ulid import ULID

#Instantiate the ULID class
ulid = ULID()
ulid.generate()  #01BX5ZZKBKACTAV9WEVGEMMVRZ
```

## 2.1 Seeding Time

You can instantiate the instance of the ULID class with a seed time which will output the same string for the time component. This could be useful when migrating to ulid

```
from ulid import ULID

#Instantiate the ULID class
ulid = ULID(1469918176385)
ulid.generate()  #01ARYZ6S41TSV4RRFFQ69G5FAV
```

## 2.2 Monotonic ULIDs

```
from ulid import Monotonic

#Instantiate the Monotonic Class
ulid = Monotonic()

# Same timestamp when calls are made within the same
# millisecond and least-significant random bit is incremented by 1
ulid.generate()  #01DC8Y7RBV4RSXX0437Z1RQR11
```

(continues on next page)

(continued from previous page)

```
ulid.generate()  #01DC8Y7RBV4RSXX0437Z1RQR12
ulid.generate()  #01DC8Y7RBV4RSXX0437Z1RQR13
ulid.generate()  #01DC8Y7RBV4RSXX0437Z1RQR14
ulid.generate()  #01DC8Y7RBV4RSXX0437Z1RQR15
ulid.generate()  #01DC8Y7RBV4RSXX0437Z1RQR16
ulid.generate()  #01DC8Y7RBV4RSXX0437Z1RQR17
ulid.generate()  #01DC8Y7RBV4RSXX0437Z1RQR18
ulid.generate()  #01DC8Y7RBV4RSXX0437Z1RQR19
```

CHAPTER 3

# ulid package

## 3.1 ulid module

This module provides immutable ULID objects (class ULID) according to the ULID spec and the functions *generate()* to generate ulids according to the specifications, *encode()* to transform a given integer to the canonical string representation of an ULID, and *decode()* to take a canonically encoded string and break it down into it's timestamp and randomness components. The module also provides Monotonic sort order guarantee for ULIDs via the Monotonic class and it's associated generate() function.

**class** ulid.ulid.**Monotonic**(*seed=None*)

Bases: *ulid.ulid.ULID*

The Monotonic class represent an extension of the base ULID class with the addition of a monotonic sort order (correctly detects and handles the same millisecond)

**MAX_EPOCH_TIME = 281474976710655**

**crockford_base = '0123456789ABCDEFGHJKMNPQRSTVWXYZ'**

**decode**(*s: str*) → Tuple[int, int]

Given a properly formed ULID, return a tuple containing the timestamp and the randomness component as ints

**Parameters** **s** (*str*) – ULID string to decode

**Returns** (timestamp, randomness)

**Return type** tuple(int, int)

**Raises** TypeError, ValueError

```
>>> from ulid import ULID
>>> ulid = ULID()
>>> ulid.decode('01BX5ZZKBKACTAV9WEVGEMMVRY')
(1508808576371, 392928161897179156999966)
```

**encode**(*i: int*) → str

Convert a given integer into the canonical ULID string format

**Parameters** **i** (*int*) – The integer to convert

**Returns** Canonically encoded ULID string

**Return type** str

**Raises** TypeError, ValueError

```
>>> from ulid import ULID
>>> ulid = ULID()
>>> ulid.encode(340282366920938463463374607431768167)
00864KEJY6MZQSVCHD1SB08637
```

**encode_timestamp**(*t: int*) → str

Convert a given unix timestamp into the canonical ULID string format

**Parameters** **t** (*int*) – The unix timestamp to convert

**Returns** Canonically encoded ULID string

**Return type** str

**Raises** TypeError, ValueError

```
>>> from ulid import ULID
>>> ulid = ULID()
>>> ulid.encode(281474976710655)
7ZZZZZZZZZ
```

**generate**() → str

Generate a 26 character ULID string encoded in Crockford's Base32

**Returns** Canonically encoded ULID string

**Return type** str

**Raises** ValueError

```
>>> from ulid import Monotonic
>>> ulid = Monotonic()
>>> ulid.generate()
01BX5ZZKBKACTAV9WEVGEMMVRZ
```

**pretty_print**(*s: str*) → None

Print the given ULID string in a binary layout

```
>>> from ulid import ULID
>>> ulid = ULID()
>>> ulid.pretty_print(ulid_str)
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       32_bit_uint_time_high                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     16_bit_uint_time_low      |       16_bit_uint_random      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       32_bit_uint_random                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       32_bit_uint_random                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

**class** ulid.ulid.**ULID**(*seed=None*)

Bases: object

Instances of the ULID class represent ULIDS as specified in the ULID spec. ULIDS have 128-bit compatibility with UUID, are Lexicographically sortable, case insensitive and URL safe

**MAX_EPOCH_TIME = 281474976710655**

**crockford_base = '0123456789ABCDEFGHJKMNPQRSTVWXYZ'**

**decode**(*s: str*) → Tuple[int, int]

Given a properly formed ULID, return a tuple containing the timestamp and the randomness component as ints

**Parameters** **s** (*str*) – ULID string to decode

**Returns** (timestamp, randomness)

**Return type** tuple(int, int)

**Raises** TypeError, ValueError

```
>>> from ulid import ULID
>>> ulid = ULID()
>>> ulid.decode('01BX5ZZKBKACTAV9WEVGEMMVRY')
(1508808576371, 3929281618971791569999966)
```

**encode**(*i: int*) → str

Convert a given integer into the canonical ULID string format

**Parameters** **i** (*int*) – The integer to convert

**Returns** Canonically encoded ULID string

**Return type** str

**Raises** TypeError, ValueError

```
>>> from ulid import ULID
>>> ulid = ULID()
>>> ulid.encode(340282366920938463463374607431768167)
00864KEJY6MZQSVCHD1SB08637
```

**encode_timestamp**(*t: int*) → str

Convert a given unix timestamp into the canonical ULID string format

**Parameters** **t** (*int*) – The unix timestamp to convert

**Returns** Canonically encoded ULID string

**Return type** str

**Raises** TypeError, ValueError

```
>>> from ulid import ULID
>>> ulid = ULID()
>>> ulid.encode(281474976710655)
7ZZZZZZZZZ
```

**generate**() → str

Generate a 26 character ULID string encoded in Crockford's Base32

**Returns** Canonically encoded ULID string

**Return type** str

```
>>> from ulid import ULID
>>> ulid = ULID()
>>> ulid.generate()
01BX5ZZKBKACTAV9WEVGEMMVRZ
```

**pretty_print**(*s: str*) → None

Print the given ULID string in a binary layout

```
>>> from ulid import ULID
>>> ulid = ULID()
>>> ulid.pretty_print(ulid_str)
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      32_bit_uint_time_high                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     16_bit_uint_time_low      |       16_bit_uint_random      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       32_bit_uint_random                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       32_bit_uint_random                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

## 3.2 Module contents

### 3.2.1 ULID class

**class** ulid.**ULID**(*seed=None*)

Instances of the ULID class represent ULIDS as specified in the ULID spec. ULIDS have 128-bit compatibility with UUID, are Lexicographically sortable, case insensitive and URL safe

**decode**(*s: str*) → Tuple[int, int]

Given a properly formed ULID, return a tuple containing the timestamp and the randomness component as ints

**Parameters** **s** (*str*) – ULID string to decode

**Returns** (timestamp, randomness)

**Return type** tuple(int, int)

**Raises** TypeError, ValueError

```
>>> from ulid import ULID
>>> ulid = ULID()
>>> ulid.decode('01BX5ZZKBKACTAV9WEVGEMMVRY')
(1508808576371, 392928161897179156999966)
```

**encode**(*i: int*) → str

Convert a given integer into the canonical ULID string format

**Parameters** **i** (*int*) – The integer to convert

**Returns** Canonically encoded ULID string

**Return type** str

**Raises** TypeError, ValueError

```
>>> from ulid import ULID
>>> ulid = ULID()
>>> ulid.encode(340282366920938463463374607431768167)
00864KEJY6MZQSVCHD1SB08637
```

**encode_timestamp**(*t: int*) → str

Convert a given unix timestamp into the canonical ULID string format

**Parameters** **t** (*int*) – The unix timestamp to convert

**Returns** Canonically encoded ULID string

**Return type** str

**Raises** TypeError, ValueError

```
>>> from ulid import ULID
>>> ulid = ULID()
>>> ulid.encode(281474976710655)
7ZZZZZZZZZ
```

**generate**() → str

Generate a 26 character ULID string encoded in Crockford's Base32

**Returns** Canonically encoded ULID string

**Return type** str

```
>>> from ulid import ULID
>>> ulid = ULID()
>>> ulid.generate()
01BX5ZZKBKACTAV9WEVGEMMVRZ
```

**pretty_print**(*s: str*) → None

Print the given ULID string in a binary layout

```
>>> from ulid import ULID
>>> ulid = ULID()
>>> ulid.pretty_print(ulid_str)
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      32_bit_uint_time_high                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     16_bit_uint_time_low      |       16_bit_uint_random      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       32_bit_uint_random                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       32_bit_uint_random                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

### 3.2.2 Monotonic class

**class** ulid.**Monotonic**(*seed=None*)

The Monotonic class represent an extension of the base ULID class with the addition of a monotonic sort order (correctly detects and handles the same millisecond)

**decode**(*s: str*) → Tuple[int, int]

Given a properly formed ULID, return a tuple containing the timestamp and the randomness component as ints

**Parameters** **s** (*str*) – ULID string to decode

**Returns** (timestamp, randomness)

**Return type** tuple(int, int)

**Raises** TypeError, ValueError

```
>>> from ulid import ULID
>>> ulid = ULID()
>>> ulid.decode('01BX5ZZKBKACTAV9WEVGEMMVRY')
(1508808576371, 392928161897179156999966)
```

**encode**(*i: int*) → str

Convert a given integer into the canonical ULID string format

**Parameters** **i** (*int*) – The integer to convert

**Returns** Canonically encoded ULID string

**Return type** str

**Raises** TypeError, ValueError

```
>>> from ulid import ULID
>>> ulid = ULID()
>>> ulid.encode(340282366920938463463374607431768167)
00864KEJY6MZQSVCHD1SB08637
```

**encode_timestamp**(*t: int*) → str

Convert a given unix timestamp into the canonical ULID string format

**Parameters** **t** (*int*) – The unix timestamp to convert

**Returns** Canonically encoded ULID string

**Return type** str

**Raises** TypeError, ValueError

```
>>> from ulid import ULID
>>> ulid = ULID()
>>> ulid.encode(281474976710655)
7ZZZZZZZZZ
```

**generate**() → str

Generate a 26 character ULID string encoded in Crockford's Base32

**Returns** Canonically encoded ULID string

**Return type** str

**Raises** ValueError

```
>>> from ulid import Monotonic
>>> ulid = Monotonic()
>>> ulid.generate()
01BX5ZZKBKACTAV9WEVGEMMVRZ
```

**pretty_print**(*s: str*) → None

Print the given ULID string in a binary layout

```
>>> from ulid import ULID
>>> ulid = ULID()
>>> ulid.pretty_print(ulid_str)
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      32_bit_uint_time_high                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     16_bit_uint_time_low      |       16_bit_uint_random      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       32_bit_uint_random                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       32_bit_uint_random                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

# CHAPTER 4

# Inspiration

UUID can be suboptimal for many uses-cases because:

- It isn't the most character efficient way of encoding 128 bits of randomness
- UUID v1/v2 is impractical in many environments, as it requires access to a unique, stable MAC address
- UUID v3/v5 requires a unique seed and produces randomly distributed IDs, which can cause fragmentation in many data structures
- UUID v4 provides no other information than randomness which can cause fragmentation in many data structures

Instead, herein is proposed ULID:

- 128-bit compatibility with UUID
- 1.21e+24 unique ULIDs per millisecond
- Lexicographically sortable!
- Canonically encoded as a 26 character string, as opposed to the 36 character UUID
- Uses Crockford's base32 for better efficiency and readability (5 bits per character)
- Case insensitive
- No special characters (URL safe)
- Monotonic sort order (correctly detects and handles the same millisecond)

# CHAPTER 5

# Prior Art

Partly inspired by:

- Instagram
- Firebase

CHAPTER 6

# README


The py-ulid library is a *minimal*, self-contained implementation of the ULID (Universally Unique Lexicographically Sortable Identifier) specification in Python. For more information, please refer to the official specification.

## 6.1 Installation

You can install the py-ulid library from PyPi

```
pip install py-ulid
```

The py-ulid library can be used in any version of python >= 3.5 and does not require *any* additional packages or modules.

## 6.2 How to use

To generate a ULID, simple run the generate() function

```
from ulid import ULID

#Instantiate the ULID class
ulid = ULID()
ulid.generate()  #01BX5ZZKBKACTAV9WEVGEMMVRZ
```

### 6.2.1 Seeding Time

You can instantiate the instance of the ULID class with a seed time which will output the same string for the time component. This could be useful when migrating to ulid

```
from ulid import ULID

#Instantiate the ULID class
ulid = ULID(1469918176385)
ulid.generate()  #01ARYZ6S41TSV4RRFFQ69G5FAV
```

### 6.2.2 Monotonic ULIDs

```
from ulid import Monotonic

#Instantiate the Monotonic Class
ulid = Monotonic()

# Same timestamp when calls are made within the same
# millisecond and least-significant random bit is incremented by 1
ulid.generate()  #01DC8Y7RBV4RSXX0437Z1RQR11
ulid.generate()  #01DC8Y7RBV4RSXX0437Z1RQR12
ulid.generate()  #01DC8Y7RBV4RSXX0437Z1RQR13
ulid.generate()  #01DC8Y7RBV4RSXX0437Z1RQR14
ulid.generate()  #01DC8Y7RBV4RSXX0437Z1RQR15
ulid.generate()  #01DC8Y7RBV4RSXX0437Z1RQR16
ulid.generate()  #01DC8Y7RBV4RSXX0437Z1RQR17
ulid.generate()  #01DC8Y7RBV4RSXX0437Z1RQR18
ulid.generate()  #01DC8Y7RBV4RSXX0437Z1RQR19
```

CHAPTER 7

# Indices and tables

- genindex
- modindex
- search

# Python Module Index

## u

# Index