
Binmap

Jul 11, 2020

Contents:

1	binmap	3
2	Indices and tables	5
	Python Module Index	7
	Index	9

Dataclass for go to and from binary data

It follows dataclass pattern with typehinting as the binary format. Temperature with one unsigned byte:

```
class Temperature(BinmapDataclass):
    temp: unsignedchar = 0

t = Temperature()
t.temp = 22
print(bytes(t))
b'\x16'

t2 = Temperature(b'\x20')
print(t2.temp)
32
```

Temperature and humidity consisting of one signed byte for temperature and one unsigned byte for humidity:

```
class TempHum(BinmapDataclass):
    temp: signedchar = 0
    hum: unsignedchar = 0

th = TempHum()
th.temp = -10
th.humidity = 60
print(bytes(th))
b'\xfc<'

th2 = TempHum(b'\xea\x41')
print(th2.temp)
-22
print(th2.hum)
65
```


CHAPTER 1

binmap

class binmap.**BaseDescriptor** (*name=""*)

Bases: object

Base class for all descriptors

Parameters **name** – Variable name

class binmap.**BinField** (*name=""*)

Bases: *binmap.BaseDescriptor*

BinField descriptor tries to pack it into a struct before setting the value as a bounds checker

class binmap.**BinmapDataclass** (*_binarydata: dataclasses.InitVar = b""*)

Bases: object

Dataclass that does the converting to and from binary data

frombytes (*value: bytes*)

Unpacks value to each field :param bytes value: binary string to unpack

class binmap.**ConstField** (*name=""*)

Bases: *binmap.BinField*

ConstField descriptor keeps it's value

class binmap.**EnumField** (*name=""*)

Bases: *binmap.BinField*

EnumField descriptor uses “enum” to map to and from strings. Accepts both strings and values when setting. Only values that has a corresponding string is allowed.

class binmap.**PaddingField** (*name=""*)

Bases: *binmap.BaseDescriptor*

PaddingField descriptor is used to “pad” data with values unused for real data

binmap.**constant** (*value: Union[int, float, str]*) → dataclasses.Field

Field generator function for constant elements

Parameters **value** – Constant value for the field.

Returns dataclass field

`binmap.enumfield(enumclass: enum.IntEnum, default: enum.IntEnum = None) → dataclasses.Field`
Field generator function for enum field

Parameters

- **enumclass** (*IntEnum*) – Class with enums.
- **default** (*IntEnum*) – default value

Returns dataclass field

`binmap.padding(length: int = 1) → dataclasses.Field`
Field generator function for padding elements

Parameters **length** (*int*) – Number of bytes of padded field

Returns dataclass field

`binmap.stringfield(length: int = 1, default: bytes = b'', fillchar: bytes = b' ') → dataclasses.Field`
Field generator function for string fields.

Parameters

- **length** (*int*) – length of the string.
- **default** (*bytes*) – default value of the string
- **fillchar** (*bytes*) – char to pad the string with

Returns dataclass field

CHAPTER 2

Indices and tables

- `genindex`
- `modindex`
- `search`

b

`binmap`, 3

B

`BaseDescriptor` (*class in binmap*), 3
`BinField` (*class in binmap*), 3
`binmap` (*module*), 3
`BinmapDataclass` (*class in binmap*), 3

C

`constant()` (*in module binmap*), 3
`ConstField` (*class in binmap*), 3

E

`EnumField` (*class in binmap*), 3
`enumfield()` (*in module binmap*), 4

F

`frombytes()` (*binmap.BinmapDataclass method*), 3

P

`padding()` (*in module binmap*), 4
`PaddingField` (*class in binmap*), 3

S

`stringfield()` (*in module binmap*), 4