structures

Release 3.0

May 01, 2014

The **structures** package is a library for creating data structures with type controling and casting. There is wide enough set of data types and a convenient method of designing of new. The **structures** are similar to the basicproperty but more user-friendly and readable.

The full documentation can be found there: http://zzzsochi.github.com/structures/

Source code placed on github.

CHAPTER 1

Python 3

Version >= 2.0 only for python 3!

For python 2 you must install structures<2.

Documentation writed for python 3.

CHAPTER 2

Concept

All the structures are created from the class inheritance **Structure**:

```
>>> from structures import Structure
>>> class S(Structure):
...     attr = 10
...
>>> s = S()
>>> s.attr
10
>>> s.attr = 'string'
>>> s.attr
'string'
```

Structures can inherit from other structures.

2.1 Types of attributes

```
>>> from structures import *
>>> class S(Structure):
...     i = Integer
...     s = String('string')  # setting default value
...
>>> s = S()
>>> s.s
'string'
>>> s.i
Traceback (most recent call last):
  ...
AttributeError: i
```

Now, if you assign a values to attributes *s* and *i* it will be automatically cast to the appropriate types.

```
>>> s.i = '13'
>>> type(s.i), s.i
(<type 'int'>, 13)
>>> s.s = 3.14
>>> type(s.s), s.s
(<type 'str'>, u'3.14')
>>> s.i = 'not a number'
Traceback (most recent call last):
```

```
  ...
ValueError: invalid literal for int() with base 10: 'not a number'
```

2.2 Structure can contain other structures

```
>>> from structures import *
>>> class S(Structure):
...     class s(Structure):
...         i = Integer(13)
...     f = Float(3.14)
...     d = Decimal('8.62')
...
>>> s = S()
>>> isinstance(s.s, Structure)
True
>>> s.s.i
13
>>> s.s.i = 9.18
>>> s.s.i
9
```

CHAPTER 3

Contents:

3.1 Types of attributes

Contents

3.1.1 Standart types

Default argument is took the first parametr in all standart types. In case the class (not instance) is used, it is emulated the absent this attribute.

```
>>> from structures import *
>>> class S(Structure):
...     d1 = Decimal
...     d2 = Decimal('2.62')
...
>>> s = S()
>>> s.d1
```

```
Traceback (most recent call last):
  ...
AttributeError: d1
>>> s.d2
Decimal('2.62')
>>> s.d1 = '3.14'
>>> s.d1
Decimal('3.14')
```

A simple types

Type	func
Integer	int
Float	float
Decimal	decimal.Decimal
Boolean	bool
Bytes	bytes
Binary	deprecated
String	str

Integer *(default=NoDefault)*

int is used for casting. It's used base 10 for strings.

Float *(default=NoDefault)*

float is used for casting.

Decimal *(default=NoDefault)*

decimal.Decimal is used for casting.

Boolean *(default=NoDefault)*

bool is used for casting.

Bytes *(default=NoDefault)*

bytes is used for casting.

Binary *(default=NoDefault)*

Deprecated. Use *Bytes*.

String *(default=NoDefault, enc="UTF-8")*

str is used for casting. Value **enc** is the encoding for casting from **bytes**.

Containers

Type	func
List	list
Tuple	tuple
Set	set
FrozenSet	frozenset
Dict	dict

Defult values are recreated in the mutable containers types (*List, Set & Dict*) during creating the instance of structure.

```
>>> from structures import *
>>> l = [1, 2]
>>> class S(Structure):
...     ll = List(l)
...
>>> s1 = S()
>>> s1.ll
[1, 2]
>>> assert s1.ll is not l
>>> assert s1.ll == l
>>> s2 = S()
>>> assert s1.ll is not s2.ll
>>> assert s1.ll == s2.ll
```

List *(default=NoDefault)*

list is used for casting.

Tuple *(default=NoDefault)*

tuple is used for casting.

Set *(default=NoDefault)*

set is used for casting.

Dict *(default=NoDefault)*

dict is used for casting.

Date and time types

Value **format** is **time.strftime** formating string. If it's **None** (default), ISO-like format is used.

DateTime *(default=NoDefault, format=None)*

It accepts **datetime.datetime**, **datetime.date** (setting time as *00:00:00*) and strings.

One of next formats is used for parsing strings if **format** is **None**:

- YYYY-MM-DD HH:MM:SS.mmmmmm
- YYYY-MM-DDTHH:MM:SS.mmmmmm
- YYYY-MM-DD HH:MM:SS
- YYYY-MM-DDTHH:MM:SS
- YYYY-MM-DD HH:MM
- YYYY-MM-DDTHH:MM
- YYYY-MM-DD

Date *(default=NoDefault, format=None)*

It accepts **datetime.date**, **datetime.datetime** (date is took only) and strings.

One of next formats is used for parsing strings if **format** is **None** and date is took only:

- YYYY-MM-DD HH:MM:SS.mmmmmm
- YYYY-MM-DDTHH:MM:SS.mmmmmm
- YYYY-MM-DD HH:MM:SS
- YYYY-MM-DDTHH:MM:SS
- YYYY-MM-DD HH:MM
- YYYY-MM-DDTHH:MM
- YYYY-MM-DD

Time *(default=NoDefault, format=None)*

It accepts **datetime.time**, **datetime.datetime** (time is took only) and strings.

One of next formats is used for parsing strings if **format** is **None** and time is took only:

- YYYY-MM-DD HH:MM:SS.mmmmmm
- YYYY-MM-DDTHH:MM:SS.mmmmmm
- YYYY-MM-DD HH:MM:SS
- YYYY-MM-DDTHH:MM:SS
- YYYY-MM-DD HH:MM
- YYYY-MM-DDTHH:MM
- HH:MM:SS
- HH:MM

3.1.2 Building custom types

Pleace read the **structures.types** module for detals... It's realy simple!

3.2 Transformation structures into a dict and backwards

E.g. transfirmation structures into a dict is used for passing structures to **pickle**...

3.2.1 to_dict(structure)

The function creates a dict with structure's contents. It works recursively.

```
>>> from structures import *
>>> class S(Structure):
...     class s(Structure):
...         i = Integer(13)
...     f = Float(3.14)
...     l = List([1, set([2, 3])])
...
>>> s = S()
>>> assert to_dict(s) == {'s': {'i': 13}, 'f': 3.14, 'l': [1, set([2, 3])]}
```

3.2.2 from_dict(structure_class, data)

The function creates structure **structure_class** from the dict **data**. It creates substructures appropriate types too i.e. it works recursively.

```
>>> from structures import *
>>> class S(Structure):
...     class s(Structure):
...         i = Integer(13)
...     f = Float(3.14)
...     d = Decimal('8.62')
...
>>> s = from_dict(S, {'s': {'i': 26}, 'd': '2.64', 'f': 9.82})
>>> assert s.f == 9.82
>>> assert s.s.i == 26
>>> s.d
Decimal('2.64')
```