Introduction

Construct is a powerful **declarative** and **symmetrical** parser and builder for binary data.

Instead of writing *imperative code* to parse a piece of data, you declaratively define a *data structure* that describes your data. As this data structure is not code, you can use it in one direction to *parse* data into Pythonic objects, and in the other direction, to *build* objects into binary data.

The library provides both simple, atomic constructs (such as integers of various sizes), as well as composite ones which allow you form hierarchical and sequential structures of increasing complexity. Construct features **bit and byte granularity**, easy debugging and testing, an **easy-to-extend subclass system**, and lots of primitive constructs to make your work easier:

Fields: raw bytes or numerical types
Structs and Sequences: combine simpler constructs into more complex ones
Bitwise: splitting bytes into bit-grained fields
Adapters: change how data is represented
Arrays/Ranges: duplicate constructs
Meta-constructs: use the context (history) to compute the size of data
If/Switch: branch the computational path based on the context
On-demand (lazy) parsing: read and parse only the fields what you require
Pointers: jump from here to there in the data stream
Tunneling: prefix data with a byte count or compress it

Example ¶

A `Struct` is a collection of ordered, named fields:

```
>>> format = Struct(
...     "signature" / Const(b"BMP"),
...     "width" / Int8ub,
...     "height" / Int8ub,
...     "pixels" / Array(this.width * this.height, Byte),
... )
>>> format.build(dict(width=3,height=2,pixels=[7,8,9,11,12,13]))
b'BMP\x03\x02\x07\x08\t\x0b\x0c\r'
>>> format.parse(b'BMP\x03\x02\x07\x08\t\x0b\x0c\r')
Container(signature=b'BMP')(width=3)(height=2)(pixels=[7, 8, 9, 11, 12, 13])
```

A `Sequence` is a collection of ordered fields, and differs from `Array` and `GreedyRange` in that those two are homogenous:

```
>>> format = Sequence(PascalString(Byte, "utf8"), GreedyRange(Byte))
>>> format.build([u"lalaland", [255,1,2]])
b'\nlalaland\xff\x01\x02'
>>> format.parse(b"\x004361789432197")
['', [52, 51, 54, 49, 55, 56, 57, 52, 51, 50, 49, 57, 55]]
```

Construct has been used to parse:

Networking formats like Ethernet, IP, ICMP, IGMP, TCP, UDP, DNS, DHCP
Binary file formats like Bitmaps, PNG, GIF, EMF, WMF
Executable binaries formats like ELF32, PE32
Filesystem layouts like Ext2, Fat16, MBR

See more examples in **current gallery** and in **deprecated gallery**.

Development and support

Please use **github issues** to ask general questions, make feature requests (and vote for them), report issues and bugs, and to submit PRs. Feel free to request any changes that would support your project.

Main documentation is at **readthedocs**, which is substantial. Source is at **github**. Releases are available at **pypi**.

latest

Requirements

Construct should run on CPython 3.8 3.9 3.10 3.11 3.12 (and probably beta) and PyPy implementations. PyPy achieves much better performance.

Following modules are needed only if you want to use certain features:

Numpy is optional, if you want to serialize arrays using Numpy protocol. Otherwise arrays can still be serialized using PrefixedArray.
Arrow is optional, if you want to use Timestamp class.
Different Python versions support different compression modules (like gzip lzma), if you want to use Compressed class.
Ruamel.yaml is optional, if you want to use KaitaiStruct (KSY) exporter.
Cloudpickle is optional, if you want to serialize the classes.
LZ4 is optional, if you want to use CompressedLZ4 class.
Cryptography is optional, if you want to use Encrypted* classes.

Installing

The library is downloadable and installable from Pypi. Just use standard command-line. There are no hard dependencies, but if you would like to install all supported (not required) modules listed above, you can use the 2nd command-line form.

```
pip install construct
pip install construct[extras]
```

Type Hints / Type Annotations

As an extension to this library there is the **construct-typing** library, which provides PEP 561 compliant stub files for this library. It also provides extended adapters to describe complex structures using PEP 526 type annotations for improved static code analysis with mypy.

```
pip install construct-typing
```

Visual Editor

Another fancy extension to this library is the **construct-editor** visual editor for binary blobs that uses Construct parsing classes internally.

```
pip install construct-editor
construct-editor
```