Sphinx Test Documentation

Release 0.1

davidlj95 & ccebrecos

Jul 06, 2018

Contents:

This documentation aims to provide an example of how to properly document a Python package using the following technologies:

- Sphinx with the following extensions:
  - Autodocs to document Python's docstrings
  - Napoleon to use Google docstrings style
  - Doctest to test if the code in the comments do return what is expected

In order for the documentation to be visually friendly, we use the Read the docs theme

We also use the following notation for documenting types and use static typing checkers like MyPy

- PEP 484 for type hints and documentation

CHAPTER 1

How to

In order to create the documentation, as Sphinx mentions in their getting started page, you have a `Makefile` for UNIX-like systems and a `make.bat` file for Windows systems so in order to build the documentation locally you just have to execute the following command:

```
make html
```

To build the documentation in HTML. Then just open the `index.html` file in your browser and you're ready to go.

CHAPTER 2

Repository

All this code is located at a GitHub repository

CHAPTER 3

Documentation

This documentation uses code from a GitHub project in order to see how well it gets documented. Indeed this sample project was made to be able to document that project. This project contains just one module called `bitcoin`

3.1 bitcoin package

Bitcoin framework module.

Intended to create transactions with smart contracts in Bitcoin in an easy way

exception bitcoin.**AddressPKHLengthError**
Bases: `ValueError`

Error raised when the public key hash has an incorrect length

exception bitcoin.**AddressSHLengthError**
Bases: `ValueError`

Error raised when the redeem script hash has an incorrect length

class bitcoin.**P2PKHAddress**(*public_key_hash: bytes*)
Bases: `object`

Models a legacy P2PKH Address

__init__(*public_key_hash: bytes*) → None
Initializes a P2PKH given its public key hash.

Modifies the attribute `_script_hash`

Parameters **public_key_hash** – public key hash to set

Raises

- `AssertionError` – public_key_hash argument is not a bytes object
- *`AddressPKHLengthError`* – public_key_hash argument has not the proper length of a public key hash

do_things(*obj: str*) → bytes
Does some things.

Parameters **obj** – object of options

Returns things modified by `obj` as bytes

public_key_hash
Returns the public key hash stored in the address.

Getter Returns the public key hash as bytes object

Setter Sets the public key hash, while verifying the length and type

class `bitcoin.`**P2SHAddress**(*script_hash: bytes*)
Bases: `object`

Models a legacy P2SH Address

__init__(*script_hash: bytes*) → None
Initializes a P2SH given its script hash.

Modifies the attribute `_script_hash`

Parameters **script_hash** – script hash to set

Raises

- `AssertionError` – script_hash argument is not a bytes object
- *AddressSHLengthError* – script_hash argument has not the proper length of a script hash

do_things(*obj: str*) → bytes
Does some things.

Parameters **obj** – object of options

Returns things modified by `obj` as bytes

script_hash
Returns the public key hash stored in the address.

Getter Returns the script hash as bytes object

Setter Sets the script hash, while verifying the length and type

`bitcoin.`**b58decode**(*value: str*) → bytes
Decodes a base58 string into a bytes object.

`bitcoin.`**b58encode**(*value: bytes*) → str
Encodes bytes using base58 convention into an human-readable string.

Sources

- Bitcoin Wiki: base 58

3.1.1 Subpackages

bitcoin.address package

Addresses module.

Contains models to handle Bitcoin addresses

class bitcoin.address.**P2PKHAddress**(*public_key_hash: bytes*)
Bases: object

Models a legacy P2PKH Address

__init__(*public_key_hash: bytes*) → None
Initializes a P2PKH given its public key hash.

Modifies the attribute _script_hash

Parameters **public_key_hash** – public key hash to set

Raises

- AssertionError – public_key_hash argument is not a bytes object
- *AddressPKHLengthError* – public_key_hash argument has not the proper length of a public key hash

do_things(*obj: str*) → bytes
Does some things.

Parameters **obj** – object of options

Returns things modified by obj as bytes

public_key_hash
Returns the public key hash stored in the address.

Getter Returns the public key hash as bytes object

Setter Sets the public key hash, while verifying the length and type

class bitcoin.address.**P2SHAddress**(*script_hash: bytes*)
Bases: object

Models a legacy P2SH Address

__init__(*script_hash: bytes*) → None
Initializes a P2SH given its script hash.

Modifies the attribute _script_hash

Parameters **script_hash** – script hash to set

Raises

- AssertionError – script_hash argument is not a bytes object
- *AddressSHLengthError* – script_hash argument has not the proper length of a script hash

do_things(*obj: str*) → bytes
Does some things.

Parameters **obj** – object of options

Returns things modified by obj as bytes

script_hash
Returns the public key hash stored in the address.

Getter Returns the script hash as bytes object

Setter Sets the script hash, while verifying the length and type

exception bitcoin.address.**AddressPKHLengthError**
Bases: ValueError

Error raised when the public key hash has an incorrect length

exception bitcoin.address.**AddressSHLengthError**
Bases: ValueError

Error raised when the redeem script hash has an incorrect length

Submodules

bitcoin.address.p2pkh module

Models P2PKH Addresses.

A P2PKH address is used by user to send bitcoins to another user who controls the private key behind the public key (whose hash appears in the address)

Most information was extracted from the Bitcoin Wiki.

exception bitcoin.address.p2pkh.**AddressPKHLengthError**
Bases: ValueError

Error raised when the public key hash has an incorrect length

class bitcoin.address.p2pkh.**P2PKHAddress**(*public_key_hash: bytes*)
Bases: object

Models a legacy P2PKH Address

__init__(*public_key_hash: bytes*) → None
Initializes a P2PKH given its public key hash.

Modifies the attribute _script_hash

Parameters **public_key_hash** – public key hash to set

Raises

- AssertionError – public_key_hash argument is not a bytes object
- *AddressPKHLengthError* – public_key_hash argument has not the proper length of a public key hash

do_things(*obj: str*) → bytes
Does some things.

Parameters **obj** – object of options

Returns things modified by obj as bytes

public_key_hash
Returns the public key hash stored in the address.

Getter Returns the public key hash as bytes object

Setter Sets the public key hash, while verifying the length and type

bitcoin.address.p2pkh.**PKH_LENGTH = 20**
int – length of the public key hashes stored in addresses

bitcoin.address.p2pkh.**from_hex**(*public_key_hash_hex: str*) → bitcoin.address.p2pkh.P2PKHAddress

Creates a P2PKH address from public key hash in hexadecimal.

Converts the hexadecimal string into bytes and then uses those bytes to create a P2PKH address.

```
>>> 1+1
2
```

Parameters **public_key_hash_hex** – string containing public key hash in hexadecimal

Returns pay to public key hash address

bitcoin.address.p2sh module

Models P2SH Addresses.

A P2SH address is used by user to send bitcoins to a Bitcoin smart contract

Most information was extracted from the Bitcoin Wiki.

exception bitcoin.address.p2sh.**AddressSHLengthError**

Bases: ValueError

Error raised when the redeem script hash has an incorrect length

class bitcoin.address.p2sh.**P2SHAddress**(*script_hash: bytes*)

Bases: object

Models a legacy P2SH Address

__init__(*script_hash: bytes*) → None

Initializes a P2SH given its script hash.

Modifies the attribute _script_hash

Parameters **script_hash** – script hash to set

Raises

- AssertionError – script_hash argument is not a bytes object
- *AddressSHLengthError* – script_hash argument has not the proper length of a script hash

do_things(*obj: str*) → bytes

Does some things.

Parameters **obj** – object of options

Returns things modified by obj as bytes

script_hash

Returns the public key hash stored in the address.

Getter Returns the script hash as bytes object

Setter Sets the script hash, while verifying the length and type

bitcoin.address.p2sh.**SH_LENGTH = 20**

int – length of the script hashes stored in addresses

bitcoin.address.p2sh.**from_hex**(*script_hash_hex: str*) → bitcoin.address.p2sh.P2SHAddress
Creates a P2SH address from script hash in hexadecimal.

Converts the hexadecimal string into bytes and then uses those bytes to create a P2SH address.

```
>>> 2+3
5
```

Parameters **script_hash_hex** – string containing script hash in hexadecimal

Returns pay to script hash address

bitcoin.encoding package

Provides methods to allow encoding and decoding Bitcoin addresses.

bitcoin.encoding.**b58encode**(*value: bytes*) → str
Encodes bytes using base58 convention into an human-readable string.

Sources

- Bitcoin Wiki: base 58

bitcoin.encoding.**b58decode**(*value: str*) → bytes
Decodes a base58 string into a bytes object.

bitcoin.encoding.**bech32encode**(*hrp*, *version*, *program*)
Encodes bytes using bech32 convention into an human-readable string.

bitcoin.encoding.**bech32decode**(*string*)
Decodes a bech32 string into a bytes object.

Submodules

bitcoin.encoding.base58 module

Provides methods to encode/decode to/from base58.

bitcoin.encoding.base58.**ALPHABET = '123456789ABCDEFGHJKLMNPQRSTUVWXYZabcdefghijkmnopqrstuv**
str – base58 domain

bitcoin.encoding.base58.**decode**(*value: str*) → bytes
Decodes a base58 string into a bytes object.

bitcoin.encoding.base58.**encode**(*value: bytes*) → str
Encodes bytes using base58 convention into an human-readable string.

Sources

- Bitcoin Wiki: base 58

bitcoin.encoding.bech32 module

Provides methods to encode/decode to/from bech32.

Sources

- Pieter Wuille reference implementation

bitcoin.encoding.bech32.**decode**(*string*)
Decodes a bech32 string into a bytes object.

bitcoin.encoding.bech32.**encode**(*hrp*, *version*, *program*)
Encodes bytes using bech32 convention into an human-readable string.

Python Module Index

b

Index

Symbols

A

B

D

E

F

P

S