HDWallet

Release v2.1.1

Meheret Tesfaye Batu

Dec 23, 2022

CONTENTS

1 Hierarchical Deterministic Wallet **1**

2 Installing HDWallet **5**
2.1 Development 5

3 Command Line Interface (CLI) **7**

4 Cryptocurrencies **9**

5 HDWallet **15**
5.1 BIP32HDWallet 27
5.2 BIP44HDWallet 28
5.3 BIP49HDWallet 28
5.4 BIP84HDWallet 29
5.5 BIP141HDWallet 30

6 Derivation **33**
6.1 BIP32Derivation 34
6.2 BIP44Derivation 37
6.3 BIP49Derivation 38
6.4 BIP84Derivation 38
6.5 BIP141Derivation 39

7 Utils **41**

Python Module Index **45**

Index **47**

CHAPTER

ONE

HIERARCHICAL DETERMINISTIC WALLET

Python-based library for the implementation of a hierarchical deterministic wallet generator for over 140+ multiple cryptocurrencies. It allows the handling of multiple coins, multiple accounts, external and internal chains per account and millions of addresses per the chain.

Simple Bitcoin mainnet HDWallet generator:

```
#!/usr/bin/env python3

from hdwallet import HDWallet
from hdwallet.utils import generate_entropy
from hdwallet.symbols import BTC as SYMBOL
from typing import Optional

import json

# Choose strength 128, 160, 192, 224 or 256
STRENGTH: int = 160  # Default is 128
# Choose language english, french, italian, spanish, chinese_simplified, chinese_
↪traditional, japanese or korean
LANGUAGE: str = "korean"  # Default is english
# Generate new entropy hex string
ENTROPY: str = generate_entropy(strength=STRENGTH)
# Secret passphrase for mnemonic
PASSPHRASE: Optional[str] = None  # "meherett"

# Initialize Bitcoin mainnet HDWallet
hdwallet: HDWallet = HDWallet(symbol=SYMBOL, use_default_path=False)
# Get Bitcoin HDWallet from entropy
hdwallet.from_entropy(
    entropy=ENTROPY, language=LANGUAGE, passphrase=PASSPHRASE
)

# Derivation from path
# hdwallet.from_path("m/44'/0'/0'/0/0")
# Or derivation from index
hdwallet.from_index(44, hardened=True)
hdwallet.from_index(0, hardened=True)
hdwallet.from_index(0, hardened=True)
hdwallet.from_index(0)
```

(continues on next page)

(continued from previous page)

```
hdwallet.from_index(0)

# Print all Bitcoin HDWallet information's
print(json.dumps(hdwallet.dumps(), indent=4, ensure_ascii=False))
```


```
{
    "cryptocurrency": "Bitcoin",
    "symbol": "BTC",
    "network": "mainnet",
    "strength": 160,
    "entropy": "c5b0d0ee698f3f72b6265f1bc591f8f2d7afa6dd",
    "mnemonic": "                ",
    "language": "korean",
    "passphrase": null,
    "seed":
↪"5a9b9667ccd07b3c641b1ba95e9119dd1d5a3034fd46cd2f27fc1f160c7dcd824fc0ab4710a9ae90582dffc3b0803bcbc0a8
↪",
    "root_xprivate_key":
↪"xprv9s21ZrQH143K2qMHU8aghJ4MoQR5g5mowXbeP2vCP937bseZGX929dmJudL7u4xRxtKvh58pxz1PhtCbWW2yUH14jdduKVMV
↪",
    "root_xpublic_key":
↪"xpub661MyMwAqRbcFKRkaA7h4S16MSFa5YVfJkXFBRKowUa6Ufyhp4TGhS5nkvkLXSmdNjoszzDkU26WW2rg1zBsQBt6Pv3T8oLE
↪",
    "xprivate_key":
↪"xprvA2YyMZWyPK2xo4eZgyypp2CzcHnxNzGbruGg7vmgaAVCtBtrjwzuhXJBNM3FrwBh85ajxHErNR6ByN77WJARpC1HDC7kTwa2y
↪",
    "xpublic_key":
↪"xpub6FYKm53sDgbG1Yj2o1WqBA9jAKdSnSzTE8CGvKBJ8W2BkzE1HVKAFKcfDcCHKpL5BQRg2HjbNSt55jpFshY7W1KFtp7zjB3Dk
↪",
    "uncompressed":
↪"08101637 0b45d7e23bd89b07d6886036f5e4df9a129eee3b488c177ba7881856e24d337b280f9d32539a22445e567543b39b
↪",
    "compressed": "03081016370b45d7e23bd89b07d6886036f5e4df9a129eee3b488c177ba7881856",
    "chain_code": "cf9ee427ed8073e009a5743056e8cf19167f67ca5082c2c6635b391e9a4e0b0d",
    "private_key": "f79495fda777197ce73551bcd8e162ceca19167575760d3cc2bced4bf2a213dc",
    "public_key": "03081016370b45d7e23bd89b07d6886036f5e4df9a129eee3b488c177ba7881856",
    "wif": "L5WyVfBu8Sz3iGZtrwJVSP2wDJmu7HThGd1EGekFBnviWgzLXpJd",
    "finger_print": "ac13e305",
    "semantic": "p2pkh",
    "path": "m/44'/0'/0'/0/0",
    "hash": "ac13e305a88bd9968f1c058fcf5d9a6b1b9ef484",
    "addresses": {
        "p2pkh": "1Ggs3kkNrPPWoW17iDFQWgMdw3CD8BzBiv",
        "p2sh": "3GQVUFePz517Hf61Vsa9H2tHj5jw5y6ngV",
        "p2wpkh": "bc1q4sf7xpdg30vedrcuqk8u7hv6dvdeaayy3uw5cj",
        "p2wpkh_in_p2sh": "3JyV5aSgdVYEjQodPWHfvehQ5227EDr3sN",
        "p2wsh": "bc1qnk0s9q4379n6v9vg0lnhdu5qhjyx99u2xm238pmckmjg9v29q54saddzp9",
        "p2wsh_in_p2sh": "3MmsEoP7GLHzuLVgkAtcRtyXLTWh8zNAcd"
    }
}
```


For more info see the BIP specs.

BIP's	Titles
BIP39	Mnemonic code for generating deterministic keys
BIP85	Deterministic Entropy From BIP32 Keychains
BIP32	Hierarchical Deterministic Wallets
BIP44	Multi-Account Hierarchy for Deterministic Wallets
BIP49	Derivation scheme for P2WPKH-nested-in-P2SH based accounts
BIP84	Derivation scheme for P2WPKH based accounts
BIP141	Segregated Witness (Consensus layer)

CHAPTER

TWO

INSTALLING HDWALLET

The easiest way to install `hdwallet` is via pip:

```
$ pip install hdwallet
```

To install `hdwallet` command line interface globally, for Linux *sudo* may be required:

```
$ pip install hdwallet[cli]
```

After you have installed, type `hdwallet` to verify that it worked:

```
$ hdwallet
Usage: hdwallet [OPTIONS] COMMAND [ARGS]...

Options:
  -v, --version  Show HDWallet version and exit.
  -h, --help     Show this message and exit.

Commands:
  generate (g)  Select Generate for HDWallet.
  list (l)      Select List for HDWallet information.
```

If you want to run the latest version of the code, you can install from git:

```
$ pip install git+git://github.com/meherett/python-hdwallet.git
```

For the versions available, see the tags on this repository.

2.1 Development

We welcome pull requests. To get started, just fork this github repository, clone it locally, and run:

```
$ pip install -e .[cli,tests,docs] -r requirements.txt
```

CHAPTER

THREE

COMMAND LINE INTERFACE (CLI)

CHAPTER

FOUR

CRYPTOCURRENCIES

This library simplifies the process of generating a new HDWallet's for:

Note: All Cryptocurrencies testnet networks default paths are set to `m/44'/1'/0'/0/0` value.

Cryptocurrencies	Symbols	Mainnet	Testnet	Segwit	Coin Type	Default Paths
Anon	ANON	Yes	No	No	220	m/44'/220'/0'/0/0
Argoneum	AGM	Yes	No	No	421	m/44'/421'/0'/0/0
Artax	XAX	Yes	No	No	219	m/44'/219'/0'/0/0
Aryacoin	AYA	Yes	No	No	357	m/44'/357'/0'/0/0
Asiacoin	AC	Yes	No	No	51	m/44'/51'/0'/0/0
Atom	ATOM	Yes	No	Yes	118	m/44'/118'/0'/0/0
Auroracoin	AUR	Yes	No	No	85	m/44'/85'/0'/0/0
Axe	AXE	Yes	No	No	4242	m/44'/4242'/0'/0/0
Bata	BTA	Yes	No	No	89	m/44'/89'/0'/0/0
Beetle Coin	BEET	Yes	No	No	800	m/44'/800'/0'/0/0
Bela Coin	BELA	Yes	No	No	73	m/44'/73'/0'/0/0
Bit Cloud	BTDX	Yes	No	No	218	m/44'/218'/0'/0/0
Bit Send	BSD	Yes	No	No	91	m/44'/91'/0'/0/0
Bitcoin Cash	BCH	Yes	No	Yes	145	m/44'/145'/0'/0/0
Bitcoin Gold	BTG	Yes	No	Yes	156	m/44'/156'/0'/0/0
Bitcoin	BTC, BTCTEST	Yes	Yes	Yes	0	m/44'/0'/0'/0/0
Bitcoin Plus	XBC	Yes	No	No	65	m/44'/65'/0'/0/0
Bitcoin SV	BSV	Yes	No	No	236	m/44'/236'/0'/0/0
BitcoinZ	BTCZ	Yes	No	No	177	m/44'/177'/0'/0/0
Bitcore	BTX	Yes	No	Yes	160	m/44'/160'/0'/0/0
Blackcoin	BLK	Yes	No	No	10	m/44'/10'/0'/0/0
Block Stamp	BST	Yes	No	Yes	254	m/44'/254'/0'/0/0
Blocknode	BND, BNDTEST	Yes	Yes	No	2941	m/44'/2941'/0'/0/0
Bolivarcoin	BOLI	Yes	No	No	278	m/44'/278'/0'/0/0
Brit Coin	BRIT	Yes	No	No	70	m/44'/70'/0'/0/0
CPU Chain	CPU	Yes	No	Yes	363	m/44'/363'/0'/0/0

continues on next page

Table 1 – continued from previous page

Cryptocurrencies	Symbols	Mainnet	Testnet	Segwit	Coin Type	Default Paths
Canada eCoin	CDN	Yes	No	No	34	m/44'/34'/0'/0/0
Cannacoin	CCN	Yes	No	No	19	m/44'/19'/0'/0/0
Clams	CLAM	Yes	No	No	23	m/44'/23'/0'/0/0
Club Coin	CLUB	Yes	No	No	79	m/44'/79'/0'/0/0
Compcoin	CMP	Yes	No	No	71	m/44'/71'/0'/0/0
Crane Pay	CRP	Yes	No	Yes	2304	m/44'/2304'/0'/0/0
Crave	CRAVE	Yes	No	No	186	m/44'/186'/0'/0/0
Dash	DASH, DASHTEST	Yes	Yes	No	5	m/44'/5'/0'/0/0
Deep Onion	ONION	Yes	No	Yes	305	m/44'/305'/0'/0/0
Defcoin	DFC	Yes	No	No	1337	m/44'/1337'/0'/0/0
Denarius	DNR	Yes	No	No	116	m/44'/116'/0'/0/0
Diamond	DMD	Yes	No	No	152	m/44'/152'/0'/0/0
Digi Byte	DGB	Yes	No	Yes	20	m/44'/20'/0'/0/0
Digitalcoin	DGC	Yes	No	No	18	m/44'/18'/0'/0/0
Dogecoin	DOGE, DOGETEST	Yes	Yes	No	3	m/44'/3'/0'/0/0
EDR Coin	EDRC	Yes	No	No	56	m/44'/56'/0'/0/0
Ecoin	ECN	Yes	No	No	115	m/44'/115'/0'/0/0
Einsteinium	EMC2	Yes	No	No	41	m/44'/41'/0'/0/0
Elastos	ELA	Yes	No	No	2305	m/44'/2305'/0'/0/0
Energi	NRG	Yes	No	No	9797	m/44'/9797'/0'/0/0
Ethereum	ETH	Yes	No	Yes	60	m/44'/60'/0'/0/0
Europe Coin	ERC	Yes	No	No	151	m/44'/151'/0'/0/0
Exclusive Coin	EXCL	Yes	No	No	190	m/44'/190'/0'/0/0
FIX	FIX, FIXTEST	Yes	Yes	No	336	m/44'/336'/0'/0/0
Feathercoin	FTC	Yes	No	No	8	m/44'/8'/0'/0/0
Firstcoin	FRST	Yes	No	No	167	m/44'/167'/0'/0/0
Flashcoin	FLASH	Yes	No	No	120	m/44'/120'/0'/0/0
Fuji Coin	FJC	Yes	No	Yes	75	m/44'/75'/0'/0/0
GCR Coin	GCR	Yes	No	No	49	m/44'/49'/0'/0/0
Game Credits	GAME	Yes	No	No	101	m/44'/101'/0'/0/0
Go Byte	GBX	Yes	No	No	176	m/44'/176'/0'/0/0
Gridcoin	GRC	Yes	No	No	84	m/44'/84'/0'/0/0
Groestl Coin	GRS, GRSTEST	Yes	Yes	Yes	17	m/44'/17'/0'/0/0
Gulden	NLG	Yes	No	No	87	m/44'/87'/0'/0/0
Hellenic-coin	HNC	Yes	No	No	168	m/44'/168'/0'/0/0
Hempcoin	THC	Yes	No	No	113	m/44'/113'/0'/0/0
Hush	HUSH	Yes	No	No	197	m/44'/197'/0'/0/0
IX Coin	IXC	Yes	No	No	86	m/44'/86'/0'/0/0

continues on next page

Table 1 – continued from previous page

Cryptocurrencies	Symbols	Mainnet	Testnet	Segwit	Coin Type	Default Paths
Insane Coin	INSN	Yes	No	No	68	m/44'/68'/0'/0/0
Internet Of People	IOP	Yes	No	No	66	m/44'/66'/0'/0/0
Jumbucks	JBS	Yes	No	No	26	m/44'/26'/0'/0/0
Kobocoin	KOBO	Yes	No	No	196	m/44'/196'/0'/0/0
Komodo	KMD	Yes	No	No	141	m/44'/141'/0'/0/0
LBRY Credits	LBC	Yes	No	No	140	m/44'/140'/0'/0/0
Linx	LINX	Yes	No	No	114	m/44'/114'/0'/0/0
Litecoin Cash	LCC	Yes	No	No	192	m/44'/192'/0'/0/0
Litecoin	LTC, LTCTEST	Yes	Yes	Yes	2	m/44'/2'/0'/0/0
LitecoinZ	LTZ	Yes	No	No	221	m/44'/221'/0'/0/0
Lkrcoin	LKR	Yes	No	No	557	m/44'/557'/0'/0/0
Lynx	LYNX	Yes	No	No	191	m/44'/191'/0'/0/0
Mazacoin	MZC	Yes	No	No	13	m/44'/13'/0'/0/0
Megacoin	MEC	Yes	No	No	217	m/44'/217'/0'/0/0
Minexcoin	MNX	Yes	No	No	182	m/44'/182'/0'/0/0
Monacoin	MONA	Yes	No	Yes	22	m/44'/22'/0'/0/0
Monkey Project	MONK	Yes	No	Yes	214	m/44'/214'/0'/0/0
Myriadcoin	XMY	Yes	No	No	90	m/44'/90'/0'/0/0
NIX	NIX	Yes	No	Yes	400	m/44'/400'/0'/0/0
Namecoin	NMC	Yes	No	No	7	m/44'/7'/0'/0/0
Navcoin	NAV	Yes	No	No	130	m/44'/130'/0'/0/0
Neblio	NEBL	Yes	No	No	146	m/44'/146'/0'/0/0
Neoscoin	NEOS	Yes	No	No	25	m/44'/25'/0'/0/0
Neurocoin	NRO	Yes	No	No	110	m/44'/110'/0'/0/0
New York Coin	NYC	Yes	No	No	179	m/44'/179'/0'/0/0
Novacoin	NVC	Yes	No	No	50	m/44'/50'/0'/0/0
NuBits	NBT	Yes	No	No	12	m/44'/12'/0'/0/0
NuShares	NSR	Yes	No	No	11	m/44'/11'/0'/0/0
OK Cash	OK	Yes	No	No	69	m/44'/69'/0'/0/0
Omni	OMNI, OMNITEST	Yes	Yes	No	200	m/44'/200'/0'/0/0
Onix Coin	ONX	Yes	No	No	174	m/44'/174'/0'/0/0
Peercoin	PPC	Yes	No	No	6	m/44'/6'/0'/0/0
Pesobit	PSB	Yes	No	No	62	m/44'/62'/0'/0/0
Phore	PHR	Yes	No	No	444	m/44'/444'/0'/0/0
Pinkcoin	PINK	Yes	No	No	117	m/44'/117'/0'/0/0
Pivx	PIVX, PIVXTEST	Yes	Yes	No	119	m/44'/119'/0'/0/0
Posw Coin	POSW	Yes	No	No	47	m/44'/47'/0'/0/0
Potcoin	POT	Yes	No	No	81	m/44'/81'/0'/0/0
Project Coin	PRJ	Yes	No	No	533	m/44'/533'/0'/0/0

continues on next page

Table 1 – continued from previous page

Cryptocurrencies	Symbols	Mainnet	Testnet	Segwit	Coin Type	Default Paths
Putincoin	PUT	Yes	No	No	122	m/44'/122'/0'/0/0
Qtum	QTUM, QTUMTEST	Yes	Yes	Yes	2301	m/44'/2301'/0'/0/0
RSK	RBTC, RBTCTEST	Yes	Yes	No	137	m/44'/137'/0'/0/0
Rapids	RPD	Yes	No	No	320	m/44'/320'/0'/0/0
Ravencoin	RVN	Yes	No	No	175	m/44'/175'/0'/0/0
Reddcoin	RDD	Yes	No	No	4	m/44'/4'/0'/0/0
Rubycoin	RBY	Yes	No	No	16	m/44'/16'/0'/0/0
Safecoin	SAFE	Yes	No	No	19165	m/44'/19165'/0'/0/0
Saluscoin	SLS	Yes	No	No	572	m/44'/572'/0'/0/0
Scribe	SCRIBE	Yes	No	No	545	m/44'/545'/0'/0/0
Shadow Cash	SDC, SDCTEST	Yes	Yes	No	35	m/44'/35'/0'/0/0
Slimcoin	SLM, SLMTEST	Yes	Yes	No	63	m/44'/63'/0'/0/0
Smileycoin	SMLY	Yes	No	No	59	m/44'/59'/0'/0/0
Solarcoin	SLR	Yes	No	No	58	m/44'/58'/0'/0/0
Stash	STASH	Yes	No	No	49344	m/44'/49344'/0'/0/0
Stratis	STRAT, STRATTEST	Yes	Yes	No	105	m/44'/105'/0'/0/0
Sugarchain	SUGAR, SUGARTEST	Yes	Yes	Yes	408	m/44'/408'/0'/0/0
Syscoin	SYS	Yes	No	Yes	57	m/44'/57'/0'/0/0
TOA Coin	TOA	Yes	No	No	159	m/44'/159'/0'/0/0
Thought AI	THT	Yes	No	No	502	m/44'/502'/0'/0/0
Tron	TRX	Yes	No	No	195	m/44'/195'/0'/0/0
Twins	TWINS, TWINSTEST	Yes	Yes	No	970	m/44'/970'/0'/0/0
Ultimate Secure Cash	USC	Yes	No	No	112	m/44'/112'/0'/0/0
Unobtanium	UNO	Yes	No	No	92	m/44'/92'/0'/0/0
Virtual Cash	VASH	Yes	No	No	33	m/44'/33'/0'/0/0
Vcash	VC	Yes	No	No	127	m/44'/127'/0'/0/0
Verge Currency	XVG	Yes	No	No	77	m/44'/77'/0'/0/0
Vertcoin	VTC	Yes	No	Yes	28	m/44'/28'/0'/0/0
Viacoin	VIA, VIATEST	Yes	Yes	Yes	14	m/44'/14'/0'/0/0
Vivo	VIVO	Yes	No	No	166	m/44'/166'/0'/0/0
Whitecoin	XWC	Yes	No	No	559	m/44'/559'/0'/0/0
Wincoin	WC	Yes	No	No	181	m/44'/181'/0'/0/0
XUEZ	XUEZ	Yes	No	No	225	m/44'/225'/0'/0/0
XinFin	XDC	Yes	No	Yes	550	m/44'/550'/0'/0/0

continues on next page

Table 1 – continued from previous page

Cryptocurrencies	Symbols	Mainnet	Testnet	Segwit	Coin Type	Default Paths
ZClassic	ZCL	Yes	No	No	147	m/44'/147'/0'/0/0
Zcash	ZEC, ZECTEST	Yes	Yes	No	133	m/44'/133'/0'/0/0
Zencash	ZEN	Yes	No	No	121	m/44'/121'/0'/0/0

CHAPTER

FIVE

HDWALLET

Class Hierarchical Deterministic Wallet

class hdwallet.hdwallet.**HDWallet**(*symbol: str = 'BTC'*, *cryptocurrency: Optional[Any] = None*, *semantic: Optional[str] = None*, *use_default_path: bool = False*)

Hierarchical Deterministic Wallet

Parameters

- **symbol** (*str*) – Cryptocurrency symbol, defaults to `BTC`.
- **cryptocurrency** (*Cryptocurrency*) – Cryptocurrency instance, defaults to `None`.
- **semantic** (*str*) – Extended semantic, defaults to `P2PKH`.
- **use_default_path** (*bool*) – Use default derivation path, defaults to `False`.

Returns HDWallet – Hierarchical Deterministic Wallet instance.

Note: To initialize HDWallet symbol or cryptocurrency is required.

---

from_entropy(*entropy: str*, *language: str = 'english'*, *passphrase: Optional[str] = None*) → *hdwallet.hdwallet.HDWallet*

Master from Entropy hex string.

Parameters

- **entropy** (*str*) – Entropy hex string.
- **language** (*str*) – Mnemonic language, default to `english`.
- **passphrase** (*str*) – Mnemonic passphrase or password, default to `None`.

Returns HDWallet – Hierarchical Deterministic Wallet instance.

```
>>> from hdwallet import HDWallet
>>> from hdwallet.symbols import BTC
>>> hdwallet = HDWallet(symbol=BTC)
>>> hdwallet.from_entropy(entropy="ee535b143b0d9d1f87546f9df0d06b1a", language=
↪"english", passphrase=None)
<hdwallet.hdwallet.HDWallet object at 0x000001E8BFB98D60>
```

from_mnemonic(*mnemonic: str*, *language: Optional[str] = None*, *passphrase: Optional[str] = None*) → *hdwallet.hdwallet.HDWallet*

Master from Mnemonic words.

Parameters

- **mnemonic** (*str*) – Mnemonic words.
- **language** (*str*) – Mnemonic language, default to None.
- **passphrase** (*str*) – Mnemonic passphrase or password, default to None.

Returns HDWallet – Hierarchical Deterministic Wallet instance.

```
>>> from hdwallet import HDWallet
>>> from hdwallet.symbols import BTC
>>> hdwallet = HDWallet(symbol=BTC)
>>> hdwallet.from_mnemonic(mnemonic="sceptre capter sequence girafe absolu␣
↪relatif fleur zoologie muscle sirop saboter parure", passphrase=None)
<hdwallet.hdwallet.HDWallet object at 0x000001E8BFB98D60>
```

from_seed(*seed: str*) → *hdwallet.hdwallet.HDWallet*

Master from Seed hex string.

Parameters **seed** (*str*) – Seed hex string.

Returns HDWallet – Hierarchical Deterministic Wallet instance.

```
>>> from hdwallet import HDWallet
>>> from hdwallet.symbols import ETH
>>> hdwallet = HDWallet(symbol=ETH)
>>> hdwallet.from_seed(seed=
↪"8d5f4fe5b81a6a6a18b08603b6b3f59df9f4bbb25d10c55d23e0cbdc5ee385e5fddad9d7e6114f11afdec459283
↪")
<hdwallet.hdwallet.HDWallet object at 0x000001E8BFB98D60>
```

from_xprivate_key(*xprivate_key: str*, *strict: bool = False*) → *hdwallet.hdwallet.HDWallet*

Master from XPrivate Key.

Parameters

- **xprivate_key** (*str*) – Root or Non-Root XPrivate key.
- **strict** (*bool*) – Strict for must be root xprivate key, default to False.

Returns HDWallet – Hierarchical Deterministic Wallet instance.

```
>>> from hdwallet import HDWallet
>>> from hdwallet.symbols import BTC
>>> hdwallet = HDWallet(symbol=BTC)
>>> hdwallet.from_xprivate_key(xprivate_key=
↪"xprv9s21ZrQH143K3xPGUzpogJeKtRdjHkK6muBJo8v7rEVRzT83xJgNcLpMoJXUf9wJFKfuHR4SGvfgdShh4t9Vmjj
↪")
<hdwallet.hdwallet.HDWallet object at 0x000001E8BFB98D60>
```

from_xpublic_key(*xpublic_key: str*, *strict: bool = False*) → *hdwallet.hdwallet.HDWallet*

Master from XPublic Key.

Parameters

- **xpublic_key** (*str*) – Root or Non-Root XPublic key.
- **strict** (*bool*) – Strict for must be root xpublic key, default to False.

Returns HDWallet – Hierarchical Deterministic Wallet instance.

```
>>> from hdwallet import HDWallet
>>> from hdwallet.symbols import BTC
>>> hdwallet = HDWallet(symbol=BTC)
>>> hdwallet.from_xpublic_key(xpublic_key=
↪"xpub661MyMwAqRbcGSTjb2Mp3Sb4STUDhD2x986ubXKjQa2QsFTCVqzdA98qeZjcncHT1AaZcMSjiP1HJ16jH97q72R
↪")
<hdwallet.hdwallet.HDWallet object at 0x000001E8BFB98D60>
```

from_wif(*wif: str*) → *hdwallet.hdwallet.HDWallet*

Master from Wallet Important Format (WIF).

Parameters **wif** (*str*) – Wallet important format.

Returns HDWallet – Hierarchical Deterministic Wallet instance.

```
>>> from hdwallet import HDWallet
>>> from hdwallet.symbols import BTC
>>> hdwallet = HDWallet(symbol=BTC)
>>> hdwallet.from_wif(wif="KzsHWUJsrTWUUhBGPfMMxLLydiH7NhEn6z7mKHXD5qNkUWaC4TEn
↪")
<hdwallet.hdwallet.HDWallet object at 0x000001E8BFB98D60>
```

from_private_key(*private_key: str*) → *hdwallet.hdwallet.HDWallet*

Master from Private Key.

Parameters **private_key** (*str*) – Private key.

Returns HDWallet – Hierarchical Deterministic Wallet instance.

```
>>> from hdwallet import HDWallet
>>> from hdwallet.symbols import BTC
>>> hdwallet = HDWallet(symbol=BTC)
>>> hdwallet.from_private_key(private_key=
↪"6cd78b0d69eab1a47bfa53a52b9d8c4331e858b5d7a599270a95d9735fdb0b94")
<hdwallet.hdwallet.HDWallet object at 0x000001E8BFB98D60>
```

from_public_key(*public_key: str*) → *hdwallet.hdwallet.HDWallet*

Master from Public Key.

Parameters **public_key** (*str*) – Public key.

Returns HDWallet – Hierarchical Deterministic Wallet instance.

```
>>> from hdwallet import HDWallet
>>> from hdwallet.symbols import BTC
>>> hdwallet = HDWallet(symbol=BTC)
>>> hdwallet.from_public_key(public_key=
↪"02f93f58b97c3bb616645c3dda256ec946d87c45baf531984c022dd0fd1503b0a8")
<hdwallet.hdwallet.HDWallet object at 0x000001E8BFB98D60>
```

from_path(*path: Union[str,* hdwallet.derivations.Derivation*]*) → *hdwallet.hdwallet.HDWallet*

Derivation from Path.

Parameters **path** (*str*, Derivation) – Derivation path.

Returns HDWallet – Hierarchical Deterministic Wallet instance.

```
>>> from hdwallet import HDWallet
>>> from hdwallet.symbols import BTC
>>> hdwallet = HDWallet(symbol=BTC)
>>> hdwallet.from_xprivate_key(xprivate_key=
↪"xprv9s21ZrQH143K3xPGUzpogJeKtRdjHkK6muBJo8v7rEVRzT83xJgNcLpMoJXUf9wJFKfuHR4SGvfgdShh4t9Vmjj
↪")
>>> hdwallet.from_path(path="m/44'/0'/'0/0/0")
<hdwallet.hdwallet.HDWallet object at 0x000001E8BFB98D60>
```

from_index(*index: int*, *hardened: bool = False*) → *hdwallet.hdwallet.HDWallet*

Derivation from Index.

Parameters

- **index** (*int*) – Derivation index.
- **hardened** (*bool*) – Hardened address, default to `False`.

Returns HDWallet – Hierarchical Deterministic Wallet instance.

```
>>> from hdwallet import HDWallet
>>> from hdwallet.symbols import BTC
>>> hdwallet = HDWallet(symbol=BTC)
>>> hdwallet.from_xprivate_key(xprivate_key=
↪"xprv9s21ZrQH143K3xPGUzpogJeKtRdjHkK6muBJo8v7rEVRzT83xJgNcLpMoJXUf9wJFKfuHR4SGvfgdShh4t9Vmjj
↪")
>>> hdwallet.from_index(index=44, hardened=True)
>>> hdwallet.from_index(index=0, hardened=True)
>>> hdwallet.from_index(index=0, hardened=True)
>>> hdwallet.from_index(index=0)
>>> hdwallet.from_index(index=0)
<hdwallet.hdwallet.HDWallet object at 0x000001E8BFB98D60>
```

root_xprivate_key(*encoded: bool = True*) → Optional[str]

Get Root XPrivate Key.

Parameters **encoded** (*bool*) – Encoded root xprivate key, default to `True`.

Returns str – Root XPrivate Key.

```
>>> from hdwallet import HDWallet
>>> from hdwallet.symbols import BTC
>>> hdwallet = HDWallet(symbol=BTC)
>>> hdwallet.from_entropy(entropy="ee535b143b0d9d1f87546f9df0d06b1a")
>>> hdwallet.from_path(path="m/44'/0'/'0/0/0")
>>> hdwallet.root_xprivate_key()

↪"xprv9s21ZrQH143K3xPGUzpogJeKtRdjHkK6muBJo8v7rEVRzT83xJgNcLpMoJXUf9wJFKfuHR4SGvfgdShh4t9Vmjj
↪"
```

root_xpublic_key(*encoded: bool = True*) → Optional[str]

Get Root XPublic Key.

Parameters **encoded** (*bool*) – Encoded root xpublic key, default to `True`.

Returns str – Root XPublic Key.

```
>>> from hdwallet import HDWallet
>>> from hdwallet.symbols import BTC
>>> hdwallet = HDWallet(symbol=BTC)
>>> hdwallet.from_entropy(entropy="ee535b143b0d9d1f87546f9df0d06b1a")
>>> hdwallet.from_path(path="m/44'/0'/'0/0/0")
>>> hdwallet.root_xpublic_key()

↪"xpub661MyMwAqRbcGSTjb2Mp3Sb4STUDhD2x986ubXKjQa2QsFTCVqzdA98qeZjcncHT1AaZcMSjiP1HJ16jH97q72Ru
↪"
```

xprivate_key(*encoded=True*) → Optional[str]

Get XPrivate Key.

Parameters **encoded** (*bool*) – Encoded xprivate key, default to `True`.

Returns str – Root XPrivate Key.

```
>>> from hdwallet import HDWallet
>>> from hdwallet.symbols import BTC
>>> hdwallet = HDWallet(symbol=BTC)
>>> hdwallet.from_entropy(entropy="ee535b143b0d9d1f87546f9df0d06b1a")
>>> hdwallet.from_path(path="m/44'/0'/'0/0/0")
>>> hdwallet.xprivate_key()

↪"xprvA3BYGWQ9FmhyaNRRXB2f1LphNPnaY9T6gngw4BaTbkFtscSH4RCuJhgWUSKs9S6ciGioHd4TX4UeyUg53MkfN9X
↪"
```

xpublic_key(*encoded: bool = True*) → Optional[str]

Get XPublic Key.

Parameters **encoded** (*bool*) – Encoded xpublic key, default to `True`.

Returns str – XPublic Key.

```
>>> from hdwallet import HDWallet
>>> from hdwallet.symbols import BTC
>>> hdwallet = HDWallet(symbol=BTC)
>>> hdwallet.from_entropy(entropy="ee535b143b0d9d1f87546f9df0d06b1a")
>>> hdwallet.from_path(path="m/44'/0'/'0/0/0")
>>> hdwallet.xpublic_key()

↪"xpub6GAtg1w369GGnrVtdCZfNUmRvRd4wcAx41cXrZz5A5nskQmRbxX9rVzzKiRU4JruirBrfm4KQXNSU7GfqL1tzZWp
↪"
```

clean_derivation() → *hdwallet.hdwallet.HDWallet*

Clean derivation Path or Indexes.

Returns HDWallet – Hierarchical Deterministic Wallet instance.

```
>>> from hdwallet import HDWallet
>>> from hdwallet.symbols import BTC
>>> hdwallet = HDWallet(symbol=BTC)
>>> hdwallet.from_xprivate_key(xprivate_key=
↪"xprv9s21ZrQH143K3xPGUzpogJeKtRdjHkK6muBJo8v7rEVRzT83xJgNcLpMoJXUf9wJFKfuHR4SGvfgdShh4t9Vmjj
↪")
```

(continues on next page)

(continued from previous page)

```
>>> hdwallet.from_path(path="m/44'/0'/'0/0/0")
>>> hdwallet.path()
"m/44'/0'/'0/0/0"
>>> hdwallet.clean_derivation()
<hdwallet.hdwallet.HDWallet object at 0x000001E8BFB98D60>
>>> hdwallet.path()
None
```

uncompressed(*compressed: Optional[str] = None*) → str

Get Uncommpresed Public Key.

Parameters **compressed** (*str*) – Compressed public key, default to None.

Returns str – Uncommpresed public key.

```
>>> from hdwallet import HDWallet
>>> from hdwallet.symbols import BTC
>>> hdwallet = HDWallet(symbol=BTC)
>>> hdwallet.from_mnemonic(mnemonic="venture fitness paper little blush april␣
↪rigid where find volcano fetch crack label polar dash")
>>> hdwallet.from_path(path="m/44'/0'/0'/0/0")
>>> hdwallet.uncompressed()

↪"f93f58b97c3bb616645c3dda256ec946d87c45baf531984c022dd0fd1503b0a875f63285a539213ac241fc4a88e
↪"
```

compressed(*uncompressed: Optional[str] = None*) → str

Get Compresed Public Key.

Parameters **uncompressed** (*str*) – Uncompressed public key, default to None.

Returns str – Commpresed public key.

```
>>> from hdwallet import HDWallet
>>> from hdwallet.symbols import BTC
>>> hdwallet = HDWallet(symbol=BTC)
>>> hdwallet.from_mnemonic(mnemonic="venture fitness paper little blush april␣
↪rigid where find volcano fetch crack label polar dash")
>>> hdwallet.from_path(path="m/44'/0'/0'/0/0")
>>> hdwallet.compressed()
"02f93f58b97c3bb616645c3dda256ec946d87c45baf531984c022dd0fd1503b0a8"
```

private_key() → str

Get Private Key.

Returns str – Private key.

```
>>> from hdwallet import HDWallet
>>> from hdwallet.symbols import BTC
>>> hdwallet = HDWallet(symbol=BTC)
>>> hdwallet.from_mnemonic(mnemonic="venture fitness paper little blush april␣
↪rigid where find volcano fetch crack label polar dash")
>>> hdwallet.from_path(path="m/44'/0'/0'/0/0")
>>> hdwallet.private_key()
"6cd78b0d69eab1a47bfa53a52b9d8c4331e858b5d7a599270a95d9735fdb0b94"
```

public_key(*compressed: bool = True*, *private_key: Optional[str] = None*) → str

Get Public Key.

Parameters

- **compressed** (*bool*) – Compressed public key, default to `True`.
- **private_key** (*str*) – Private key hex string, default to `None`.

Returns str – Public key.

```
>>> from hdwallet import HDWallet
>>> from hdwallet.symbols import BTC
>>> hdwallet = HDWallet(symbol=BTC)
>>> hdwallet.from_mnemonic(mnemonic="venture fitness paper little blush april␣
↪rigid where find volcano fetch crack label polar dash")
>>> hdwallet.from_path(path="m/44'/0'/0'/0/0")
>>> hdwallet.public_key()
"02f93f58b97c3bb616645c3dda256ec946d87c45baf531984c022dd0fd1503b0a8"
```

strength() → Optional[int]

Get Entropy strength.

Returns int – Entropy strength.

```
>>> from hdwallet import HDWallet
>>> from hdwallet.symbols import BTC
>>> hdwallet = HDWallet(symbol=BTC)
>>> hdwallet.from_mnemonic(mnemonic="venture fitness paper little blush april␣
↪rigid where find volcano fetch crack label polar dash")
>>> hdwallet.strength()
160
```

entropy() → Optional[str]

Get Entropy hex string.

Returns str – Entropy hex string.

```
>>> from hdwallet import HDWallet
>>> from hdwallet.symbols import BTC
>>> hdwallet = HDWallet(symbol=BTC)
>>> hdwallet.from_mnemonic(mnemonic="venture fitness paper little blush april␣
↪rigid where find volcano fetch crack label polar dash")
>>> hdwallet.entropy()
"f24afe7fc1418815ee7fd256beb55518e7c34ecd"
```

mnemonic() → Optional[str]

Get Mnemonic words.

Returns str – Mnemonic words.

```
>>> from hdwallet import HDWallet
>>> from hdwallet.symbols import BTC
>>> hdwallet = HDWallet(symbol=BTC)
>>> hdwallet.from_mnemonic(mnemonic="venture fitness paper little blush april␣
↪rigid where find volcano fetch crack label polar dash")
```

(continues on next page)

(continued from previous page)

```
>>> hdwallet.mnemonic()
"venture fitness paper little blush april rigid where find volcano fetch crack␣
↪label polar dash"
```

passphrase() → Optional[str]

Get Mnemonic passphrase.

Returns str – Mnemonic passphrase.

```
>>> from hdwallet import HDWallet
>>> from hdwallet.symbols import BTC
>>> hdwallet = HDWallet(symbol=BTC)
>>> hdwallet.from_mnemonic(mnemonic="venture fitness paper little blush april␣
↪rigid where find volcano fetch crack label polar dash", passphrase="meherett")
>>> hdwallet.passphrase()
"meherett"
```

language() → Optional[str]

Get Mnemonic language.

Returns str – Mnemonic language.

```
>>> from hdwallet import HDWallet
>>> from hdwallet.symbols import BTC
>>> hdwallet = HDWallet(symbol=BTC)
>>> hdwallet.from_mnemonic(mnemonic="venture fitness paper little blush april␣
↪rigid where find volcano fetch crack label polar dash", passphrase="meherett")
>>> hdwallet.language()
"english"
```

cryptocurrency() → Optional[str]

Get Cryptocurrency name.

Returns str – Cryptocurrency name.

```
>>> from hdwallet import HDWallet
>>> from hdwallet.symbols import BTC
>>> hdwallet = HDWallet(symbol=BTC)
>>> hdwallet.from_mnemonic(mnemonic="venture fitness paper little blush april␣
↪rigid where find volcano fetch crack label polar dash", passphrase="meherett")
>>> hdwallet.cryptocurrency()
"Bitcoin"
```

symbol() → Optional[str]

Get Cryptocurrency symbol.

Returns str – Cryptocurrency symbol.

```
>>> from hdwallet import HDWallet
>>> from hdwallet.symbols import BTC
>>> hdwallet = HDWallet(symbol=BTC)
>>> hdwallet.from_mnemonic(mnemonic="venture fitness paper little blush april␣
↪rigid where find volcano fetch crack label polar dash", passphrase="meherett")
```

(continues on next page)

(continued from previous page)

```
>>> hdwallet.symbol()
"BTC"
```

semantic() → Optional[str]

Get Extended semantic.

Returns str – Extended semantic.

```
>>> from hdwallet import HDWallet
>>> from hdwallet.symbols import BTC
>>> hdwallet = HDWallet(symbol=BTC)
>>> hdwallet.from_mnemonic(mnemonic="venture fitness paper little blush april␣
↪rigid where find volcano fetch crack label polar dash", passphrase="meherett")
>>> hdwallet.semantic()
"p2pkh"
```

network() → Optional[str]

Get Cryptocurrency network type.

Returns str – Cryptocurrency network type.

```
>>> from hdwallet import HDWallet
>>> from hdwallet.symbols import BTC
>>> hdwallet = HDWallet(symbol=BTC)
>>> hdwallet.from_mnemonic(mnemonic="venture fitness paper little blush april␣
↪rigid where find volcano fetch crack label polar dash", passphrase="meherett")
>>> hdwallet.network()
"mainnet"
```

seed() → Optional[str]

Get Seed hex string.

Returns str – Seed hex string.

```
>>> from hdwallet import HDWallet
>>> from hdwallet.symbols import BTC
>>> hdwallet = HDWallet(symbol=BTC)
>>> hdwallet.from_mnemonic(mnemonic="venture fitness paper little blush april␣
↪rigid where find volcano fetch crack label polar dash", passphrase="meherett")
>>> hdwallet.from_path(path="m/44'/0'/0'/0/0")
>>> hdwallet.seed()

↪"8d5f4fe5b81a6a6a18b08603b6b3f59df9f4bbb25d10c55d23e0cbdc5ee385e5fddad9d7e6114f11afdec459283
↪"
```

path() → Optional[str]

Get Derivation path.

Returns str – Drivation path.

```
>>> from hdwallet import HDWallet
>>> from hdwallet.symbols import BTC
>>> hdwallet = HDWallet(symbol=BTC)
>>> hdwallet.from_mnemonic(mnemonic="venture fitness paper little blush april␣
↪rigid where find volcano fetch crack label polar dash", passphrase="meherett")
```

(continues on next page)

(continued from previous page)

```
>>> hdwallet.from_path(path="m/44'/0'/0'/0/0")
>>> hdwallet.path()
"m/44'/0'/0'/0/0"
```

chain_code() → Optional[str]

Get Chain code.

Returns str – Chain code.

```
>>> from hdwallet import HDWallet
>>> from hdwallet.symbols import BTC
>>> hdwallet = HDWallet(symbol=BTC)
>>> hdwallet.from_mnemonic(mnemonic="venture fitness paper little blush april␣
↪rigid where find volcano fetch crack label polar dash", passphrase="meherett")
>>> hdwallet.from_path(path="m/44'/0'/0'/0/0")
>>> hdwallet.chain_code()
"ed45793b944d1f22522f2d6f48c487029fae2cfcd999ed23087a148bcdda6314"
```

hash(*private_key: Optional[str] = None*)

Get Public Key Hash.

Returns str – Identifier.

```
>>> from hdwallet import HDWallet
>>> from hdwallet.symbols import BTC
>>> hdwallet = HDWallet(symbol=BTC)
>>> hdwallet.from_mnemonic(mnemonic="venture fitness paper little blush april␣
↪rigid where find volcano fetch crack label polar dash", passphrase="meherett")
>>> hdwallet.from_path(path="m/44'/0'/0'/0/0")
>>> hdwallet.hash()
"4d887566d408dfe5ea8090f2b716f9639523ca89"
```

finger_print() → str

Get Finger print.

Returns str – Finger print.

```
>>> from hdwallet import HDWallet
>>> from hdwallet.symbols import BTC
>>> hdwallet = HDWallet(symbol=BTC)
>>> hdwallet.from_mnemonic(mnemonic="venture fitness paper little blush april␣
↪rigid where find volcano fetch crack label polar dash", passphrase="meherett")
>>> hdwallet.from_path(path="m/44'/0'/0'/0/0")
>>> hdwallet.finger_print()
"4d887566"
```

p2pkh_address() → str

Get Pay to Public Key Hash (P2PKH) address.

Returns str – P2PKH address.

```
>>> from hdwallet import HDWallet
>>> from hdwallet.symbols import BTC
>>> hdwallet = HDWallet(symbol=BTC)
```

(continues on next page)

(continued from previous page)

```
>>> hdwallet.from_mnemonic(mnemonic="venture fitness paper little blush april␣
↪rigid where find volcano fetch crack label polar dash", passphrase="meherett")
>>> hdwallet.from_path(path="m/44'/0'/0'/0/0")
>>> hdwallet.p2pkh_address()
"184xW5gWDnhS7LriL2JAZs1XGTJjimz7pq"
```

p2sh_address() → str

Get Pay to Script Hash (P2SH) address.

Returns str – P2SH address.

```
>>> from hdwallet import HDWallet
>>> from hdwallet.symbols import BTC
>>> hdwallet = HDWallet(symbol=BTC)
>>> hdwallet.from_mnemonic(mnemonic="venture fitness paper little blush april␣
↪rigid where find volcano fetch crack label polar dash", passphrase="meherett")
>>> hdwallet.from_path(path="m/44'/0'/0'/0/0")
>>> hdwallet.p2sh_address()
"3Jp6ad4ErhibQmhSRfavbPRiUyg2xTTT4j"
```

p2wpkh_address() → Optional[str]

Get Pay to Witness Public Key Hash (P2WPKH) address.

Returns str – P2WPKH address.

```
>>> from hdwallet import HDWallet
>>> from hdwallet.symbols import BTC
>>> hdwallet = HDWallet(symbol=BTC)
>>> hdwallet.from_mnemonic(mnemonic="venture fitness paper little blush april␣
↪rigid where find volcano fetch crack label polar dash", passphrase="meherett")
>>> hdwallet.from_path(path="m/44'/0'/0'/0/0")
>>> hdwallet.p2wpkh_address()
"bc1qfky82ek5pr07t65qjretw9hevw2j8j5fdrn5hc"
```

p2wpkh_in_p2sh_address() → Optional[str]

Get P2WPKH nested in P2SH address.

Returns str – P2WPKH nested in P2SH address.

```
>>> from hdwallet import HDWallet
>>> from hdwallet.symbols import BTC
>>> hdwallet = HDWallet(symbol=BTC)
>>> hdwallet.from_mnemonic(mnemonic="venture fitness paper little blush april␣
↪rigid where find volcano fetch crack label polar dash", passphrase="meherett")
>>> hdwallet.from_path(path="m/44'/0'/0'/0/0")
>>> hdwallet.p2wpkh_in_p2sh_address()
"3CCrxPrHNa6ePbnB7qjh7S3vaPx9qiLc3e"
```

p2wsh_address() → Optional[str]

Get Pay to Witness Script Hash (P2WSH) address.

Returns str – P2WSH address.

```
>>> from hdwallet import HDWallet
>>> from hdwallet.symbols import BTC
>>> hdwallet = HDWallet(symbol=BTC)
>>> hdwallet.from_mnemonic(mnemonic="venture fitness paper little blush april␣
↪rigid where find volcano fetch crack label polar dash", passphrase="meherett")
>>> hdwallet.from_path(path="m/44'/0'/0'/0/0")
>>> hdwallet.p2wsh_address()
"bc1qaj2xa9j6eegcxls3y8p6erw6vdgdxynasrd4hl3xuctht5edu3msdeshgf"
```

p2wsh_in_p2sh_address() → Optional[str]

Get P2WSH nested in P2SH address.

Returns str – P2WSH nested in P2SH address.

```
>>> from hdwallet import HDWallet
>>> from hdwallet.symbols import BTC
>>> hdwallet = HDWallet(symbol=BTC)
>>> hdwallet.from_mnemonic(mnemonic="venture fitness paper little blush april␣
↪rigid where find volcano fetch crack label polar dash", passphrase="meherett")
>>> hdwallet.from_path(path="m/44'/0'/0'/0/0")
>>> hdwallet.p2wsh_in_p2sh_address()
"38YMonfh2yLFRViLrM2kdvZx8ctcp1vbbV"
```

wif() → Optional[str]

Get Wallet Important Format.

Returns str – Wallet Important Format.

```
>>> from hdwallet import HDWallet
>>> from hdwallet.symbols import BTC
>>> hdwallet = HDWallet(symbol=BTC)
>>> hdwallet.from_mnemonic(mnemonic="venture fitness paper little blush april␣
↪rigid where find volcano fetch crack label polar dash", passphrase="meherett")
>>> hdwallet.from_path(path="m/44'/0'/0'/0/0")
>>> hdwallet.wif()
"KzsHWUJsrTWUUhBGPfMMxLLydiH7NhEn6z7mKHXD5qNkUWaC4TEn"
```

dumps() → dict

Get All HDWallet imformations.

Returns dict – All HDWallet imformations.

```
>>> from hdwallet import HDWallet
>>> from hdwallet.symbols import BTC
>>> hdwallet = HDWallet(symbol=BTC)
>>> hdwallet.from_mnemonic(mnemonic="venture fitness paper little blush april␣
↪rigid where find volcano fetch crack label polar dash", passphrase="meherett")
>>> hdwallet.from_path(path="m/44'/0'/0'/0/0")
>>> hdwallet.dumps()
{'cryptocurrency': 'Bitcoin', 'symbol': 'BTC', 'network': 'mainnet', 'strength
↪': 160, 'entropy': 'f24afe7fc1418815ee7fd256beb55518e7c34ecd', 'mnemonic':
↪'venture fitness paper little blush april rigid where find volcano fetch␣
↪crack label polar dash', 'language': 'english', 'passphrase': None, 'seed':
↪'8d5f4fe5b81a6a6a18b08603b6b3f59df9f4bbb25d10c55d23e0cbdc5ee385e5fddad9d7e6114f11afdec459283
↪', 'root_xprivate_key':
↪'xprv9s21ZrQH143K3xPGUzpogJeKtRdjHkK6muBJo8v7rEVRzT83xJgNcLpMoJXUf9wJFKfuHR4SGvfgdShh4t9Vmjj
↪', 'root_xpublic_key':
↪'xpub661MyMwAqRbcGSTjb2Mp3Sb4STUDhD2x986ubXKjQa2QsFTCVqzdA98[illegible]P1HJ16jH97q72R
↪', 'xprivate_key':
↪'xprvA3BYGWQ9FmhyaNRRXB2f1LphNPnaY9T6gngw4BaTbkFtscSH4RCuJhgWUSKs9S6ciGioHd4TX4UeyUg53MkfN9X
↪', 'xpublic_key':
```

(continues on next page)

(continued from previous page)

5.1 BIP32HDWallet

Class BIP32 Hierarchical Deterministic Wallet

class hdwallet.hdwallet.**BIP32HDWallet**(*symbol: str = 'BTC', cryptocurrency: Optional[Any] = None, purpose: Union[int, Tuple[int, bool]] = 0, coin_type: Union[int, Tuple[int, bool]] = 0, account: Union[int, Tuple[int, bool]] = 0, change: bool = False, address: Union[int, Tuple[int, bool]] = 0*)

BIP32 Hierarchical Deterministic Wallet

Parameters

- **symbol** (*str*) – Cryptocurrency symbol, defaults to BTC.
- **cryptocurrency** (*Cryptocurrency*) – Cryptocurrency instance, default to None.
- **purpose** (*int, tuple*) – Purpose index, default to 0.
- **coin_type** (*int, tuple*) – Coin type, default to 0.
- **account** (*int, tuple*) – Account index, default to 0.
- **change** (*bool*) – Change addresses, default to False.
- **address** (*int, tuple*) – Address index, default to 0.

Returns BIP32HDWallet – BIP32 Hierarchical Deterministic Wallet instance.

```
>>> from hdwallet import BIP32HDWallet
>>> from hdwallet.cryptocurrencies import QtumMainnet
>>> bip32_hdwallet: BIP32HDWallet = BIP32HDWallet(cryptocurrency=QtumMainnet,␣
↪purpose=0, coin_type=0, account=0, change=False, address=0)
<hdwallet.hdwallet.BIP32HDWallet object at 0x000001EBC58E9F70>
```

address() → str

Get Pay to Public Key Hash (P2PKH) address.

Returns str – P2PKH address.

```
>>> from hdwallet import BIP32HDWallet
>>> from hdwallet.symbols import BTC
>>> bip32_hdwallet: BIP32HDWallet = BIP32HDWallet(symbol=BTC, purpose=44, coin_
↪type=0, account=0, change=False, address=0)
>>> bip32_hdwallet.from_mnemonic(mnemonic="venture fitness paper little blush␣
↪april rigid where find volcano fetch crack label polar dash", passphrase=
↪"meherett")
>>> bip32_hdwallet.address()
"184xW5gWDnhS7LriL2JAZs1XGTJjimz7pq"
```

5.2 BIP44HDWallet

Class BIP44 Hierarchical Deterministic Wallet

class hdwallet.hdwallet.**BIP44HDWallet**(*symbol: str = 'BTC', cryptocurrency: Optional[Any] = None, account: Union[int, Tuple[int, bool]] = 0, change: bool = False, address: Union[int, Tuple[int, bool]] = 0*)

BIP44 Hierarchical Deterministic Wallet

Parameters

- **symbol** (*str*) – Cryptocurrency symbol, defaults to BTC.
- **cryptocurrency** (*Cryptocurrency*) – Cryptocurrency instance, default to None.
- **account** (*int, tuple*) – Account index, default to 0.
- **change** (*bool*) – Change addresses, default to False.
- **address** (*int, tuple*) – Address index, default to 0.

Returns BIP44HDWallet – BIP44 Hierarchical Deterministic Wallet instance.

```
>>> from hdwallet import BIP44HDWallet
>>> from hdwallet.cryptocurrencies import QtumMainnet
>>> bip44_hdwallet: BIP44HDWallet = BIP44HDWallet(cryptocurrency=QtumMainnet,␣
↪account=0, change=False, address=0)
<hdwallet.hdwallet.BIP44HDWallet object at 0x000001EBC58E9F70>
```

address() → str

Get Pay to Public Key Hash (P2PKH) address.

Returns str – P2PKH address.

```
>>> from hdwallet import BIP44HDWallet
>>> from hdwallet.symbols import BTC
>>> bip44_hdwallet: BIP44HDWallet = BIP44HDWallet(symbol=BTC, account=0,␣
↪change=False, address=0)
>>> bip44_hdwallet.from_mnemonic(mnemonic="venture fitness paper little blush␣
↪april rigid where find volcano fetch crack label polar dash", passphrase=
↪"meherett")
>>> bip44_hdwallet.address()
"184xW5gWDnhS7LriL2JAZs1XGTJjimz7pq"
```

5.3 BIP49HDWallet

Class BIP49 Hierarchical Deterministic Wallet

class hdwallet.hdwallet.**BIP49HDWallet**(*symbol: str = 'BTC', cryptocurrency: Optional[Any] = None, account: Union[int, Tuple[int, bool]] = 0, change: bool = False, address: Union[int, Tuple[int, bool]] = 0*)

BIP49 Hierarchical Deterministic Wallet

Parameters

- **symbol** (*str*) – Cryptocurrency symbol, defaults to BTC.

- **cryptocurrency** (*Cryptocurrency*) – Cryptocurrency instance, default to None.
- **account** (*int, tuple*) – Account index, default to 0.
- **change** (*bool*) – Change addresses, default to False.
- **address** (*int, tuple*) – Address index, default to 0.

Returns BIP49HDWallet – BIP49 Hierarchical Deterministic Wallet instance.

```
>>> from hdwallet import BIP49HDWallet
>>> from hdwallet.cryptocurrencies import QtumMainnet
>>> bip49_hdwallet: BIP49HDWallet = BIP49HDWallet(cryptocurrency=QtumMainnet,␣
↪account=0, change=False, address=0)
<hdwallet.hdwallet.BIP49HDWallet object at 0x000001EBC58E9F70>
```

address() → str

Get P2WPKH nested in P2SH address.

Returns str – P2PKH In P2SH address.

```
>>> from hdwallet import BIP49HDWallet
>>> from hdwallet.symbols import BTC
>>> bip49_hdwallet: BIP49HDWallet = BIP49HDWallet(symbol=BTC, account=0,␣
↪change=False, address=0)
>>> bip49_hdwallet.from_mnemonic(mnemonic="venture fitness paper little blush␣
↪april rigid where find volcano fetch crack label polar dash", passphrase=
↪"meherett")
>>> bip49_hdwallet.address()
"3HtjZPoiUh9DA3kzQL9XZ29aFdCzouWB6T"
```

5.4 BIP84HDWallet

Class BIP84 Hierarchical Deterministic Wallet

class hdwallet.hdwallet.**BIP84HDWallet**(*symbol: str = 'BTC', cryptocurrency: Optional[Any] = None, account: Union[int, Tuple[int, bool]] = 0, change: bool = False, address: Union[int, Tuple[int, bool]] = 0*)

BIP84 Hierarchical Deterministic Wallet

Parameters

- **symbol** (*str*) – Cryptocurrency symbol, defaults to BTC.
- **cryptocurrency** (*Cryptocurrency*) – Cryptocurrency instance, default to None.
- **account** (*int, tuple*) – Account index, default to 0.
- **change** (*bool*) – Change addresses, default to False.
- **address** (*int, tuple*) – Address index, default to 0.

Returns BIP84HDWallet – BIP84 Hierarchical Deterministic Wallet instance.

```
>>> from hdwallet import BIP84HDWallet
>>> from hdwallet.cryptocurrencies import QtumMainnet
```

(continues on next page)

(continued from previous page)

```
>>> bip84_hdwallet: BIP84HDWallet = BIP84HDWallet(cryptocurrency=QtumMainnet,␣
↪account=0, change=False, address=0)
<hdwallet.hdwallet.BIP84HDWallet object at 0x000001EBC58E9F70>
```

address() → str

Get Pay to Witness Public Key Hash (P2WPKH) address.

Returns str – Pay to Witness Public Key Hash (P2WPKH) address.

```
>>> from hdwallet import BIP84HDWallet
>>> from hdwallet.symbols import BTC
>>> bip84_hdwallet: BIP84HDWallet = BIP84HDWallet(symbol=BTC, account=0,␣
↪change=False, address=0)
>>> bip84_hdwallet.from_mnemonic(mnemonic="venture fitness paper little blush␣
↪april rigid where find volcano fetch crack label polar dash", passphrase=
↪"meherett")
>>> bip84_hdwallet.address()
"bc1qs95czhkhawdq958gjscrw4mh6amu2ysx20w86d"
```

5.5 BIP141HDWallet

Class BIP141 Hierarchical Deterministic Wallet

class hdwallet.hdwallet.**BIP141HDWallet**(*symbol: str = 'BTC'*, *cryptocurrency: Optional[Any] = None*, *path: Optional[Union[str,* hdwallet.derivations.Derivation*]] = None*, *semantic: str = 'p2wpkh'*)

BIP141 Hierarchical Deterministic Wallet

Parameters

- **symbol** (*str*) – Cryptocurrency symbol, defaults to BTC.
- **cryptocurrency** (*Cryptocurrency*) – Cryptocurrency instance, defaults to None.
- **path** (*str*) – Derivation path.
- **semantic** (*str*) – Extended semantic, defaults to P2WPKH.

Returns BIP141HDWallet – BIP141 Hierarchical Deterministic Wallet instance.

```
>>> from hdwallet import BIP141HDWallet
>>> from hdwallet.cryptocurrencies import QtumMainnet
>>> bip141_hdwallet: BIP141HDWallet = BIP141HDWallet(cryptocurrency=QtumMainnet,␣
↪path="m/0/0", semantic="p2wpkh")
<hdwallet.hdwallet.BIP141HDWallet object at 0x000001EBC58E9F70>
```

address() → str

Get P2WPKH, P2WPKH_IN_P2SH, P2WSH or P2WPKH_IN_P2SH addresses by semantic.

Returns str – P2WPKH, P2WPKH_IN_P2SH, P2WSH or P2WPKH_IN_P2SH addresses.

```
>>> from hdwallet import BIP141HDWallet
>>> from hdwallet.symbols import BTC
>>> bip141_hdwallet: BIP141HDWallet = BIP141HDWallet(symbol=BTC, path="m/44'/0'/
↪0'/0/0", semantic="p2wsh")
```

(continues on next page)

(continued from previous page)

```
>>> bip141_hdwallet.from_mnemonic(mnemonic="venture fitness paper little blush␣
↪april rigid where find volcano fetch crack label polar dash", passphrase=
↪"meherett")
>>> bip141_hdwallet.address()
"bc1qaj2xa9j6eegcxls3y8p6erw6vdgdxynasrd4hl3xuctht5edu3msdeshgf"
```

CHAPTER

SIX

DERIVATION

class hdwallet.derivations.**Derivation**(*path: Optional[str] = None*, *semantic: str = 'p2pkh'*)

Hierarchical Deterministic Wallet Derivation's

Parameters

- **path** (*str*) – Derivation path.
- **semantic** (*str*) – Extended semantic, defaults to P2PKH.

Returns Derivation – Derivation instance.

```
>>> from hdwallet.derivations import Derivation
>>> Derivation()
<hdwallet.derivations.Derivation object at 0x000001EBC58E9F70>
>>> str(Derivation())
""
>>> str(Derivation(path="m/44'/0'/0'/0/0", semantic="p2pkh"))
"m/44'/0'/0'/0/0"
```

Note: Do not forget all derivation paths are start swith 'm/' prefix.

classmethod from_path(*path: str*) → *hdwallet.derivations.Derivation*

Derivation from path.

Parameters **path** (*str*, Derivation) – Derivation path.

Returns Derivation – Derivation instance.

```
>>> from hdwallet.derivations import Derivation
>>> derivation = Derivation()
>>> derivation.from_path(path="m/44'/0'/'0/0/0")
<hdwallet.derivation.Derivation object at 0x000001E8BFB98D60>
```

from_index(*index: int*, *hardened: bool = False*) → *hdwallet.derivations.Derivation*

Derivation from path.

Parameters

- **index** (*int*) – Derivation index.
- **hardened** (*bool*) – Hardened address, default to False.

Returns Derivation – Derivation instance.

```
>>> from hdwallet.derivations import Derivation
>>> derivation = Derivation()
>>> derivation.from_index(index=44, hardened=True)
>>> derivation.from_index(index=0, hardened=True)
>>> derivation.from_index(index=0, hardened=True)
>>> derivation.from_index(index=0)
>>> derivation.from_index(index=0)
<hdwallet.derivation.Derivation object at 0x000001E8BFB98D60>
```

clean_derivation() → *hdwallet.derivations.Derivation*

Clean derivation path or indexes.

Returns Derivation – Derivation instance.

```
>>> from hdwallet.derivations import Derivation
>>> derivation = Derivation()
>>> derivation.from_path(path="m/44'/0'/'0/0/0")
>>> str(derivation)
"m/44'/0'/'0/0/0"
>>> derivation.clean_derivation()
<hdwallet.wallet.HDWallet object at 0x000001E8BFB98D60>
>>> str(derivation)
""
```

6.1 BIP32Derivation

class hdwallet.derivations.**BIP32Derivation**(*cryptocurrency: Optional[Any] = None*, *purpose: Union[int, Tuple[int, bool]] = 0*, *coin_type: Union[int, Tuple[int, bool]] = 0*, *account: Union[int, Tuple[int, bool]] = 0*, *change: bool = False*, *address: Union[int, Tuple[int, bool]] = 0*)

Hierarchical Deterministic Wallet BIP32 Derivation

Parameters

- **cryptocurrency** (*Cryptocurrency*) – Cryptocurrency instance, default to None.
- **purpose** (*int, tuple*) – Purpose index, default to 0.
- **coin_type** (*int, tuple*) – Coin type, default to 0.
- **account** (*int, tuple*) – Account index, default to 0.
- **change** (*bool*) – Change addresses, default to False.
- **address** (*int, tuple*) – Address index, default to 0.

Returns BIP32Derivation – BIP32Derivation instance.

```
>>> from hdwallet.derivations import BIP32Derivation
>>> from hdwallet.cryptocurrencies import BitcoinMainnet
>>> BIP32Derivation(cryptocurrency=BitcoinMainnet)
<hdwallet.derivations.Derivation object at 0x000001EBC58E9F70>
>>> str(BIP32Derivation(cryptocurrency=BitcoinMainnet))
"m/0'/0'/0'/0/0"
```

from_purpose(*purpose: int*, *hardened: bool = True*) → *hdwallet.derivations.BIP32Derivation*

Derivation from purpose index.

Parameters

- **purpose** (*int*) – Purpose index.
- **hardened** (*bool*) – Hardened, default to True.

Returns BIP32Derivation – BIP32Derivation instance.

```
>>> from hdwallet.derivations import BIP32Derivation
>>> bip32_derivation = BIP32Derivation()
>>> bip32_derivation.from_purpose(purpose=0, hardened=True)
<hdwallet.derivation.BIP32Derivation object at 0x000001E8BFB98D60>
```

from_coin_type(*coin_type: int*, *hardened: bool = True*) → *hdwallet.derivations.BIP32Derivation*

Derivation from Coin Type index.

Parameters

- **coin_type** (*int*) – Coin type index.
- **hardened** (*bool*) – Hardened, default to True.

Returns BIP32Derivation – BIP32Derivation instance.

```
>>> from hdwallet.derivations import BIP32Derivation
>>> bip32_derivation = BIP32Derivation()
>>> bip32_derivation.from_coin_type(coin_type=56, hardened=True)
<hdwallet.derivation.BIP32Derivation object at 0x000001E8BFB98D60>
```

from_account(*account: int*, *hardened: bool = True*) → *hdwallet.derivations.BIP32Derivation*

Derivation from Account index.

Parameters

- **account** (*int*) – Coin type index.
- **hardened** (*bool*) – Hardened, default to True.

Returns BIP32Derivation – BIP32Derivation instance.

```
>>> from hdwallet.derivations import BIP32Derivation
>>> bip32_derivation = BIP32Derivation()
>>> bip32_derivation.from_account(account=1, hardened=True)
<hdwallet.derivation.BIP32Derivation object at 0x000001E8BFB98D60>
```

from_change(*change: bool*) → *hdwallet.derivations.BIP32Derivation*

Derivation from external Change.

Parameters **change** (*bool*) – External chnage.

Returns BIP32Derivation – BIP32Derivation instance.

```
>>> from hdwallet.derivations import BIP32Derivation
>>> bip32_derivation = BIP32Derivation()
>>> bip32_derivation.from_account(change=True)
<hdwallet.derivation.BIP32Derivation object at 0x000001E8BFB98D60>
```

from_address(*address: int*, *hardened: bool = False*) → *hdwallet.derivations.BIP32Derivation*

Derivation from Address index.

Parameters

- **address** (*int*) – Address index.
- **hardened** (*bool*) – Hardened, default to True.

Returns BIP32Derivation – BIP32Derivation instance.

```
>>> from hdwallet.derivations import BIP32Derivation
>>> bip32_derivation = BIP32Derivation()
>>> bip32_derivation.from_address(address=1, hardened=True)
<hdwallet.derivation.BIP32Derivation object at 0x000001E8BFB98D60>
```

clean_derivation() → *hdwallet.derivations.BIP32Derivation*

Clean derivation path or indexes.

Returns Derivation – Derivation instance.

```
>>> from hdwallet.derivations import Derivation
>>> derivation = Derivation()
>>> derivation.from_path(path="m/44'/0'/'0/0/0")
>>> str(derivation)
"m/44'/0'/'0/0/0"
>>> derivation.clean_derivation()
<hdwallet.wallet.HDWallet object at 0x000001E8BFB98D60>
>>> str(derivation)
""
```

purpose() → str

Gey Purpose index.

Returns str – Purpose index.

```
>>> from hdwallet.derivations import BIP32Derivation
>>> bip32_derivation = BIP32Derivation()
>>> bip32_derivation.from_purpose(purpose=0, hardened=True)
>>> bip32_derivation.purpose()
"0'"
```

coin_type() → str

Gey Coin Type index.

Returns str – Coin Type index.

```
>>> from hdwallet.derivations import BIP32Derivation
>>> bip32_derivation = BIP32Derivation()
>>> bip32_derivation.coin_type(coin_type=15, hardened=True)
>>> bip32_derivation.coin_type()
"15'"
```

account() → str

Gey Account index.

Returns str – Account index.

```
>>> from hdwallet.derivations import BIP32Derivation
>>> bip32_derivation = BIP32Derivation()
>>> bip32_derivation.from_account(account=1, hardened=True)
>>> bip32_derivation.account()
"1'"
```

change(*number: bool = False*) → Union[str, bool]

Gey external Change.

Parameters **number** (*bool*) – Return type, default to `False`.

Returns str – External change index.

```
>>> from hdwallet.derivations import BIP32Derivation
>>> bip32_derivation = BIP32Derivation()
>>> bip32_derivation.from_change(change=True)
>>> bip32_derivation.change(number=True)
"1"
>>> bip32_derivation.change(number=False)
True
```

address() → str

Gey Address index.

Returns str – Address index.

```
>>> from hdwallet.derivations import BIP32Derivation
>>> bip32_derivation = BIP32Derivation()
>>> bip32_derivation.from_address(address=1, hardened=True)
>>> bip32_derivation.address()
"1"
```

6.2 BIP44Derivation

class hdwallet.derivations.**BIP44Derivation**(*cryptocurrency: Any*, *account: Union[int, Tuple[int, bool]] = 0*, *change: bool = False*, *address: Union[int, Tuple[int, bool]] = 0*)

Hierarchical Deterministic Wallet BIP44 Derivation

Parameters

- **cryptocurrency** (*Cryptocurrency*) – Cryptocurrency instance.
- **account** (*int, tuple*) – Account index, default to `0`.
- **change** (*bool*) – Change addresses, default to `False`.
- **address** (*int, tuple*) – Address index, default to `0`.

Returns BIP44Derivation – BIP44Derivation instance.

```
>>> from hdwallet.derivations import BIP44Derivation
>>> from hdwallet.cryptocurrencies import BitcoinMainnet
>>> BIP44Derivation(cryptocurrency=BitcoinMainnet)
<hdwallet.derivations.Derivation object at 0x000001EBC58E9F70>
```

(continues on next page)

(continued from previous page)

```
>>> str(BIP44Derivation(cryptocurrency=BitcoinMainnet))
"m/44'/0'/0'/0/0"
```

6.3 BIP49Derivation

class hdwallet.derivations.**BIP49Derivation**(*cryptocurrency: Any*, *account: Union[int, Tuple[int, bool]] = 0*, *change: bool = False*, *address: Union[int, Tuple[int, bool]] = 0*)

Hierarchical Deterministic Wallet BIP49 Derivation

Parameters

- **cryptocurrency** (*Cryptocurrency*) – Cryptocurrency instance.
- **account** (*int, tuple*) – Account index, default to `0`.
- **change** (*bool*) – Change addresses, default to `False`.
- **address** (*int, tuple*) – Address index, default to `0`.

Returns BIP49Derivation – BIP49Derivation instance.

```
>>> from hdwallet.derivations import BIP49Derivation
>>> from hdwallet.cryptocurrencies import BitcoinMainnet
>>> BIP49Derivation(cryptocurrency=BitcoinMainnet)
<hdwallet.derivations.Derivation object at 0x000001EBC58E9F70>
>>> str(BIP49Derivation(cryptocurrency=BitcoinMainnet))
"m/49'/0'/0'/0/0"
```

6.4 BIP84Derivation

class hdwallet.derivations.**BIP84Derivation**(*cryptocurrency: Any*, *account: Union[int, Tuple[int, bool]] = 0*, *change: bool = False*, *address: Union[int, Tuple[int, bool]] = 0*)

Hierarchical Deterministic Wallet BIP84 Derivation

Parameters

- **cryptocurrency** (*Cryptocurrency*) – Cryptocurrency instance.
- **account** (*int, tuple*) – Account index, default to `0`.
- **change** (*bool*) – Change addresses, default to `False`.
- **address** (*int, tuple*) – Address index, default to `0`.

Returns BIP84Derivation – BIP84Derivation instance.

```
>>> from hdwallet.derivations import BIP84Derivation
>>> from hdwallet.cryptocurrencies import BitcoinMainnet
>>> BIP84Derivation(cryptocurrency=BitcoinMainnet)
<hdwallet.derivations.Derivation object at 0x000001EBC58E9F70>
>>> str(BIP84Derivation(cryptocurrency=BitcoinMainnet))
"m/84'/0'/0'/0/0"
```

6.5 BIP141Derivation

class hdwallet.derivations.**BIP141Derivation**(*cryptocurrency: Any*, *path: Optional[Union[str, hdwallet.derivations.Derivation]] = None*, *semantic: str = 'p2wpkh'*)

Hierarchical Deterministic Wallet BIP141 Derivation

Parameters

- **path** (*str*) – Derivation path, default to None.
- **semantic** (*str*) – Extended semantic, defaults to P2WPKH.

Returns BIP141Derivation – BIP141Derivation instance.

```
>>> from hdwallet.derivations import BIP141Derivation
>>> from hdwallet.cryptocurrencies import BitcoinMainnet
>>> BIP141Derivation(cryptocurrency=BitcoinMainnet)
<hdwallet.derivations.Derivation object at 0x000001EBC58E9F70>
>>> str(BIP141Derivation(cryptocurrency=BitcoinMainnet))
"m/44'/0'/0'/0/0"
```

CHAPTER

SEVEN

UTILS

hdwallet.utils.**generate_passphrase**(*length: int = 32*) → str

Generate entropy hex string.

Parameters **length** (*int*) – Passphrase length, default to 32.

Returns str – Passphrase hex string.

```
>>> from hdwallet.utils import generate_passphrase
>>> generate_passphrase(length=32)
"N39rPfa3QvF2Tm2nPyoBpXNiBFXJywTz"
```

hdwallet.utils.**generate_entropy**(*strength: int = 128*) → str

Generate entropy hex string.

Parameters **strength** (*int*) – Entropy strength, default to 128.

Returns str – Entropy hex string.

```
>>> from hdwallet.utils import generate_entropy
>>> generate_entropy(strength=128)
"ee535b143b0d9d1f87546f9df0d06b1a"
```

hdwallet.utils.**generate_mnemonic**(*language: str = 'english'*, *strength: int = 128*) → str

Generate mnemonic words.

Parameters

- **language** (*str*) – Mnemonic language, default to english.
- **strength** (*int*) – Entropy strength, default to 128.

Returns str – Mnemonic words.

```
>>> from hdwallet.utils import generate_mnemonic
>>> generate_mnemonic(language="french")
"sceptre capter sequence girafe absolu relatif fleur zoologie muscle sirop saboter␣
↪parure"
```

hdwallet.utils.**is_entropy**(*entropy: str*) → bool

Check entropy hex string.

Parameters **entropy** (*str*) – Mnemonic words.

Returns bool – Entropy valid/invalid.

```
>>> from hdwallet.utils import is_entropy
>>> is_entropy(entropy="ee535b143b0d9d1f87546f9df0d06b1a")
True
```

hdwallet.utils.**is_mnemonic**(*mnemonic: str*, *language: Optional[str] = None*) → bool

Check mnemonic words.

Parameters

- **mnemonic** (*str*) – Mnemonic words.
- **language** (*str*) – Mnemonic language, default to None.

Returns bool – Mnemonic valid/invalid.

```
>>> from hdwallet.utils import is_mnemonic
>>> is_mnemonic(mnemonic="sceptre capter sequence girafe absolu relatif fleur␣
↪zoologie muscle sirop saboter parure")
True
```

hdwallet.utils.**get_entropy_strength**(*entropy: str*) → int

Get entropy strength.

Parameters **entropy** (*str*) – Entropy hex string.

Returns int – Entropy strength.

```
>>> from hdwallet.utils import get_entropy_strength
>>> get_entropy_strength(entropy="ee535b143b0d9d1f87546f9df0d06b1a")
128
```

hdwallet.utils.**get_mnemonic_strength**(*mnemonic: str*, *language: Optional[str] = None*) → int

Get mnemonic strength.

Parameters

- **mnemonic** (*str*) – Mnemonic words.
- **language** (*str*) – Mnemonic language, default to None.

Returns int – Mnemonic strength.

```
>>> from hdwallet.utils import get_mnemonic_strength
>>> get_mnemonic_strength(mnemonic="sceptre capter sequence girafe absolu relatif␣
↪fleur zoologie muscle sirop saboter parure")
128
```

hdwallet.utils.**get_mnemonic_language**(*mnemonic: str*) → str

Get mnemonic language.

Parameters **mnemonic** (*str*) – Mnemonic words.

Returns str – Mnemonic language.

```
>>> from hdwallet.utils import get_mnemonic_language
>>> get_mnemonic_language(mnemonic="sceptre capter sequence girafe absolu relatif␣
↪fleur zoologie muscle sirop saboter parure")
"french"
```

hdwallet.utils.**entropy_to_mnemonic**(*entropy: str*, *language: str = 'english'*) → str

Get mnemonic from entropy hex string.

Parameters

- **entropy** (*str*) – Entropy hex string.
- **language** (*str*) – Mnemonic language, default to english.

Returns str – Mnemonic words.

```
>>> from hdwallet.utils import entropy_to_mnemonic
>>> entropy_to_mnemonic(entropy="ee535b143b0d9d1f87546f9df0d06b1a", language="korean
↪")
"            "
```

hdwallet.utils.**mnemonic_to_entropy**(*mnemonic: str*, *language: Optional[str] = None*) → str

Get entropy from mnemonic words.

Parameters

- **mnemonic** (*str*) – Mnemonic words.
- **language** (*str*) – Mnemonic language, default to english.

Returns str – Enropy hex string.

```
>>> from hdwallet.utils import mnemonic_to_entropy
>>> mnemonic_to_entropy(mnemonic="           ", language="korean")
"ee535b143b0d9d1f87546f9df0d06b1a"
```

PYTHON MODULE INDEX

h

hdwallet.utils, 41

INDEX

A

account() (*hdwallet.derivations.BIP32Derivation method*), 36
address() (*hdwallet.derivations.BIP32Derivation method*), 37
address() (*hdwallet.hdwallet.BIP141HDWallet method*), 30
address() (*hdwallet.hdwallet.BIP32HDWallet method*), 27
address() (*hdwallet.hdwallet.BIP44HDWallet method*), 28
address() (*hdwallet.hdwallet.BIP49HDWallet method*), 29
address() (*hdwallet.hdwallet.BIP84HDWallet method*), 30

B

BIP141Derivation (*class in hdwallet.derivations*), 39
BIP141HDWallet (*class in hdwallet.hdwallet*), 30
BIP32Derivation (*class in hdwallet.derivations*), 34
BIP32HDWallet (*class in hdwallet.hdwallet*), 27
BIP44Derivation (*class in hdwallet.derivations*), 37
BIP44HDWallet (*class in hdwallet.hdwallet*), 28
BIP49Derivation (*class in hdwallet.derivations*), 38
BIP49HDWallet (*class in hdwallet.hdwallet*), 28
BIP84Derivation (*class in hdwallet.derivations*), 38
BIP84HDWallet (*class in hdwallet.hdwallet*), 29

C

chain_code() (*hdwallet.hdwallet.HDWallet method*), 24
change() (*hdwallet.derivations.BIP32Derivation method*), 37
clean_derivation() (*hdwallet.derivations.BIP32Derivation method*), 36
clean_derivation() (*hdwallet.derivations.Derivation method*), 34
clean_derivation() (*hdwallet.hdwallet.HDWallet method*), 19
coin_type() (*hdwallet.derivations.BIP32Derivation method*), 36
compressed() (*hdwallet.hdwallet.HDWallet method*), 20
cryptocurrency() (*hdwallet.hdwallet.HDWallet method*), 22

D

Derivation (*class in hdwallet.derivations*), 33
dumps() (*hdwallet.hdwallet.HDWallet method*), 26

E

entropy() (*hdwallet.hdwallet.HDWallet method*), 21
entropy_to_mnemonic() (*in module hdwallet.utils*), 42

F

finger_print() (*hdwallet.hdwallet.HDWallet method*), 24
from_account() (*hdwallet.derivations.BIP32Derivation method*), 35
from_address() (*hdwallet.derivations.BIP32Derivation method*), 35
from_change() (*hdwallet.derivations.BIP32Derivation method*), 35
from_coin_type() (*hdwallet.derivations.BIP32Derivation method*), 35
from_entropy() (*hdwallet.hdwallet.HDWallet method*), 15
from_index() (*hdwallet.derivations.Derivation method*), 33
from_index() (*hdwallet.hdwallet.HDWallet method*), 18
from_mnemonic() (*hdwallet.hdwallet.HDWallet method*), 15
from_path() (*hdwallet.derivations.Derivation class method*), 33
from_path() (*hdwallet.hdwallet.HDWallet method*), 17
from_private_key() (*hdwallet.hdwallet.HDWallet method*), 17
from_public_key() (*hdwallet.hdwallet.HDWallet method*), 17
from_purpose() (*hdwallet.derivations.BIP32Derivation method*), 34

from_seed() (*hdwallet.hdwallet.HDWallet method*), 16
from_wif() (*hdwallet.hdwallet.HDWallet method*), 17
from_xprivate_key() (*hdwallet.hdwallet.HDWallet method*), 16
from_xpublic_key() (*hdwallet.hdwallet.HDWallet method*), 16

G

generate_entropy() (*in module hdwallet.utils*), 41
generate_mnemonic() (*in module hdwallet.utils*), 41
generate_passphrase() (*in module hdwallet.utils*), 41
get_entropy_strength() (*in module hdwallet.utils*), 42
get_mnemonic_language() (*in module hdwallet.utils*), 42
get_mnemonic_strength() (*in module hdwallet.utils*), 42

H

hash() (*hdwallet.hdwallet.HDWallet method*), 24
HDWallet (*class in hdwallet.hdwallet*), 15
hdwallet.utils
    module, 41

I

is_entropy() (*in module hdwallet.utils*), 41
is_mnemonic() (*in module hdwallet.utils*), 42

L

language() (*hdwallet.hdwallet.HDWallet method*), 22

M

mnemonic() (*hdwallet.hdwallet.HDWallet method*), 21
mnemonic_to_entropy() (*in module hdwallet.utils*), 43
module
    hdwallet.utils, 41

N

network() (*hdwallet.hdwallet.HDWallet method*), 23

P

p2pkh_address() (*hdwallet.hdwallet.HDWallet method*), 24
p2sh_address() (*hdwallet.hdwallet.HDWallet method*), 25
p2wpkh_address() (*hdwallet.hdwallet.HDWallet method*), 25
p2wpkh_in_p2sh_address() (*hdwallet.hdwallet.HDWallet method*), 25
p2wsh_address() (*hdwallet.hdwallet.HDWallet method*), 25
p2wsh_in_p2sh_address() (*hdwallet.hdwallet.HDWallet method*), 26
passphrase() (*hdwallet.hdwallet.HDWallet method*), 22
path() (*hdwallet.hdwallet.HDWallet method*), 23
private_key() (*hdwallet.hdwallet.HDWallet method*), 20
public_key() (*hdwallet.hdwallet.HDWallet method*), 20
purpose() (*hdwallet.derivations.BIP32Derivation method*), 36

R

root_xprivate_key() (*hdwallet.hdwallet.HDWallet method*), 18
root_xpublic_key() (*hdwallet.hdwallet.HDWallet method*), 18

S

seed() (*hdwallet.hdwallet.HDWallet method*), 23
semantic() (*hdwallet.hdwallet.HDWallet method*), 23
strength() (*hdwallet.hdwallet.HDWallet method*), 21
symbol() (*hdwallet.hdwallet.HDWallet method*), 22

U

uncompressed() (*hdwallet.hdwallet.HDWallet method*), 20

W

wif() (*hdwallet.hdwallet.HDWallet method*), 26

X

xprivate_key() (*hdwallet.hdwallet.HDWallet method*), 19
xpublic_key() (*hdwallet.hdwallet.HDWallet method*), 19