
HDWallet

Release 1.1.0

Meheret Tesfaye

May 23, 2021

CONTENTS

- 1 Hierarchical Deterministic Wallet 1**
- 2 Installing HDWallet 3**
 - 2.1 Development 3
- 3 Available Cryptocurrencies 5**
- 4 HDWallet 9**
 - 4.1 BIP32HDWallet 20
 - 4.2 BIP44HDWallet 21
 - 4.3 BIP49HDWallet 21
 - 4.4 BIP84HDWallet 22
 - 4.5 BIP141HDWallet 23
- 5 Derivation 25**
 - 5.1 BIP32Derivation 26
 - 5.2 BIP44Derivation 29
 - 5.3 BIP49Derivation 30
 - 5.4 BIP84Derivation 30
 - 5.5 BIP141Derivation 31
- 6 Utils 33**
- Python Module Index 37**
- Index 39**

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.

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) |

INSTALLING HDWALLET

The easiest way to install HDWallet is via pip:

```
$ pip install hdwallet
```

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 . -r requirements.txt
```


AVAILABLE CRYPTOCURRENCIES

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

| Cryptocurrencies | Symbols | Mainnet | Testnet | Seg-wit | 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 |
| 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 |

continues on next page

Table 1 – continued from previous page

| Cryptocurrencies | Symbols | Mainnet | Testnet | Seg-wit | Coin Type | Default Paths |
|--------------------|------------------------|---------|---------|---------|-----------|--------------------|
| 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, DASHT- EST | 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, DO- GETEST | 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, FIX- TEST | 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 |
| 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 |

continues on next page

Table 1 – continued from previous page

| Cryptocurrencies | Symbols | Mainnet | Testnet | Seg-wit | Coin Type | Default Paths |
|------------------|-----------------|---------|---------|---------|-----------|--------------------|
| 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, OM-NITEST | 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 |
| 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 |

continues on next page

Table 1 – continued from previous page

| Cryptocurrencies | Symbols | Mainnet | Testnet | Seg-wit | Coin Type | Default Paths |
|----------------------|-------------------|---------|---------|---------|-----------|---------------------|
| 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 | RBV | 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, STRAT-TEST | Yes | Yes | No | 105 | m/44'/105'/0'/0/0 |
| Sugarchain | SUGAR, SUG-ARTEST | 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 |
| Twins | TWINS, TWIN-TEST | 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, VIAT-EST | 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 |
| ZClassic | ZCL | Yes | No | No | 147 | m/44'/147'/0'/0/0 |
| Zcash | ZEC | Yes | No | No | 133 | m/44'/133'/0'/0/0 |
| Zencash | ZEN | Yes | No | No | 121 | m/44'/121'/0'/0/0 |

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

HDWALLET

Class Hierarchical Deterministic Wallet

```
class hdwallet.hdwallet.HDWallet (symbol: str = 'BTC', cryptocurrency: Optional[Any] = None,  
                                semantic: str = 'p2pkh', 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=
↪"8d5f4fe5b81a6a6a18b08603b6b3f59df9f4bbb25d10c55d23e0cbdc5ee385e5fddad9d7e6114f11afdec4592
↪")
<hdwallet.hdwallet.HDWallet object at 0x000001E8BFB98D60>
```

from_root_xprivate_key (*root_xprivate_key: str*) → *hdwallet.hdwallet.HDWallet*

Master from Root XPrivate Key.

Parameters **root_xprivate_key** (*str*) – Root xprivate key.

Returns HDWallet – Hierarchical Deterministic Wallet instance.

```
>>> from hdwallet import HDWallet
>>> from hdwallet.symbols import BTC
>>> hdwallet = HDWallet(symbol=BTC)
>>> hdwallet.from_root_xprivate_key(root_xprivate_key=
↪"xprv9s21ZrQH143K3xPGUzpogJeKtRdjHkK6muBJo8v7rEVRzT83xJgNcLpMoJXUf9wJfKfuHR4SGvf9gdShh4t9Vm
↪")
<hdwallet.hdwallet.HDWallet object at 0x000001E8BFB98D60>
```

from_xprivate_key (*xprivate_key: str*) → *hdwallet.hdwallet.HDWallet*

Master from XPrivate Key.

Parameters **xprivate_key** (*str*) – XPrivate key.

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=
↪"xprvA3BYGWQ9FmhyaNRRXB2f1LphNPnaY9T6gngw4BaTbkFtscSH4RCuJhgWUSks9S6ciGioHd4TX4UeyUg53Mkfn
↪")
<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_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_root_xprivate_key(root_xprivate_key=
↳ "xprv9s21ZrQH143K3xPGUzpgogJeKtRdjHkK6muBjO8v7rEVRzT83xJgNcLpMoJXUf9wJFKfuHR4SGvfGdShh4t9Vn
↳ ")
>>> hdwallet.from_path(path="m/44'/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_root_xprivate_key(root_xprivate_key=
↳ "xprv9s21ZrQH143K3xPGUzpgogJeKtRdjHkK6muBjO8v7rEVRzT83xJgNcLpMoJXUf9wJFKfuHR4SGvfGdShh4t9Vn
↳ ")
>>> 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)
```

(continues on next page)

(continued from previous page)

```
>>> 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()

↪ "xprv9s21ZrQH143K3xPGUzpogJeKtRdjHkK6muBJo8v7rEVRzT83xJgNcLpMoJXUf9wJFKfuHR4SGvfgdShh4t9Vn
↪ "
```

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()

↪ "xpub661MyMwAqRbcGSTjb2Mp3Sb4STUDhD2x986ubXKjQa2QsFTCVqzdA98qeZjcnCHT1AaZcMSjiP1HJ16jH97q7
↪ "
```

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()

↪ "xprvA3BYGWQ9FmhyaNRRXB2f1LphNPnaY9T6gngw4BaTbkFtscSH4RCuJhgWUSKs9S6ciGioHd4TX4UeyUg53MkfN
↪ "
```

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()

↪ "xpub6GAtglw369GGnrVtdCZfNUmRvRd4wcAx41cXrZz5A5nskQmRbxX9rVzzKiRU4JruirBrfm4KQXNSU7GfqLltz
↪ "
```

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_root_xprivate_key(root_xprivate_key=
↪ "xprv9s212rQH143K3xPGUzpogJeKtRdjHkK6muBJo8v7rEVRzT83xJgNcLpMoJXUf9wJFKfuHR4SGvfqdShh4t9Vm
↪ ")
>>> 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() → str

Get Uncompressed Public Key.

Returns str – Uncompressed 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()

↪ "f93f58b97c3bb616645c3dda256ec946d87c45baf531984c022dd0fd1503b0a875f63285a539213ac241fc4a8
↪ "
```

compressed() → str

Get Compressed Public Key.

Returns str – Compressed 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(private_key: Optional[str] = None) → str

Get Public Key.

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")
>>> 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
```

(continues on next page)

(continued from previous page)

```
>>> 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.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()

↳"8d5f4fe5b81a6a6a18b08603b6b3f59df9f4bbb25d10c55d23e0cbdc5ee385e5fddad9d7e6114f11afdec4592
↳"
```

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
↳")
>>> 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)
>>> 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()
"bclqfky82ek5pr07t65qjretw9hevw2j8j5fdrn5hc"
```

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()
"bclqaj2xa9j6eegcxls3y8p6erw6vdgdxynasrd4hl3xuctht5edu3msdeshgf"
```

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()
"38YMonfh2yLFRViLrM2kdvZx8ctcplvbbV"
```

wif() → 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()
"KzsHWUJsrTWUUhBGPfMMxLLydiH7NhEn6z7mKHXD5qNkUWaC4TEen"
```

dumps() → dict

Get All HDWallet informations.

Returns dict – All HDWallet informations.

```
>>> 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':
↳'8d5f4fe5b81a6a6a18b08603b6b3f59df9f4bbb25d10c55d23e0cbdc5ee385e5fddad9d7e6114f11afdec4592
↳', 'root_xprivate_key':
```

(continues on next page)

```
↳'xprv9s21ZrQH143K3xPGUzpogJeKtRdjHkK6muBjO8v7rEVRzT83xJgNcLpMoJXUf9wJfKfuHR4SGvfgdShh4t9Vn
↳', 'root_xpublic_key':
↳'xpub661MyMwAqRbcGSTjb2Mp3Sb4STUDhD2x986ubXKjQa2QsFTCVqzdA98qeZjcnCHT1AaZcM9jiP1HJ16jH97q7
↳', 'xprivate_key':
↳'xprvA3BYGWQ9FmhyaNRRXB2f1LphNPnaY9T6gngw4BaTbkFtscSH4RCuJhgWUSKs9S6ciGioHd4TX4UeyUg53Mkfn
↳', 'xpublic_key':
↳'xpub6GAtg1w369GGnrVtdCZfNUmRvRd4wcAx41cXrZz5A5nskQmRbxX9rVzzKiRU4JruirBrfm4KQXNSU7GfqLlt
```

4.1 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"
```


4.2 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"
```

4.3 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"
```

4.4 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
>>> 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"
```

4.5 BIP141HDWallet

Class BIP141 Hierarchical Deterministic Wallet

```
class hdwallet.hdwallet.BIP141HDWallet(symbol: str = 'BTC', cryptocurrency: Op-
tional[Any] = None, path: Optional[Union[str, hd-
wallet.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/0/0/0", 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")
>>> 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"
```


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 with '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)
""
```

5.1 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 change.

Returns BIP32Derivation – BIP32Derivation instance.

```
>>> from hdwallet.derivations import BIP32Derivation
>>> bip32_derivation = BIP32Derivation()
>>> bip32_derivation.from_change(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"
```

5.2 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>
>>> str(BIP44Derivation(cryptocurrency=BitcoinMainnet))
"m/44'/0'/0'/0/0"
```

5.3 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"
```

5.4 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/49'/0'/0'/0/0"
```

5.5 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/49'/0'/0'/0/0"
```


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)
"N39rPf3QvF2Tm2nPyoBpXNiBFXJyWTz"
```

`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 – Entropy hex string.

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


PYTHON MODULE INDEX

h

`hdwallet.utils`, [33](#)

INDEX

A

`account()` (*hdwallet.derivations.BIP32Derivation method*), 28
`address()` (*hdwallet.derivations.BIP32Derivation method*), 29
`address()` (*hdwallet.hdwallet.BIP141HDWallet method*), 23
`address()` (*hdwallet.hdwallet.BIP32HDWallet method*), 20
`address()` (*hdwallet.hdwallet.BIP44HDWallet method*), 21
`address()` (*hdwallet.hdwallet.BIP49HDWallet method*), 22
`address()` (*hdwallet.hdwallet.BIP84HDWallet method*), 22

B

`BIP141Derivation` (*class in hdwallet.derivations*), 31
`BIP141HDWallet` (*class in hdwallet.hdwallet*), 23
`BIP32Derivation` (*class in hdwallet.derivations*), 26
`BIP32HDWallet` (*class in hdwallet.hdwallet*), 20
`BIP44Derivation` (*class in hdwallet.derivations*), 29
`BIP44HDWallet` (*class in hdwallet.hdwallet*), 21
`BIP49Derivation` (*class in hdwallet.derivations*), 30
`BIP49HDWallet` (*class in hdwallet.hdwallet*), 21
`BIP84Derivation` (*class in hdwallet.derivations*), 30
`BIP84HDWallet` (*class in hdwallet.hdwallet*), 22

C

`chain_code()` (*hdwallet.hdwallet.HDWallet method*), 17
`change()` (*hdwallet.derivations.BIP32Derivation method*), 28
`clean_derivation()` (*hdwallet.derivations.BIP32Derivation method*), 28
`clean_derivation()` (*hdwallet.derivations.Derivation method*), 26
`clean_derivation()` (*hdwallet.hdwallet.HDWallet method*), 13

`coin_type()` (*hdwallet.derivations.BIP32Derivation method*), 28
`compressed()` (*hdwallet.hdwallet.HDWallet method*), 13
`cryptocurrency()` (*hdwallet.hdwallet.HDWallet method*), 15

D

`Derivation` (*class in hdwallet.derivations*), 25
`dumps()` (*hdwallet.hdwallet.HDWallet method*), 19

E

`entropy()` (*hdwallet.hdwallet.HDWallet method*), 14
`entropy_to_mnemonic()` (*in module hdwallet.utils*), 34

F

`finger_print()` (*hdwallet.hdwallet.HDWallet method*), 17
`from_account()` (*hdwallet.derivations.BIP32Derivation method*), 27
`from_address()` (*hdwallet.derivations.BIP32Derivation method*), 27
`from_change()` (*hdwallet.derivations.BIP32Derivation method*), 27
`from_coin_type()` (*hdwallet.derivations.BIP32Derivation method*), 27
`from_entropy()` (*hdwallet.hdwallet.HDWallet method*), 9
`from_index()` (*hdwallet.derivations.Derivation method*), 25
`from_index()` (*hdwallet.hdwallet.HDWallet method*), 11
`from_mnemonic()` (*hdwallet.hdwallet.HDWallet method*), 9
`from_path()` (*hdwallet.derivations.Derivation class method*), 25

[from_path\(\)](#) (*hdwallet.hdwallet.HDWallet method*), [11](#)
[from_private_key\(\)](#) (*hdwallet.hdwallet.HDWallet method*), [11](#)
[from_purpose\(\)](#) (*hdwallet.derivations.BIP32Derivation method*), [26](#)
[from_root_xprivate_key\(\)](#) (*hdwallet.hdwallet.HDWallet method*), [10](#)
[from_seed\(\)](#) (*hdwallet.hdwallet.HDWallet method*), [10](#)
[from_wif\(\)](#) (*hdwallet.hdwallet.HDWallet method*), [10](#)
[from_xprivate_key\(\)](#) (*hdwallet.hdwallet.HDWallet method*), [10](#)

G

[generate_entropy\(\)](#) (*in module hdwallet.utils*), [33](#)
[generate_mnemonic\(\)](#) (*in module hdwallet.utils*), [33](#)
[generate_passphrase\(\)](#) (*in module hdwallet.utils*), [33](#)
[get_entropy_strength\(\)](#) (*in module hdwallet.utils*), [34](#)
[get_mnemonic_language\(\)](#) (*in module hdwallet.utils*), [34](#)
[get_mnemonic_strength\(\)](#) (*in module hdwallet.utils*), [34](#)

H

[hash\(\)](#) (*hdwallet.hdwallet.HDWallet method*), [17](#)
[HDWallet](#) (*class in hdwallet.hdwallet*), [9](#)
[hdwallet.utils](#)
 module, [33](#)

I

[is_entropy\(\)](#) (*in module hdwallet.utils*), [33](#)
[is_mnemonic\(\)](#) (*in module hdwallet.utils*), [33](#)

L

[language\(\)](#) (*hdwallet.hdwallet.HDWallet method*), [15](#)

M

[mnemonic\(\)](#) (*hdwallet.hdwallet.HDWallet method*), [14](#)
[mnemonic_to_entropy\(\)](#) (*in module hdwallet.utils*), [35](#)
[module](#)
 hdwallet.utils, [33](#)

N

[network\(\)](#) (*hdwallet.hdwallet.HDWallet method*), [16](#)

P

[p2pkh_address\(\)](#) (*hdwallet.hdwallet.HDWallet method*), [17](#)
[p2sh_address\(\)](#) (*hdwallet.hdwallet.HDWallet method*), [18](#)
[p2wpkh_address\(\)](#) (*hdwallet.hdwallet.HDWallet method*), [18](#)
[p2wpkh_in_p2sh_address\(\)](#) (*hdwallet.hdwallet.HDWallet method*), [18](#)
[p2wsh_address\(\)](#) (*hdwallet.hdwallet.HDWallet method*), [18](#)
[p2wsh_in_p2sh_address\(\)](#) (*hdwallet.hdwallet.HDWallet method*), [19](#)
[passphrase\(\)](#) (*hdwallet.hdwallet.HDWallet method*), [15](#)
[path\(\)](#) (*hdwallet.hdwallet.HDWallet method*), [16](#)
[private_key\(\)](#) (*hdwallet.hdwallet.HDWallet method*), [13](#)
[public_key\(\)](#) (*hdwallet.hdwallet.HDWallet method*), [14](#)
[purpose\(\)](#) (*hdwallet.derivations.BIP32Derivation method*), [28](#)

R

[root_xprivate_key\(\)](#) (*hdwallet.hdwallet.HDWallet method*), [12](#)
[root_xpublic_key\(\)](#) (*hdwallet.hdwallet.HDWallet method*), [12](#)

S

[seed\(\)](#) (*hdwallet.hdwallet.HDWallet method*), [16](#)
[semantic\(\)](#) (*hdwallet.hdwallet.HDWallet method*), [16](#)
[strength\(\)](#) (*hdwallet.hdwallet.HDWallet method*), [14](#)
[symbol\(\)](#) (*hdwallet.hdwallet.HDWallet method*), [15](#)

U

[uncompressed\(\)](#) (*hdwallet.hdwallet.HDWallet method*), [13](#)

W

[wif\(\)](#) (*hdwallet.hdwallet.HDWallet method*), [19](#)

X

[xprivate_key\(\)](#) (*hdwallet.hdwallet.HDWallet method*), [12](#)
[xpublic_key\(\)](#) (*hdwallet.hdwallet.HDWallet method*), [12](#)