**Documentation**
The library utilizes classes to represent various Waves data structures:

- pywaves.Address
- pywaves.Asset
- pywaves.AssetPair
- pywaves.Order

**Code Example**
```
import pywaves as pw

myAddress =
pw.Address(privateKey='CtMQWJZqfc7PRzSWiMKaGmWFm4q2VN5fMcYyKDBPDx6S')
otherAddress = pw.Address('3PNTcNiUzppQXDL9RZrK3BcftbujiFqrAfM')
myAddress.sendWaves(otherAddress, 10000000)
myToken = myAddress.issueAsset('Token1', 'My Token', 1000, 0)
while not myToken.status():
	pass
myAddress.sendAsset(otherAddress, myToken, 50)
```

**Address Class**
**pywaves.Address(address, publicKey, privateKey, seed)** *Creates a new Address object*

**attributes:**

- *address*
- *publicKey*
- *privateKey*
- *seed*

**methods:**
balance(assetId='', confirmations=0) returns balance of Waves or other assets
assets() returns a list of assets owned by the address
issueAsset(name, description, quantity, decimals=0, reissuable=False, txFee=DEFAULT_ASSET_FEE, timestamp) issue a new asset
reissueAsset(Asset, quantity, reissuable=False, txFee=DEFAULT_ASSET_FEE, timestamp) reissue an asset
burnAsset(Asset, quantity, txFee=DEFAULT_ASSET_FEE, timestamp) burn the specified quantity of an asset
sendWaves(recipient, amount, attachment='', txFee=DEFAULT_TX_FEE, timestamp) send specified amount of Waves to recipient
sendAsset(recipient, asset, amount, attachment='', txFee=DEFAULT_TX_FEE, timestamp) send specified amount of an asset to recipient
cancelOrder(assetPair, order) cancel an order
buy(assetPair, amount price, maxLifetime=30*86400, matcherFee=DEFAULT_MATCHER_FEE, timestamp) post a buy order
tradableBalance(assetPair) get tradable balance for the specified asset pair

sell(assetPair, amount, price, maxLifetime=30*86400, matcherFee=DEFAULT_MATCHER_FEE, timestamp) post a sell order
lease(recipient, amount, txFee=DEFAULT_LEASE_FEE, timestamp) post a lease transaction
leaseCancel(leaseId, txFee=DEFAULT_LEASE_FEE, timestamp) cancel a lease
getOrderHistory(assetPair) get order history for the specified asset pair
cancelOpenOrders(assetPair) cancel all open orders for the specified asset pair
deleteOrderHistory(assetPair) delete order history for the specified asset pair
createAlias(alias, txFee=DEFAULT_ALIAS_FEE, timestamp) create alias

**Asset Class**
**pywaves.Asset(assetId)** *Creates a new Asset object*

**attributes:**
- *status*
- *assetId*
- *issuer*
- *name*
- *description*
- *quantity*
- *decimals* = 0
- *reissuable = False*

**methods:**
status() returns 'Issued' if the asset exists

**AssetPair Class**
**pywaves.AssetPair(asset1, asset2)** *Creates a new AssetPair object with 2 Asset objects*

**attributes:**
- *asset1*
- *asset2*

**methods:**
orderbook() get order book ticker() get ticker with 24h ohlcv data last() get traded price open() get 24h open price high() get 24h high price low() get 24h low price close() get 24h close price (same as last()) vwap() get 24h vwap price volume() get 24h volume priceVolume() get 24h price volume trades(n) get the last n trades trades(from, to) get the trades in from/to interval candles(timeframe, n) get the last n candles in the specified timeframe candles(timeframe, from, to) get the candles in from/to interval in the specified timeframe

**Order Class**
**pywaves.Order(orderId, assetPair, address='')** Creates a new Order object

**attributes:**

- *status*
- *orderId*
- *assetPair*
- *address*
- *matcher*
- *matcherPublicKey*

**methods:**
status() returns current order status cancel() cancel the order

**Other functions**
pywaves.setNode(node, chain) set node URL ('http://ip-address:port') and chain (either 'mainnet' or 'testnet')
pywaves.setChain(chain) set chain (either 'mainnet' or 'testnet')
pywaves.setOffline() switch to offline mode; sign tx locally without broadcasting to network
pywaves.setOnline() switch to online mode; sign tx locally a broadcast to network
pywaves.setMatcher(node) set matcher URL ('http://ip-address:port')
pywaves.setDatafeed(node) set datafeed URL ('http://ip-address:port')
pywaves.height() get blockchain height
pywaves.lastblock() get last block
pywaves.block(n) get block at specified height
pywaves.tx(id) get transaction details
pywaves.symbols() get list of symbol-asset mapping
pywaves.markets() get all traded markets with tickers
pywaves.{SYMBOL_NAME} get predefined asset for the specified symbol (pywaves.WAVES, pywaves.BTC, pywaves.USD,...)

**Default Fees**
The fees for waves/asset transfers, asset issue/reissue/burn and matcher transactions are set by default as follows:
- DEFAULT_TX_FEE = 100000
- DEFAULT_ASSET_FEE = 100000000
- DEFAULT_MATCHER_FEE = 1000000
- DEFAULT_LEASE_FEE = 100000
- DEFAULT_ALIAS_FEE = 100000

**More Examples**

**Playing with addresses:**
```
import pywaves as pw

# generate a new address
myAddress = pw.Address()

# set an address with a public key
```

```
myAddress = pw.Address('3P6WfA4qYtkgwVAsWiiB6yaea2X8zyXncJh')

# get an existing address from seed
myAddress = pw.Address(seed='seven wrist bargain hope pattern banner plastic maple
student chaos grit next space visa answer')

# get an existing address from privateKey
myAddress =
pw.Address(privateKey='CtMQWJZqfc7PRzSWiMKaGmWFm4q2VN5fMcYyKDBPDx6S')
```

**Balances:**

```
import pywaves as pw

myAddress = pw.Address('3P6WfA4qYtkgwVAsWiiB6yaea2X8zyXncJh')

# get Waves balance
print("Your balance is %18d" % myAddress.balance())

# get Waves balance after 20 confirmations
print("Your balance is %18d" % myAddress.balance(confirmations = 20))

# get an asset balance
print("Your asset balance is %18d" %
myAddress.balance('DHgwrRvVyqJsepd32YbBqUeDH4GJ1N984X8QoekjgH8J'))
```

**Waves and asset transfers:**

```
import pywaves as pw

myAddress =
pw.Address(privateKey='CtMQWJZqfc7PRzSWiMKaGmWFm4q2VN5fMcYyKDBPDx6S')

# send Waves to another address
myAddress.sendWaves(recipient =
pw.Address('3PNTcNiUzppQXDL9RZrK3BcftbujiFqrAfM'),
                amount = 100000000)

# send asset to another address
myToken = pw.Asset('4ZzED8WJXsvuo2MEm2BmZ87Azw8Sx7TVC6ufSUA5LyTV')
myAddress.sendAsset(recipient =
pw.Address('3PNTcNiUzppQXDL9RZrK3BcftbujiFqrAfM'),
                asset = myToken,
                amount = 1000)
```

**Issuing an asset:**

```
import pywaves as pw
```

```
myToken = myAddress.issueToken( name = "MyToken",
                    description = "This is my first token",
                    quantity = 1000000,
                    decimals = 2 )
```

**Create an alias:**

```
import pywaves as pw

pw.setNode(node = 'http://127.0.0.1:6869', chain = 'testnet')

myAddress =
pw.Address(privateKey='CtMQWJZqfc7PRzSWiMKaGmWFm4q2VN5fMcYyKDBPDx6S')
myAddress.createAlias("MYALIAS1")
```

**Mass payment:**

```
import pywaves as pw

recipients =  ['3PBbp6bg2YEnHfdJtYM7jzzXYQeb7sx5oFg',
          '3P4A27aCd3skNja46pcgrLYEnK36TkSzgUp',
          '3P81U3ujotNUwZMWALdcJQLzBVbrAuUQMfs',
          '3PGcKEMwQcEbmeL8Jhe9nZQRBNCNdcHCoZP',
          '3PKjtzZ4FhKrJUikbQ1hRk5xbwVKDyTyvkn']

myAddress = pw.Address(privateKey =
"CtMQWJZqfc7PRzSWiMKaGmWFm4q2VN5fMcYyKDBPDx6S")

for address in recipients:
        myAddress.sendWaves(pw.Address(address), 1000000)
```

**Token airdrop:**

```
import pywaves as pw

myAddress = pw.Address(privateKey =
'CtMQWJZqfc7PRzSWiMKaGmWFm4q2VN5fMcYyKDBPDx6S')
myToken = pw.Asset('4ZzED8WJXsvuo2MEm2BmZ87Azw8Sx7TVC6ufSUA5LyTV')
amount = 1000

with open('recipients.txt') as f:
        lines = f.readlines()
for address in lines:
        myAddress.sendAsset(pw.Address(address.strip()), myToken, amount)
```

**Playing with Waves Matcher node (DEX):**

```
import pywaves as pw

# set Matcher node to use
```

```
pw.setMatcher(node = 'http://127.0.0.1:6886')

# post a buy order
BTC = pw.Asset('4ZzED8WJXsvuo2MEm2BmZ87Azw8Sx7TVC6ufSUA5LyTV')
USD = pw.Asset('6wuo2hTaDyPQVceETj1fc5p4WoMVCGMYNASN8ym4BGiL')
BTC_USD = pw.AssetPair(BTC, USD)
myOrder = myAddress.buy(assetPair = BTC_USD, amount = 15e8, price = 95075)

# post a sell order
WCT = pw.Asset('6wuo2hTaDyPQVceETj1fc5p4WoMVCGMYNASN8ym4BGiL')
Incent = pw.Asset('FLbGXzrpqkvucZqsHDcNxePTkh2ChmEi4GdBfDRRJVof')
WCT_Incent = pw.AssetPair(WCT, Incent)
myOrder = myAddress.sell(assetPair = WCT_Incent, amount = 100e8, price = 25e8)

# post a buy order using Waves as price asset
BTC = pw.Asset('4ZzED8WJXsvuo2MEm2BmZ87Azw8Sx7TVC6ufSUA5LyTV')
BTC_WAVES = pw.AssetPair(BTC, pw.WAVES)
myOrder = myAddress.buy(assetPair = BTC_WAVES, amount = 1e8, price = 50e8)

# cancel an order
myOrder.cancel()
# or
myAddress.cancelOrder(assetPair, myOrder)
```

**Getting Market Data from Waves Data Feed (WDF):**

```
import pywaves as pw

# set the asset pair
WAVES_BTC = pw.AssetPair(pw.WAVES, pw.BTC)

# get last price and volume
print("%s %s" % (WAVES_BTC.last(), WAVES_BTC.volume()))

# get ticker
ticker = WAVES_BTC.ticker()
print(ticker['24h_open'])
print(ticker['24h_vwap'])

# get last 10 trades
trades = WAVES_BTC.trades(10)
for t in trades:
        print("%s %s %s %s" % (t['buyer'], t['seller'], t['price'], t['amount']))

# get last 10 daily OHLCV candles
ohlcv = WAVES_BTC.candles(1440, 10)
for t in ohlcv:
        print("%s %s %s %s %s" % (t['open'], t['high'], t['low'], t['close'], t['volume']))
```

**LPOS**

```
import pywaves as pw

# connect to a local testnet node
pw.setNode(node = 'http://127.0.0.1:6869', chain = 'testnet')

myAddress = pw.Address(privateKey =
'CsBpQpNE3Z1THNMS9vJPaXqYwN9Hgmhd9AsAPrM3tiuJ')
minerAddress = pw.Address('3NBThmVJmcexzJ9itP9KiiC2K6qnGQwpqMq')

# lease 1000 Waves to minerAddress
leaseId = myAddress.lease(minerAddress, 100000000000)

# revoke the lease
myAddress.leaseCancel(leaseId)
```

**Using PyWaves in a Python shell**

**Check an address balance:**

```
>>> import pywaves as pw
>>> pw.Address('3P31zvGdh6ai6JK6zZ18TjYzJsa1B83YPoj')
address = 3P31zvGdh6ai6JK6zZ18TjYzJsa1B83YPoj
publicKey = 
privateKey = 
seed = 
balances:
  Waves = 1186077288304570
  BDMRyZsmDZpgKhdM7fUTknKcUbVVkDpMcqEj31PUzjMy (Tokes) = 43570656915
  RRBqh2XxcwAdLYEdSickM589Vb4RCemBCPH5mJaWhU9 (Ripto Bux) = 
4938300000000
  4rmhfoscYcjz1imNDvtz45doouvrQqDpbX7xdfLB4guF (incentCoffee) = 7
  Ftim86CXM6hANxArJXZs2Fq7XLs3nJvgBzzEwQWwQn6N (Waves) = 
2117290600000000
  E4ip4jzTc4PCvebYn1818T4LNoYBVL3Y4Y4dMPatGwa9 (BitCoin) = 500000000000
  FLbGXzrpqkvucZqsHDcNxePTkh2ChmEi4GdBfDRRJVof (Incent) = 12302659925430
  GQr2fpkfmWjMaZCbqMxefbiwgvpcNgYdev7xpuX6xqcE (KISS) = 1000
  DxG3PLganyNzajHGzvWLjc4P3T2CpkBGxY4J9eJAAUPw (UltraCoin) = 
200000000000000
  4eWBPyY4XNPsFLoQK3iuVUfamqKLDu5o6zQCYyp9d8Ae (LIKE) = 1000
>>>
```

**Generate a new address:**

```
>>> import pywaves as pw
>>> pw.Address()
```

```
address = 3P6WfA4qYtkgwVAsWiiB6yaea2X8zyXncJh
publicKey = EYNuSmW4Adtcc6AMCZyxkiHMPmF2BZ2XxvjpBip3UFZL
privateKey = CtMQWJZqfc7PRzSWiMKaGmWFm4q2VN5fMcYyKDBPDx6S
seed = seven wrist bargain hope pattern banner plastic maple student chaos grit next
space visa answer
balances:
  Waves = 0
>>>
```

**Check an asset:**

```
>>> import pywaves as pw
>>> pw.Asset('DHgwrRvVyqJsepd32YbBqUeDH4GJ1N984X8QoekjgH8J')
status = Issued
assetId = DHgwrRvVyqJsepd32YbBqUeDH4GJ1N984X8QoekjgH8J
issuer = 3PPKF2pH4KMYgsDixjrhnWrPycVHr1Ye37V
name = WavesCommunity
description = Waves community token.
quantity = 1000000000
decimals = 2
reissuable = False
```

**Post an order and check its status:**

```
>>> myOrder = myAddress.buy(pw.AssetPair(token1, token2), 1, 25)
>>> myOrder
status = Accepted
id = ARZdYgfXz3ksRMvhnGeLLJnn3CQnz7RCa7U6dVw3zert
asset1 = AFzL992FQbhcgSZGKDKAiRWcjtthM55yVCE99hwbHf88
asset2 = 49Aha2RR2eunR3KZFwedfdi7K9v5MLQbLYcmVdp2QkZT
sender.address = 3P6WfA4qYtkgwVAsWiiB6yaea2X8zyXncJh
sender.publicKey = EYNuSmW4Adtcc6AMCZyxkiHMPmF2BZ2XxvjpBip3UFZL
matcher = http://127.0.0.1:6886
```

**Cancel the order**

```
>>> myOrder.cancel()
>>> myOrder
status = Cancelled
id = ARZdYgfXz3ksRMvhnGeLLJnn3CQnz7RCa7U6dVw3zert
asset1 = AFzL992FQbhcgSZGKDKAiRWcjtthM55yVCE99hwbHf88
asset2 = 49Aha2RR2eunR3KZFwedfdi7K9v5MLQbLYcmVdp2QkZT
sender.address = 3P6WfA4qYtkgwVAsWiiB6yaea2X8zyXncJh
sender.publicKey = EYNuSmW4Adtcc6AMCZyxkiHMPmF2BZ2XxvjpBip3UFZL
matcher = http://127.0.0.1:6886
```

**Offline signing and custom timestamps**

**Offline signing a future transaction:**

```
>>> import pywaves as pw
>>> pw.setOffline()
>>>
myAddress=pw.Address(privateKey="F2jVbjrKzjUsZ1AQRdnd8MmxFc85NQz5jwvZX4B
XswXv")
>>> recipient=pw.Address("3P8Ya6Ary5gzwnzbBXDp3xjeNG97JEiPcdA")
# sign a future tx to transfer 100 WAVES to recipient
# the tx is valid on Jan 1st, 2020 12:00pm
>>> myAddress.sendWaves(recipient, amount=100e8, timestamp=1577880000000)
{'api-endpoint': '/assets/broadcast/transfer',
 'api-type': 'POST',
 'api-data': '{"fee": 100000,
                                   "timestamp": 1577880000000,
                                   "senderPublicKey":
"27zdzBa1q46RCMamZ8gw2xrTGypZnbzXs5J1Y2HbUmEv",
                                   "amount": 10000000000,
                                   "attachment": "",
                                   "recipient": "3P8Ya6Ary5gzwnzbBXDp3xjeNG97JEiPcdA"
                                   "signature":
"YetPopTJWC4WBPXbneWv9g6YEp6J9g9rquZWjewjdQnFbmaxtXjrRsUu69NZzHebVzU
GLrhQiFFoguXJwdUn8BH"}'}
```

**Offline signing time lock/unlock transactions:**

```
>>> import pywaves as pw
>>> pw.setOffline()
>>>
myAddress=pw.Address(privateKey="F2jVbjrKzjUsZ1AQRdnd8MmxFc85NQz5jwvZX4B
XswXv")
# generate a lockbox address
>>> lockAddress=pw.Address()
# sign the 'lock' tx to send 100e8 to the lockbox (valid on Nov 1st, 2017)
>>> myAddress.sendWaves(lockAddress, 100e8, timestamp=1509537600000)
{'api-endpoint': '/assets/broadcast/transfer',
 'api-type': 'POST',
 'api-data': '{"fee": 100000,
              "timestamp": 1509537600000,
              "senderPublicKey": "27zdzBa1q46RCMamZ8gw2xrTGypZnbzXs5J1Y2HbUmEv",
              "amount": 10000000000,
              "attachment": "",
              "recipient": "3P3UbyQM9W7WzTgjYkLuBrPZZeWsiUtCcpv",
              "signature":
"5VgT6qWxJwxEyrxFNfsi67QqbyUiGq9Ka7HVzgovRTTDT8nLRyuQv2wBAJQhRiXDkTTV
6zsQmHnBkh8keCaFPoNT"}'}
# sign the 'unlock' tx to send funds back to myAddress (valid on Jan 1st, 2020)
>>> lockAddress.sendWaves(myAddress, 100e8-200000, txFee=200000,
timestamp=1577880000000)
```

```
{'api-endpoint': '/assets/broadcast/transfer',
 'api-type': 'POST',
 'api-data': '{"fee": 200000,
             "timestamp": 1577880000000,
                                 "senderPublicKey":
"52XnBGnAVZmw1CHo9aJPiMsVMiTWeNGSNN9aYJ7cDtx4",
                                 "amount": 9999800000,
                                 "attachment": "",
                                 "recipient": "3P7tfdCaTyYCfg5ojxNahEJDSS4MZ7ybXBY",
                                 "signature":
"3beyz1sqKefP96LaXWT3CxdPRW86DAxcj6wgWPyyKq3SgdotVqnKyWXDyeHnBzCq1n
C7JA9CChTmo1c1iVAv6C4T"}'}
# delete lockbox address and private key
>>> del lockAddress
```