
Ikena Client

Arash Fatahzade

May 04, 2021

CONTENTS:

1	API Reference	3
1.1	idena	3
2	Installation	35
3	Using Idena	37
3.1	Calling <i>client.init</i>	37
3.2	Setting environment variables	37
4	Getting Blockchain Info	39
5	Indices and tables	41
	Python Module Index	43
	Index	45

This is Python wrapper for Idene Node.

Please checkout API refrence for avaiable APIs.

API REFERENCE

This page contains auto-generated API reference documentation¹.

1.1 idena

1.1.1 Subpackages

`idena.apis`

Submodules

`idena.apis.account`

Module Contents

Functions

`create_account(password)`

`get_accounts()`

`lock_account(address)`

`unlock_account(address, password = None, time = None)`

`idena.apis.account.create_account` (*password*)

Parameters `password` (*None*) –

Return type `str`

`idena.apis.account.get_accounts` ()

Return type `List[str]`

`idena.apis.account.lock_account` (*address*)

Parameters `address` (*str*) –

¹ Created with sphinx-autoapi

Return type None

`idena.apis.account.unlock_account` (*address*, *password=None*, *time=None*)

Parameters

- **address** (*str*) –
- **password** (*str*) –
- **time** (*int*) –

Return type None

`idena.apis.blockchain`

Module Contents

Functions

`check_sync()`

`get_address_pending_transactions`(*address*)

`get_address_transactions`(*address*, *count = 5*,
token = None)

`get_block_by_hash`(*hash*)

`get_block_by_height`(*height*)

`get_burnt_coins()`

`get_fee_rate()`

`get_last_block()`

`get_mempool()`

`get_raw_tx`(*nonce = None*, *epoch = None*, *from_ =*
None, *to = None*, *amount = None*, *maxFee = None*,
payload = None, *tips = None*, *useProto = None*, *type*
= types.TxType.SendTx)

`get_transaction`(*hash*)

`get_transaction_receipt`(*hash*)

`send_raw_tx`(*tx*)

`idena.apis.blockchain.check_sync()`

Return type *idena.types.Sync*

`idena.apis.blockchain.get_address_pending_transactions` (*address*)

Parameters `address (str)` –

Return type `idena.types.AddressTransactions`

`idena.apis.blockchain.get_address_transactions (address, count=5, token=None)`

Parameters

- `address (str)` –
- `count (int)` –
- `token (str)` –

Return type `idena.types.AddressTransactions`

`idena.apis.blockchain.get_block_by_hash (hash)`

Parameters `hash (str)` –

Return type `idena.types.Block`

`idena.apis.blockchain.get_block_by_height (height)`

Parameters `height (int)` –

Return type `idena.types.Block`

`idena.apis.blockchain.get_burnt_coins ()`

Return type `Optional[idena.types.BurntCoins]`

`idena.apis.blockchain.get_fee_rate ()`

Return type `decimal.Decimal`

`idena.apis.blockchain.get_last_block ()`

Return type `idena.types.Block`

`idena.apis.blockchain.get_mempool ()`

Return type `Optional[str]`

`idena.apis.blockchain.get_raw_tx (nonce=None, epoch=None, from_=None, to=None, amount=None, maxFee=None, payload=None, tips=None, useProto=None, type=types.TxType.SendTx)`

Parameters

- `nonce (int)` –
- `epoch (int)` –
- `from_ (str)` –
- `to (str)` –
- `amount (decimal.Decimal)` –
- `maxFee (decimal.Decimal)` –
- `payload (str)` –
- `tips (decimal.Decimal)` –
- `useProto (bool)` –
- `type (idena.types.TxType)` –

Return type `str`

`idena.apis.blockchain.get_transaction` (*hash*)

Parameters `hash` (*str*) –

Return type `idena.types.Transaction`

`idena.apis.blockchain.get_transaction_receipt` (*hash*)

Parameters `hash` (*str*) –

Return type `Optional[idena.types.TxReceipt]`

`idena.apis.blockchain.send_raw_tx` (*tx*)

Parameters `tx` (*str*) –

Return type `str`

`idena.apis.contract`

Module Contents

Functions

`call_contract`(*from_* = None, *contract* = None, *method* = None, *amount* = None, *maxFee* = None, *args* = None, *broadcastBlock* = None)

`deploy_contract`(*from_* = None, *codeHash* = None, *amount* = None, *maxFee* = None, *args* = None)

`estimate_call`(*from_* = None, *contract* = None, *method* = None, *amount* = None, *maxFee* = None, *args* = None, *broadcastBlock* = None)

`estimate_deploy`(*from_* = None, *codeHash* = None, *amount* = None, *maxFee* = None, *args* = None)

`estimate_terminate`(*from_* = None, *contract* = None, *maxFee* = None, *args* = None)

`get_stake`(*contract*)

`read_contract_data`(*contract* = None, *key* = None, *format* = None)

`read_events`(*contract* = None)

`readonly_call_contract`(*contract* = None, *method* = None, *format* = None, *args* = None)

`subscribe_to_contract_event`(*contract* = None, *event* = None)

`terminate_contract`(*from_* = None, *contract* = None, *maxFee* = None, *args* = None)

`unsubscribe_from_contract_event`(*contract* = None, *event* = None)

`idena.apis.contract.call_contract` (*from_*=None, *contract*=None, *method*=None, *amount*=None, *maxFee*=None, *args*=None, *broadcastBlock*=None)

Parameters

- **from_** (*str*) –
- **contract** (*str*) –
- **method** (*str*) –
- **amount** (*decimal.Decimal*) –
- **maxFee** (*decimal.Decimal*) –
- **args** (*List[idena.types.DynamicArg]*) –
- **broadcastBlock** (*int*) –

Return type *str*

```
idena.apis.contract.deploy_contract (from_=None, codeHash=None, amount=None,
                                     maxFee=None, args=None)
```

Parameters

- **from_** (*str*) –
- **codeHash** (*str*) –
- **amount** (*decimal.Decimal*) –
- **maxFee** (*decimal.Decimal*) –
- **args** (*List[idena.types.DynamicArg]*) –

Return type *str*

```
idena.apis.contract.estimate_call (from_=None, contract=None, method=None,
                                   amount=None, maxFee=None, args=None, broadcast-
                                   Block=None)
```

Parameters

- **from_** (*str*) –
- **contract** (*str*) –
- **method** (*str*) –
- **amount** (*decimal.Decimal*) –
- **maxFee** (*decimal.Decimal*) –
- **args** (*List[idena.types.DynamicArg]*) –
- **broadcastBlock** (*int*) –

Return type *idena.types.ContractTxReceipt*

```
idena.apis.contract.estimate_deploy (from_=None, codeHash=None, amount=None,
                                     maxFee=None, args=None)
```

Parameters

- **from_** (*str*) –
- **codeHash** (*str*) –
- **amount** (*decimal.Decimal*) –
- **maxFee** (*decimal.Decimal*) –
- **args** (*List[idena.types.DynamicArg]*) –

Return type *idena.types.ContractTxReceipt*

`idena.apis.contract.estimate_terminate` (*from_=None, contract=None, maxFee=None, args=None*)

Parameters

- **from_** (*str*) –
- **contract** (*str*) –
- **maxFee** (*decimal.Decimal*) –
- **args** (*List[idena.types.DynamicArg]*) –

Return type *idena.types.ContractTxReceipt*

`idena.apis.contract.get_stake` (*contract*)

Parameters **contract** (*str*) –

Return type *idena.types.Stake*

`idena.apis.contract.read_contract_data` (*contract=None, key=None, format=None*)

Parameters

- **contract** (*str*) –
- **key** (*str*) –
- **format** (*str*) –

Return type *Any*

`idena.apis.contract.read_events` (*contract=None*)

Parameters **contract** (*str*) –

Return type *None*

`idena.apis.contract.readonly_call_contract` (*contract=None, method=None, format=None, args=None*)

Parameters

- **contract** (*str*) –
- **method** (*str*) –
- **format** (*str*) –
- **args** (*List[idena.types.DynamicArg]*) –

Return type *Any*

`idena.apis.contract.subscribe_to_contract_event` (*contract=None, event=None*)

Parameters

- **contract** (*str*) –
- **event** (*str*) –

Return type *None*

`idena.apis.contract.terminate_contract` (*from_=None, contract=None, maxFee=None, args=None*)

Parameters

- **from_** (*str*) –

- **contract** (*str*) –
- **maxFee** (*decimal.Decimal*) –
- **args** (*List[idena.types.DynamicArg]*) –

Return type *str*

`idena.apis.contract.unsubscribe_from_contract_event` (*contract=None, event=None*)

Parameters

- **contract** (*str*) –
- **event** (*str*) –

Return type *None*

`idena.apis.dna`

Module Contents

Functions

`activate_invite`(*nonce = None, epoch = None, key = None*)

`activate_invite_to_random_address`(*nonce = None, epoch = None, key = None*)

`become_offline`(*nonce = None, epoch = None*)

`become_online`(*nonce = None, epoch = None*)

`burn`(*nonce = None, epoch = None, from_ = None, amount = None, maxFee = None, key = None*)

`change_profile`(*info = None, nickname = None, maxFee = None*)

`export_key`(*password*)

`get_balance`(*address*)

`get_ceremony_intervals`()

`get_coinbase_address`()

`get_current_process`()

`get_epoch`()

`get_identities`()

`get_identity`(*address*)

`get_profile`(*address = None*)

continues on next page

Table 4 – continued from previous page

`isValidationReady()`

`send_invite(nonce = None, epoch = None, to = None, amount = None)`

`send_tx(nonce = None, epoch = None, from_ = None, to = None, amount = None, maxFee = None, payload = None, tips = None, useProto = None, type = types.TxType.SendTx)`

Attributes

`change_god_address`

`kill_identity`

`kill_invitee`

`send_dna`

`idena.apis.dna.change_god_address`
`idena.apis.dna.kill_identity`
`idena.apis.dna.kill_invitee`
`idena.apis.dna.send_dna`
`idena.apis.dna.activate_invite (nonce=None, epoch=None, key=None)`

Parameters

- **nonce** (*int*) –
- **epoch** (*int*) –
- **key** (*str*) –

Return type `idena.types.Invite`

`idena.apis.dna.activate_invite_to_random_address (nonce=None, epoch=None, key=None)`

Parameters

- **nonce** (*int*) –
- **epoch** (*int*) –
- **key** (*str*) –

Return type `idena.types.ActivateInviteToRandomAddr`

`idena.apis.dna.become_offline (nonce=None, epoch=None)`

Parameters

- **nonce** (*int*) –
- **epoch** (*int*) –

Return type str

`idena.apis.dna.become_online (nonce=None, epoch=None)`

Parameters

- **nonce** (*int*) –
- **epoch** (*int*) –

Return type str

`idena.apis.dna.burn (nonce=None, epoch=None, from_=None, amount=None, maxFee=None, key=None)`

Parameters

- **nonce** (*int*) –
- **epoch** (*int*) –
- **from_** (*str*) –
- **amount** (*decimal.Decimal*) –
- **maxFee** (*decimal.Decimal*) –
- **key** (*str*) –

Return type str

`idena.apis.dna.change_profile (info=None, nickname=None, maxFee=None)`

Parameters

- **info** (*str*) –
- **nickname** (*str*) –
- **maxFee** (*decimal.Decimal*) –

Return type *idena.types.ChangeProfile*

`idena.apis.dna.export_key (password)`

Parameters **password** (*str*) –

Return type str

`idena.apis.dna.get_balance (address)`

Parameters **address** (*str*) –

Return type *idena.types.Balance*

`idena.apis.dna.get_ceremony_intervals ()`

Return type *idena.types.CeremonyIntervals*

`idena.apis.dna.get_coinbase_address ()`

Return type str

`idena.apis.dna.get_current_process ()`

Return type *idena.types.State*

`idena.apis.dna.get_epoch ()`

Return type *idena.types.Epoch*

`idena.apis.dna.get_identities ()`

Return type List[idena.types.Identity]

idenapi.dna.get_identity(address)

Parameters address (str) –

Return type idena.types.Identity

idenapi.dna.get_profile(address=None)

Parameters address (str) –

Return type idena.types.Profile

idenapi.dna.is_validation_ready()

Return type bool

idenapi.dna.send_invite(nonce=None, epoch=None, to=None, amount=None)

Parameters

- nonce (int) –
- epoch (int) –
- to (str) –
- amount (decimal.Decimal) –

Return type idena.types.Invite

idenapi.dna.send_tx(nonce=None, epoch=None, from_=None, to=None, amount=None, maxFee=None, payload=None, tips=None, useProto=None, type=types.TxType.SendTx)

Parameters

- nonce (int) –
- epoch (int) –
- from_ (str) –
- to (str) –
- amount (decimal.Decimal) –
- maxFee (decimal.Decimal) –
- payload (str) –
- tips (decimal.Decimal) –
- useProto (bool) –
- type (idenapi.types.TxType) –

Return type str

`idena.apis.flip`

Module Contents

Functions

`delete_flip(hash)`

`flip_words(hash)`

`get_flip(hash)`

`get_long_flip_hashes()`

`get_raw_flip(hash)`

`get_short_flip_hashes()`

`submit_flip(publicHex = None, privateHex = None, pairId = None, hex = None)`

`submit_long_answers(answers)`

`submit_short_answers(answers)`

`idena.apis.flip.delete_flip(hash)`

Parameters `hash` (*str*) –

Return type `str`

`idena.apis.flip.flip_words(hash)`

Parameters `hash` (*str*) –

Return type `Tuple[int, int]`

`idena.apis.flip.get_flip(hash)`

Parameters `hash` (*str*) –

Return type `idena.types.Flip`

`idena.apis.flip.get_long_flip_hashes()`

Return type `List[idena.types.FlipHashes]`

`idena.apis.flip.get_raw_flip(hash)`

Parameters `hash` (*str*) –

Return type `idena.types.RawFlip`

`idena.apis.flip.get_short_flip_hashes()`

Return type `List[idena.types.FlipHashes]`

`idena.apis.flip.submit_flip(publicHex=None, privateHex=None, pairId=None, hex=None)`

Parameters

- **publicHex** (*str*) –
- **privateHex** (*str*) –
- **pairId** (*str*) –
- **hex** (*str*) –

Return type `idena.types.FlipSubmit`

`idena.apis.flip.submit_long_answers` (*answers*)

Parameters **answers** (`idena.types.FlipAnswers`) –

Return type `str`

`idena.apis.flip.submit_short_answers` (*answers*)

Parameters **answers** (`idena.types.FlipAnswers`) –

Return type `str`

`idena.apis.net`

Module Contents

Functions

`add_peers`(*url*)

`get_ipfs_addr`()

`get_peers`()

`idena.apis.net.add_peers` (*url*)

Return type `str`

`idena.apis.net.get_ipfs_addr` ()

Return type `str`

`idena.apis.net.get_peers` ()

Return type `List[idena.types.Peer]`

1.1.2 Submodules

`idena.client`

Module Contents

Functions

<code>init(rpc_node, api_key)</code>	Initialization
--------------------------------------	----------------

`idena.client.init(rpc_node, api_key)`

Initialization

Args: `rpc_node` (str): URL of rpc node `api_key` (str): API-Key

Parameters

- `rpc_node` (str) –
- `api_key` (str) –

Return type None

`idena.exceptions`

Module Contents

exception `idena.exceptions.IdenaException` (*code, message*)

Bases: Exception

Common base class for all non-exit exceptions.

`__repr__` (*self*)

Return repr(self).

`idena.types`

Module Contents

Classes

<i>ActivateInviteToRandomAddr</i>	Typed version of namedtuple.
<i>AddressTransactions</i>	Typed version of namedtuple.
<i>Answer</i>	Typed version of namedtuple.
<i>Balance</i>	Typed version of namedtuple.
<i>BaseTxArgs</i>	Typed version of namedtuple.
<i>Block</i>	Typed version of namedtuple.
<i>BurntCoins</i>	Typed version of namedtuple.
<i>CeremonyIntervals</i>	Typed version of namedtuple.
<i>ChangeProfile</i>	Typed version of namedtuple.
<i>ContractTxReceipt</i>	
<i>DynamicArg</i>	Typed version of namedtuple.
<i>Epoch</i>	Typed version of namedtuple.
<i>Flip</i>	Typed version of namedtuple.
<i>FlipAnswer</i>	Typed version of namedtuple.
<i>FlipAnswers</i>	Typed version of namedtuple.
<i>FlipHashes</i>	Typed version of namedtuple.
<i>FlipWords</i>	Typed version of namedtuple.

continues on next page

Table 9 – continued from previous page

<i>Grade</i>	
<i>Identity</i>	Typed version of namedtuple.
<i>Invite</i>	Typed version of namedtuple.
<i>Peer</i>	Typed version of namedtuple.
<i>Profile</i>	Typed version of namedtuple.
<i>RawFlip</i>	Typed version of namedtuple.
<i>SendTxArgs</i>	docstring
<i>Stake</i>	Typed version of namedtuple.
<i>State</i>	Typed version of namedtuple.
<i>Sync</i>	Typed version of namedtuple.
<i>Transaction</i>	Typed version of namedtuple.
<i>TxAddr</i>	Typed version of namedtuple.
<i>TxEvent</i>	Typed version of namedtuple.
<i>TxReceipt</i>	Typed version of namedtuple.
<i>TxType</i>	

Attributes

FlipSubmit

`idena.types.FlipSubmit`

class `idena.types.ActivateInviteToRandomAddr`

Bases: `NamedTuple`

Typed version of namedtuple.

Usage in Python versions >= 3.6:

```
class Employee(NamedTuple):
    name: str
    id: int
```

This is equivalent to:

```
Employee = collections.namedtuple('Employee', ['name', 'id'])
```

The resulting class has extra `__annotations__` and `_field_types` attributes, giving an ordered dict mapping field names to types. `__annotations__` should be preferred, while `_field_types` is kept to maintain pre PEP 526 compatibility. (The field names are in the `_fields` attribute, which is part of the namedtuple API.) Alternative equivalent keyword syntax is also accepted:

```
Employee = NamedTuple('Employee', name=str, id=int)
```

In Python versions <= 3.5 use:

```
Employee = NamedTuple('Employee', [('name', str), ('id', int)])
```

address :str

hash :str

key :str

class idena.types.AddressTransactions

Bases: NamedTuple

Typed version of namedtuple.

Usage in Python versions >= 3.6:

```
class Employee(NamedTuple):
    name: str
    id: int
```

This is equivalent to:

```
Employee = collections.namedtuple('Employee', ['name', 'id'])
```

The resulting class has extra `__annotations__` and `_field_types` attributes, giving an ordered dict mapping field names to types. `__annotations__` should be preferred, while `_field_types` is kept to maintain pre PEP 526 compatibility. (The field names are in the `_fields` attribute, which is part of the namedtuple API.) Alternative equivalent keyword syntax is also accepted:

```
Employee = NamedTuple('Employee', name=str, id=int)
```

In Python versions <= 3.5 use:

```
Employee = NamedTuple('Employee', [('name', str), ('id', int)])
```

token :Optional[str]

transactions :Optional[List[Transaction]]

class idena.types.Answer

Bases: NamedTuple

Typed version of namedtuple.

Usage in Python versions >= 3.6:

```
class Employee(NamedTuple):
    name: str
    id: int
```

This is equivalent to:

```
Employee = collections.namedtuple('Employee', ['name', 'id'])
```

The resulting class has extra `__annotations__` and `_field_types` attributes, giving an ordered dict mapping field names to types. `__annotations__` should be preferred, while `_field_types` is kept to maintain pre PEP 526 compatibility. (The field names are in the `_fields` attribute, which is part of the namedtuple API.) Alternative equivalent keyword syntax is also accepted:

```
Employee = NamedTuple('Employee', name=str, id=int)
```

In Python versions <= 3.5 use:

```
Employee = NamedTuple('Employee', [('name', str), ('id', int)])
```

left = 1

none = 0

```
right = 2
```

```
class idena.types.Balance
```

```
    Bases: NamedTuple
```

Typed version of namedtuple.

Usage in Python versions >= 3.6:

```
class Employee(NamedTuple):
    name: str
    id: int
```

This is equivalent to:

```
Employee = collections.namedtuple('Employee', ['name', 'id'])
```

The resulting class has extra `__annotations__` and `_field_types` attributes, giving an ordered dict mapping field names to types. `__annotations__` should be preferred, while `_field_types` is kept to maintain pre PEP 526 compatibility. (The field names are in the `_fields` attribute, which is part of the namedtuple API.) Alternative equivalent keyword syntax is also accepted:

```
Employee = NamedTuple('Employee', name=str, id=int)
```

In Python versions <= 3.5 use:

```
Employee = NamedTuple('Employee', [('name', str), ('id', int)])
```

```
balance :decimal.Decimal
```

```
nonce :int
```

```
stake :decimal.Decimal
```

```
class idena.types.BaseTxArgs
```

```
    Bases: NamedTuple
```

Typed version of namedtuple.

Usage in Python versions >= 3.6:

```
class Employee(NamedTuple):
    name: str
    id: int
```

This is equivalent to:

```
Employee = collections.namedtuple('Employee', ['name', 'id'])
```

The resulting class has extra `__annotations__` and `_field_types` attributes, giving an ordered dict mapping field names to types. `__annotations__` should be preferred, while `_field_types` is kept to maintain pre PEP 526 compatibility. (The field names are in the `_fields` attribute, which is part of the namedtuple API.) Alternative equivalent keyword syntax is also accepted:

```
Employee = NamedTuple('Employee', name=str, id=int)
```

In Python versions <= 3.5 use:

```
Employee = NamedTuple('Employee', [('name', str), ('id', int)])
```

```
epoch :int
```

```
nonce :int
```

```
class idena.types.Block
```

```
Bases: NamedTuple
```

Typed version of namedtuple.

Usage in Python versions >= 3.6:

```
class Employee(NamedTuple):
    name: str
    id: int
```

This is equivalent to:

```
Employee = collections.namedtuple('Employee', ['name', 'id'])
```

The resulting class has extra `__annotations__` and `_field_types` attributes, giving an ordered dict mapping field names to types. `__annotations__` should be preferred, while `_field_types` is kept to maintain pre PEP 526 compatibility. (The field names are in the `_fields` attribute, which is part of the namedtuple API.) Alternative equivalent keyword syntax is also accepted:

```
Employee = NamedTuple('Employee', name=str, id=int)
```

In Python versions <= 3.5 use:

```
Employee = NamedTuple('Employee', [('name', str), ('id', int)])
```

```
coinbase :str
```

```
flags :List[str]
```

```
hash :str
```

```
height :int
```

```
identityRoot :str
```

```
ipfsCid :str
```

```
isEmpty :bool
```

```
offlineAddress :str
```

```
parentHash :str
```

```
root :str
```

```
timestamp :int
```

```
transactions :List[str]
```

```
class idena.types.BurntCoins
```

```
Bases: NamedTuple
```

Typed version of namedtuple.

Usage in Python versions >= 3.6:

```
class Employee(NamedTuple):
    name: str
    id: int
```

This is equivalent to:

```
Employee = collections.namedtuple('Employee', ['name', 'id'])
```

The resulting class has extra `__annotations__` and `_field_types` attributes, giving an ordered dict mapping field names to types. `__annotations__` should be preferred, while `_field_types` is kept to maintain pre PEP 526 compatibility. (The field names are in the `_fields` attribute, which is part of the namedtuple API.) Alternative equivalent keyword syntax is also accepted:

```
Employee = NamedTuple('Employee', name=str, id=int)
```

In Python versions `<= 3.5` use:

```
Employee = NamedTuple('Employee', [('name', str), ('id', int)])
```

address :str

amount :decimal.Decimal

key :str

class idena.types.CeremonyIntervals

Bases: NamedTuple

Typed version of namedtuple.

Usage in Python versions `>= 3.6`:

```
class Employee(NamedTuple):
    name: str
    id: int
```

This is equivalent to:

```
Employee = collections.namedtuple('Employee', ['name', 'id'])
```

The resulting class has extra `__annotations__` and `_field_types` attributes, giving an ordered dict mapping field names to types. `__annotations__` should be preferred, while `_field_types` is kept to maintain pre PEP 526 compatibility. (The field names are in the `_fields` attribute, which is part of the namedtuple API.) Alternative equivalent keyword syntax is also accepted:

```
Employee = NamedTuple('Employee', name=str, id=int)
```

In Python versions `<= 3.5` use:

```
Employee = NamedTuple('Employee', [('name', str), ('id', int)])
```

FlipLotteryDuration :float

LongSessionDuration :float

ShortSessionDuration :float

class idena.types.ChangeProfile

Bases: NamedTuple

Typed version of namedtuple.

Usage in Python versions `>= 3.6`:

```
class Employee(NamedTuple):
    name: str
    id: int
```

This is equivalent to:

```
Employee = collections.namedtuple('Employee', ['name', 'id'])
```

The resulting class has extra `__annotations__` and `_field_types` attributes, giving an ordered dict mapping field names to types. `__annotations__` should be preferred, while `_field_types` is kept to maintain pre PEP 526 compatibility. (The field names are in the `_fields` attribute, which is part of the namedtuple API.) Alternative equivalent keyword syntax is also accepted:

```
Employee = NamedTuple('Employee', name=str, id=int)
```

In Python versions `<= 3.5` use:

```
Employee = NamedTuple('Employee', [('name', str), ('id', int)])
```

```
hash :str
```

```
txHash :str
```

```
class idena.types.ContractTxReceipt
```

```
contract :str
```

```
error :str
```

```
gasCost :decimal.Decimal
```

```
gasUsed :int
```

```
success :bool
```

```
txFee :decimal.Decimal
```

```
txHash :str
```

```
class idena.types.DynamicArg
```

```
Bases: NamedTuple
```

```
Typed version of namedtuple.
```

```
Usage in Python versions >= 3.6:
```

```
class Employee(NamedTuple):
    name: str
    id: int
```

This is equivalent to:

```
Employee = collections.namedtuple('Employee', ['name', 'id'])
```

The resulting class has extra `__annotations__` and `_field_types` attributes, giving an ordered dict mapping field names to types. `__annotations__` should be preferred, while `_field_types` is kept to maintain pre PEP 526 compatibility. (The field names are in the `_fields` attribute, which is part of the namedtuple API.) Alternative equivalent keyword syntax is also accepted:

```
Employee = NamedTuple('Employee', name=str, id=int)
```

In Python versions `<= 3.5` use:

```
Employee = NamedTuple('Employee', [('name', str), ('id', int)])
```

```
format :str
index :int
value :str
```

```
class idena.types.Epoch
```

Bases: NamedTuple

Typed version of namedtuple.

Usage in Python versions >= 3.6:

```
class Employee(NamedTuple):
    name: str
    id: int
```

This is equivalent to:

```
Employee = collections.namedtuple('Employee', ['name', 'id'])
```

The resulting class has extra `__annotations__` and `_field_types` attributes, giving an ordered dict mapping field names to types. `__annotations__` should be preferred, while `_field_types` is kept to maintain pre PEP 526 compatibility. (The field names are in the `_fields` attribute, which is part of the namedtuple API.) Alternative equivalent keyword syntax is also accepted:

```
Employee = NamedTuple('Employee', name=str, id=int)
```

In Python versions <= 3.5 use:

```
Employee = NamedTuple('Employee', [('name', str), ('id', int)])
```

```
currentPeriod :str
```

```
currentValidationStart :datetime.datetime
```

```
epoch :int
```

```
nextValidation :datetime.datetime
```

```
class idena.types.Flip
```

Bases: NamedTuple

Typed version of namedtuple.

Usage in Python versions >= 3.6:

```
class Employee(NamedTuple):
    name: str
    id: int
```

This is equivalent to:

```
Employee = collections.namedtuple('Employee', ['name', 'id'])
```

The resulting class has extra `__annotations__` and `_field_types` attributes, giving an ordered dict mapping field names to types. `__annotations__` should be preferred, while `_field_types` is kept to maintain pre PEP 526 compatibility. (The field names are in the `_fields` attribute, which is part of the namedtuple API.) Alternative equivalent keyword syntax is also accepted:

```
Employee = NamedTuple('Employee', name=str, id=int)
```

In Python versions <= 3.5 use:

```
Employee = NamedTuple('Employee', [('name', str), ('id', int)])
```

hex :str

privateHex :str

class idena.types.**FlipAnswer**

Bases: NamedTuple

Typed version of namedtuple.

Usage in Python versions >= 3.6:

```
class Employee(NamedTuple):
    name: str
    id: int
```

This is equivalent to:

```
Employee = collections.namedtuple('Employee', ['name', 'id'])
```

The resulting class has extra `__annotations__` and `_field_types` attributes, giving an ordered dict mapping field names to types. `__annotations__` should be preferred, while `_field_types` is kept to maintain pre PEP 526 compatibility. (The field names are in the `_fields` attribute, which is part of the namedtuple API.) Alternative equivalent keyword syntax is also accepted:

```
Employee = NamedTuple('Employee', name=str, id=int)
```

In Python versions <= 3.5 use:

```
Employee = NamedTuple('Employee', [('name', str), ('id', int)])
```

answer :Answer

grade :Grade

hash :str

wrongWords :List[bool]

class idena.types.**FlipAnswers**

Bases: NamedTuple

Typed version of namedtuple.

Usage in Python versions >= 3.6:

```
class Employee(NamedTuple):
    name: str
    id: int
```

This is equivalent to:

```
Employee = collections.namedtuple('Employee', ['name', 'id'])
```

The resulting class has extra `__annotations__` and `_field_types` attributes, giving an ordered dict mapping field names to types. `__annotations__` should be preferred, while `_field_types` is kept to maintain pre PEP 526 compatibility. (The field names are in the `_fields` attribute, which is part of the namedtuple API.) Alternative equivalent keyword syntax is also accepted:

```
Employee = NamedTuple('Employee', name=str, id=int)
```

In Python versions <= 3.5 use:

```
Employee = NamedTuple('Employee', [('name', str), ('id', int)])
```

answers :List[FlipAnswer]

class idena.types.**FlipHashes**

Bases: NamedTuple

Typed version of namedtuple.

Usage in Python versions >= 3.6:

```
class Employee(NamedTuple):
    name: str
    id: int
```

This is equivalent to:

```
Employee = collections.namedtuple('Employee', ['name', 'id'])
```

The resulting class has extra `__annotations__` and `_field_types` attributes, giving an ordered dict mapping field names to types. `__annotations__` should be preferred, while `_field_types` is kept to maintain pre PEP 526 compatibility. (The field names are in the `_fields` attribute, which is part of the namedtuple API.) Alternative equivalent keyword syntax is also accepted:

```
Employee = NamedTuple('Employee', name=str, id=int)
```

In Python versions <= 3.5 use:

```
Employee = NamedTuple('Employee', [('name', str), ('id', int)])
```

available :bool

extra :bool

hash :str

ready :bool

class idena.types.**FlipWords**

Bases: NamedTuple

Typed version of namedtuple.

Usage in Python versions >= 3.6:

```
class Employee(NamedTuple):
    name: str
    id: int
```

This is equivalent to:

```
Employee = collections.namedtuple('Employee', ['name', 'id'])
```

The resulting class has extra `__annotations__` and `_field_types` attributes, giving an ordered dict mapping field names to types. `__annotations__` should be preferred, while `_field_types` is kept to maintain pre PEP 526 compatibility. (The field names are in the `_fields` attribute, which is part of the namedtuple API.) Alternative equivalent keyword syntax is also accepted:

```
Employee = NamedTuple('Employee', name=str, id=int)
```

In Python versions <= 3.5 use:

```
Employee = NamedTuple('Employee', [('name', str), ('id', int)])
```

```
id :int
```

```
used :bool
```

```
words :List[int]
```

```
class idena.types.Grade
```

```
gradeA = 5
```

```
gradeB = 4
```

```
gradeC = 3
```

```
gradeD = 2
```

```
gradeNone = 0
```

```
gradeReported = 1
```

```
class idena.types.Identity
```

```
Bases: NamedTuple
```

```
Typed version of namedtuple.
```

```
Usage in Python versions >= 3.6:
```

```
class Employee(NamedTuple):
    name: str
    id: int
```

This is equivalent to:

```
Employee = collections.namedtuple('Employee', ['name', 'id'])
```

The resulting class has extra `__annotations__` and `_field_types` attributes, giving an ordered dict mapping field names to types. `__annotations__` should be preferred, while `_field_types` is kept to maintain pre PEP 526 compatibility. (The field names are in the `_fields` attribute, which is part of the namedtuple API.) Alternative equivalent keyword syntax is also accepted:

```
Employee = NamedTuple('Employee', name=str, id=int)
```

In Python versions <= 3.5 use:

```
Employee = NamedTuple('Employee', [('name', str), ('id', int)])
```

```
address :str
```

```
age :int
```

```
availableFlips :int
```

```
code :bytes
```

```
flipKeyWordPairs :Optional[List[FlipWords]]
```

```
flips :Optional[List[str]]
```

```
generation :int
invitees :Optional[List[TxAddr]]
invites :int
lastValidationFlags :Optional[List[str]]
madeFlips :int
online :bool
penalty :decimal.Decimal
profileHash :str
pubkey :str
requiredFlips :int
stake :decimal.Decimal
state :str
totalQualifiedFlips :int
totalShortFlipPoints :float
```

```
class idena.types.Invite
```

Bases: NamedTuple

Typed version of namedtuple.

Usage in Python versions >= 3.6:

```
class Employee(NamedTuple):
    name: str
    id: int
```

This is equivalent to:

```
Employee = collections.namedtuple('Employee', ['name', 'id'])
```

The resulting class has extra `__annotations__` and `_field_types` attributes, giving an ordered dict mapping field names to types. `__annotations__` should be preferred, while `_field_types` is kept to maintain pre PEP 526 compatibility. (The field names are in the `_fields` attribute, which is part of the namedtuple API.) Alternative equivalent keyword syntax is also accepted:

```
Employee = NamedTuple('Employee', name=str, id=int)
```

In Python versions <= 3.5 use:

```
Employee = NamedTuple('Employee', [('name', str), ('id', int)])
```

```
hash :str
```

```
key :str
```

```
receiver :str
```

```
class idena.types.Peer
```

Bases: NamedTuple

Typed version of namedtuple.

Usage in Python versions >= 3.6:

```
class Employee(NamedTuple):
    name: str
    id: int
```

This is equivalent to:

```
Employee = collections.namedtuple('Employee', ['name', 'id'])
```

The resulting class has extra `__annotations__` and `_field_types` attributes, giving an ordered dict mapping field names to types. `__annotations__` should be preferred, while `_field_types` is kept to maintain pre PEP 526 compatibility. (The field names are in the `_fields` attribute, which is part of the namedtuple API.) Alternative equivalent keyword syntax is also accepted:

```
Employee = NamedTuple('Employee', name=str, id=int)
```

In Python versions ≤ 3.5 use:

```
Employee = NamedTuple('Employee', [('name', str), ('id', int)])
```

```
addr :str
```

```
id :str
```

```
class idena.types.Profile
```

```
Bases: NamedTuple
```

Typed version of namedtuple.

Usage in Python versions ≥ 3.6 :

```
class Employee(NamedTuple):
    name: str
    id: int
```

This is equivalent to:

```
Employee = collections.namedtuple('Employee', ['name', 'id'])
```

The resulting class has extra `__annotations__` and `_field_types` attributes, giving an ordered dict mapping field names to types. `__annotations__` should be preferred, while `_field_types` is kept to maintain pre PEP 526 compatibility. (The field names are in the `_fields` attribute, which is part of the namedtuple API.) Alternative equivalent keyword syntax is also accepted:

```
Employee = NamedTuple('Employee', name=str, id=int)
```

In Python versions ≤ 3.5 use:

```
Employee = NamedTuple('Employee', [('name', str), ('id', int)])
```

```
info :bytes
```

```
nickname :str
```

```
class idena.types.RawFlip
```

```
Bases: NamedTuple
```

Typed version of namedtuple.

Usage in Python versions ≥ 3.6 :

```
class Employee(NamedTuple):
    name: str
    id: int
```

This is equivalent to:

```
Employee = collections.namedtuple('Employee', ['name', 'id'])
```

The resulting class has extra `__annotations__` and `_field_types` attributes, giving an ordered dict mapping field names to types. `__annotations__` should be preferred, while `_field_types` is kept to maintain pre PEP 526 compatibility. (The field names are in the `_fields` attribute, which is part of the namedtuple API.) Alternative equivalent keyword syntax is also accepted:

```
Employee = NamedTuple('Employee', name=str, id=int)
```

In Python versions `<= 3.5` use:

```
Employee = NamedTuple('Employee', [('name', str), ('id', int)])
```

```
privateHex :str
publicHex :str
class idena.types.SendTxArgs
    Bases: BaseTxArgs
    docstring
    amount :decimal.Decimal
    from_ :str
    maxFee :decimal.Decimal
    payload :str
    tips :decimal.Decimal
    to :str
    type :TxType
    useProto :bool
```

```
class idena.types.Stake
    Bases: NamedTuple
    Typed version of namedtuple.
    Usage in Python versions >= 3.6:
```

```
class Employee(NamedTuple):
    name: str
    id: int
```

This is equivalent to:

```
Employee = collections.namedtuple('Employee', ['name', 'id'])
```

The resulting class has extra `__annotations__` and `_field_types` attributes, giving an ordered dict mapping field names to types. `__annotations__` should be preferred, while `_field_types` is kept to maintain pre PEP 526

compatibility. (The field names are in the `_fields` attribute, which is part of the namedtuple API.) Alternative equivalent keyword syntax is also accepted:

```
Employee = namedtuple('Employee', name=str, id=int)
```

In Python versions `<= 3.5` use:

```
Employee = namedtuple('Employee', [('name', str), ('id', int)])
```

Hash :str

Stake :decimal.Decimal

class `idena.types.State`

Bases: namedtuple

Typed version of namedtuple.

Usage in Python versions `>= 3.6`:

```
class Employee(namedtuple):
    name: str
    id: int
```

This is equivalent to:

```
Employee = collections.namedtuple('Employee', ['name', 'id'])
```

The resulting class has extra `__annotations__` and `_field_types` attributes, giving an ordered dict mapping field names to types. `__annotations__` should be preferred, while `_field_types` is kept to maintain pre PEP 526 compatibility. (The field names are in the `_fields` attribute, which is part of the namedtuple API.) Alternative equivalent keyword syntax is also accepted:

```
Employee = namedtuple('Employee', name=str, id=int)
```

In Python versions `<= 3.5` use:

```
Employee = namedtuple('Employee', [('name', str), ('id', int)])
```

name :str

class `idena.types.Sync`

Bases: namedtuple

Typed version of namedtuple.

Usage in Python versions `>= 3.6`:

```
class Employee(namedtuple):
    name: str
    id: int
```

This is equivalent to:

```
Employee = collections.namedtuple('Employee', ['name', 'id'])
```

The resulting class has extra `__annotations__` and `_field_types` attributes, giving an ordered dict mapping field names to types. `__annotations__` should be preferred, while `_field_types` is kept to maintain pre PEP 526 compatibility. (The field names are in the `_fields` attribute, which is part of the namedtuple API.) Alternative equivalent keyword syntax is also accepted:

```
Employee = NamedTuple('Employee', name=str, id=int)
```

In Python versions <= 3.5 use:

```
Employee = NamedTuple('Employee', [('name', str), ('id', int)])
```

```
currentBlock :int
genesisBlock :int
highestBlock :int
syncing :bool
wrongTime :bool
```

class idena.types.Transaction

Bases: NamedTuple

Typed version of namedtuple.

Usage in Python versions >= 3.6:

```
class Employee(NamedTuple):
    name: str
    id: int
```

This is equivalent to:

```
Employee = collections.namedtuple('Employee', ['name', 'id'])
```

The resulting class has extra `__annotations__` and `_field_types` attributes, giving an ordered dict mapping field names to types. `__annotations__` should be preferred, while `_field_types` is kept to maintain pre PEP 526 compatibility. (The field names are in the `_fields` attribute, which is part of the namedtuple API.) Alternative equivalent keyword syntax is also accepted:

```
Employee = NamedTuple('Employee', name=str, id=int)
```

In Python versions <= 3.5 use:

```
Employee = NamedTuple('Employee', [('name', str), ('id', int)])
```

```
amount :str
blockHash :str
epoch :int
from_ :str
hash :str
maxFee :str
nonce :int
payload :str
timestamp :int
tips :str
to :Optional[str]
```

```

type :str
usedFee :str
class idena.types.TxAddr
    Bases: NamedTuple
    Typed version of namedtuple.
    Usage in Python versions >= 3.6:

```

```

class Employee(NamedTuple):
    name: str
    id: int

```

This is equivalent to:

```
Employee = collections.namedtuple('Employee', ['name', 'id'])
```

The resulting class has extra `__annotations__` and `_field_types` attributes, giving an ordered dict mapping field names to types. `__annotations__` should be preferred, while `_field_types` is kept to maintain pre PEP 526 compatibility. (The field names are in the `_fields` attribute, which is part of the namedtuple API.) Alternative equivalent keyword syntax is also accepted:

```
Employee = NamedTuple('Employee', name=str, id=int)
```

In Python versions <= 3.5 use:

```
Employee = NamedTuple('Employee', [('name', str), ('id', int)])
```

```

Address :str
TxHash :str
class idena.types.TxEvent
    Bases: NamedTuple
    Typed version of namedtuple.
    Usage in Python versions >= 3.6:

```

```

class Employee(NamedTuple):
    name: str
    id: int

```

This is equivalent to:

```
Employee = collections.namedtuple('Employee', ['name', 'id'])
```

The resulting class has extra `__annotations__` and `_field_types` attributes, giving an ordered dict mapping field names to types. `__annotations__` should be preferred, while `_field_types` is kept to maintain pre PEP 526 compatibility. (The field names are in the `_fields` attribute, which is part of the namedtuple API.) Alternative equivalent keyword syntax is also accepted:

```
Employee = NamedTuple('Employee', name=str, id=int)
```

In Python versions <= 3.5 use:

```
Employee = NamedTuple('Employee', [('name', str), ('id', int)])
```

```
data :List[bytes]
```

```
eventName :str
```

```
class idena.types.TxReceipt
```

```
    Bases: NamedTuple
```

Typed version of namedtuple.

Usage in Python versions >= 3.6:

```
class Employee(NamedTuple):
    name: str
    id: int
```

This is equivalent to:

```
Employee = collections.namedtuple('Employee', ['name', 'id'])
```

The resulting class has extra `__annotations__` and `_field_types` attributes, giving an ordered dict mapping field names to types. `__annotations__` should be preferred, while `_field_types` is kept to maintain pre PEP 526 compatibility. (The field names are in the `_fields` attribute, which is part of the namedtuple API.) Alternative equivalent keyword syntax is also accepted:

```
Employee = NamedTuple('Employee', name=str, id=int)
```

In Python versions <= 3.5 use:

```
Employee = NamedTuple('Employee', [('name', str), ('id', int)])
```

```
contractAddress :str
```

```
error :str
```

```
events :List[TxEvent]
```

```
from_ :str
```

```
gasCost :int
```

```
gasUsed :int
```

```
success :bool
```

```
txHash :str
```

```
class idena.types.TxType
```

```
    ActivationTx = 1
```

```
    BurnTx = 12
```

```
    ChangeGodAddressTx = 11
```

```
    ChangeProfileTx = 13
```

```
    EvidenceTx = 8
```

```
    InviteTx = 2
```

```
    KillInviteeTx = 10
```

```
    KillTx = 3
```

```
    OnlineStatusTx = 9
```

```
    SendTx = 0
```

```
SubmitAnswersHashTx = 5
```

```
SubmitFlipTx = 4
```

```
SubmitLongAnswersTx = 7
```

```
SubmitShortAnswersTx = 6
```

- Highly inspired by [Web3.py](#) and [cookiecutter-pypackage](#).

INSTALLATION

Idena.py can be installed using `pip` as follows:

```
$ pip install idena
```

For the development, clone the repository then:

```
$ pip install -e .
```


USING IDENA

This library depends on a connection to an Idena node and there are 2 ways to configure them.

3.1 Calling *client.init*

```
>>> from idena import client
>>> client.init('http://localhost:9009/', 'api-key')
```

3.2 Setting environment variables

Set *IDENA_RPC_NODE* and *IDENA_API_KEY* envvars:

```
$ export IDENA_RPC_NODE=http://localhost:9009/
$ export IDENA_API_KEY=api-key
```


GETTING BLOCKCHAIN INFO

```
>>> client.blockchain.get_last_block()

Block(coinbase='0xbe854231db69ab042073b7ff8309ae3ee265a40f',
      hash='0xa88e6ab305d7ee311ad2de35338cdbf7e664d860709e5a53f0307baeaa6f968',
      parentHash='0xe324a208892241e0294e5a6334965660375dda7a3ad8d8a42a5f3f2ef2857a22',
      height=2159398,
      timestamp=1606904853,
      root='0xd874709bdd4c6fcd95e2e531cc07a4ce42ab23334dfd12ceb45350535b36664c',
      identityRoot='0x9f3661f19e13d4860f2f2f1610abbbaf86abc8adf1a2781b371189e683745a97',
      ipfsCid=None,
      transactions=None,
      flags=['OfflinePropose'],
      isEmpty=False,
      offlineAddress='0x0df427ad7e1906ab4fcc5fd31118932256f5dc7a')
```


INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

i

- idena, [3](#)
- idena.apis, [3](#)
- idena.apis.account, [3](#)
- idena.apis.blockchain, [4](#)
- idena.apis.contract, [6](#)
- idena.apis.dna, [9](#)
- idena.apis.flip, [13](#)
- idena.apis.net, [14](#)
- idena.client, [14](#)
- idena.exceptions, [15](#)
- idena.types, [15](#)

Symbols

`__repr__()` (*idena.exceptions.IdenaException* method), 15

A

`activate_invite()` (*in module idena.apis.dna*), 10
`activate_invite_to_random_address()` (*in module idena.apis.dna*), 10
`ActivateInviteToRandomAddr` (*class in idena.types*), 16
`ActivationTx` (*idena.types.TxType* attribute), 32
`add_peers()` (*in module idena.apis.net*), 14
`addr` (*idena.types.Peer* attribute), 27
`address` (*idena.types.ActivateInviteToRandomAddr* attribute), 16
`address` (*idena.types.BurntCoins* attribute), 20
`address` (*idena.types.Identity* attribute), 25
`Address` (*idena.types.TxAddr* attribute), 31
`AddressTransactions` (*class in idena.types*), 17
`age` (*idena.types.Identity* attribute), 25
`amount` (*idena.types.BurntCoins* attribute), 20
`amount` (*idena.types.SendTxArgs* attribute), 28
`amount` (*idena.types.Transaction* attribute), 30
`Answer` (*class in idena.types*), 17
`answer` (*idena.types.FlipAnswer* attribute), 23
`answers` (*idena.types.FlipAnswers* attribute), 24
`available` (*idena.types.FlipHashes* attribute), 24
`availableFlips` (*idena.types.Identity* attribute), 25

B

`Balance` (*class in idena.types*), 18
`balance` (*idena.types.Balance* attribute), 18
`BaseTxArgs` (*class in idena.types*), 18
`become_offline()` (*in module idena.apis.dna*), 10
`become_online()` (*in module idena.apis.dna*), 11
`Block` (*class in idena.types*), 19
`blockHash` (*idena.types.Transaction* attribute), 30
`burn()` (*in module idena.apis.dna*), 11
`BurntCoins` (*class in idena.types*), 19
`BurnTx` (*idena.types.TxType* attribute), 32

C

`call_contract()` (*in module idena.apis.contract*), 6
`CeremonyIntervals` (*class in idena.types*), 20
`change_god_address` (*in module idena.apis.dna*), 10
`change_profile()` (*in module idena.apis.dna*), 11
`ChangeGodAddressTx` (*idena.types.TxType* attribute), 32
`ChangeProfile` (*class in idena.types*), 20
`ChangeProfileTx` (*idena.types.TxType* attribute), 32
`check_sync()` (*in module idena.apis.blockchain*), 4
`code` (*idena.types.Identity* attribute), 25
`coinbase` (*idena.types.Block* attribute), 19
`contract` (*idena.types.ContractTxReceipt* attribute), 21
`contractAddress` (*idena.types.TxReceipt* attribute), 32
`ContractTxReceipt` (*class in idena.types*), 21
`create_account()` (*in module idena.apis.account*), 3
`currentBlock` (*idena.types.Sync* attribute), 30
`currentPeriod` (*idena.types.Epoch* attribute), 22
`currentValidationStart` (*idena.types.Epoch* attribute), 22

D

`data` (*idena.types.TxEvent* attribute), 31
`delete_flip()` (*in module idena.apis.flip*), 13
`deploy_contract()` (*in module idena.apis.contract*), 7
`DynamicArg` (*class in idena.types*), 21

E

`Epoch` (*class in idena.types*), 22
`epoch` (*idena.types.BaseTxArgs* attribute), 18
`epoch` (*idena.types.Epoch* attribute), 22
`epoch` (*idena.types.Transaction* attribute), 30
`error` (*idena.types.ContractTxReceipt* attribute), 21
`error` (*idena.types.TxReceipt* attribute), 32
`estimate_call()` (*in module idena.apis.contract*), 7
`estimate_deploy()` (*in module idena.apis.contract*), 7

estimate_terminate() (in module idena.apis.contract), 7
 eventName (idenatypes.TxEvent attribute), 32
 events (idenatypes.TxReceipt attribute), 32
 EvidenceTx (idenatypes.TxType attribute), 32
 export_key() (in module idena.apis.dna), 11
 extra (idenatypes.FlipHashes attribute), 24

F

flags (idenatypes.Block attribute), 19
 Flip (class in idenatypes), 22
 flip_words() (in module idena.apis.flip), 13
 FlipAnswer (class in idenatypes), 23
 FlipAnswers (class in idenatypes), 23
 FlipHashes (class in idenatypes), 24
 flipKeywordPairs (idenatypes.Identity attribute), 25
 FlipLotteryDuration (idenatypes.CeremonyIntervals attribute), 20
 flips (idenatypes.Identity attribute), 25
 FlipSubmit (in module idenatypes), 16
 FlipWords (class in idenatypes), 24
 format (idenatypes.DynamicArg attribute), 21
 from_ (idenatypes.SendTxArgs attribute), 28
 from_ (idenatypes.Transaction attribute), 30
 from_ (idenatypes.TxReceipt attribute), 32

G

gasCost (idenatypes.ContractTxReceipt attribute), 21
 gasCost (idenatypes.TxReceipt attribute), 32
 gasUsed (idenatypes.ContractTxReceipt attribute), 21
 gasUsed (idenatypes.TxReceipt attribute), 32
 generation (idenatypes.Sync attribute), 26
 genesisBlock (idenatypes.Sync attribute), 30
 get_accounts() (in module idena.apis.account), 3
 get_address_pending_transactions() (in module idena.apis.blockchain), 4
 get_address_transactions() (in module idena.apis.blockchain), 5
 get_balance() (in module idena.apis.dna), 11
 get_block_by_hash() (in module idena.apis.blockchain), 5
 get_block_by_height() (in module idena.apis.blockchain), 5
 get_burnt_coins() (in module idena.apis.blockchain), 5
 get_ceremony_intervals() (in module idena.apis.dna), 11
 get_coinbase_address() (in module idena.apis.dna), 11
 get_current_process() (in module idena.apis.dna), 11
 get_epoch() (in module idena.apis.dna), 11
 get_fee_rate() (in module idena.apis.blockchain), 5
 get_flip() (in module idena.apis.flip), 13
 get_identities() (in module idena.apis.dna), 11
 get_identity() (in module idena.apis.dna), 12
 get_ipfs_addr() (in module idena.apis.net), 14
 get_last_block() (in module idena.apis.blockchain), 5
 get_long_flip_hashes() (in module idena.apis.flip), 13
 get_mempool() (in module idena.apis.blockchain), 5
 get_peers() (in module idena.apis.net), 14
 get_profile() (in module idena.apis.dna), 12
 get_raw_flip() (in module idena.apis.flip), 13
 get_raw_tx() (in module idena.apis.blockchain), 5
 get_short_flip_hashes() (in module idena.apis.flip), 13
 get_stake() (in module idena.apis.contract), 8
 get_transaction() (in module idena.apis.blockchain), 5
 get_transaction_receipt() (in module idena.apis.blockchain), 6
 Grade (class in idenatypes), 25
 grade (idenatypes.FlipAnswer attribute), 23
 gradeA (idenatypes.Grade attribute), 25
 gradeB (idenatypes.Grade attribute), 25
 gradeC (idenatypes.Grade attribute), 25
 gradeD (idenatypes.Grade attribute), 25
 gradeNone (idenatypes.Grade attribute), 25
 gradeReported (idenatypes.Grade attribute), 25

H

hash (idenatypes.ActivateInviteToRandomAddr attribute), 16
 hash (idenatypes.Block attribute), 19
 hash (idenatypes.ChangeProfile attribute), 21
 hash (idenatypes.FlipAnswer attribute), 23
 hash (idenatypes.FlipHashes attribute), 24
 hash (idenatypes.Invite attribute), 26
 Hash (idenatypes.Stake attribute), 29
 hash (idenatypes.Transaction attribute), 30
 height (idenatypes.Block attribute), 19
 hex (idenatypes.Flip attribute), 23
 highestBlock (idenatypes.Sync attribute), 30

I

id (idenatypes.FlipWords attribute), 25
 id (idenatypes.Peer attribute), 27
 idena
 module, 3
 idena.apis
 module, 3
 idena.apis.account
 module, 3

idena.apis.blockchain
 module, 4
 idena.apis.contract
 module, 6
 idena.apis.dna
 module, 9
 idena.apis.flip
 module, 13
 idena.apis.net
 module, 14
 idena.client
 module, 14
 idena.exceptions
 module, 15
 idena.types
 module, 15
 IdenaException, 15
 Identity (class in idena.types), 25
 identityRoot (idenatypes.Block attribute), 19
 index (idenatypes.DynamicArg attribute), 22
 info (idenatypes.Profile attribute), 27
 init() (in module idena.client), 15
 Invite (class in idena.types), 26
 invitees (idenatypes.Identity attribute), 26
 invites (idenatypes.Identity attribute), 26
 InviteTx (idenatypes.TxType attribute), 32
 ipfsCid (idenatypes.Block attribute), 19
 isEmpty (idenatypes.Block attribute), 19
 isValidatationReady() (in module idena.apis.dna), 12

K

key (idenatypes.ActivateInviteToRandomAddr attribute), 16
 key (idenatypes.BurntCoins attribute), 20
 key (idenatypes.Invite attribute), 26
 kill_identity (in module idena.apis.dna), 10
 kill_invitee (in module idena.apis.dna), 10
 KillInviteeTx (idenatypes.TxType attribute), 32
 KillTx (idenatypes.TxType attribute), 32

L

lastValidationFlags (idenatypes.Identity attribute), 26
 left (idenatypes.Answer attribute), 17
 lock_account() (in module idena.apis.account), 3
 LongSessionDuration
 (idenatypes.CeremonyIntervals attribute), 20

M

madeFlips (idenatypes.Identity attribute), 26
 maxFee (idenatypes.SendTxArgs attribute), 28
 maxFee (idenatypes.Transaction attribute), 30

module
 idenatypes, 3
 idenatypes.apis, 3
 idenatypes.apis.account, 3
 idenatypes.apis.blockchain, 4
 idenatypes.apis.contract, 6
 idenatypes.apis.dna, 9
 idenatypes.apis.flip, 13
 idenatypes.apis.net, 14
 idenatypes.client, 14
 idenatypes.exceptions, 15
 idenatypes.types, 15

N

name (idenatypes.State attribute), 29
 nextValidation (idenatypes.Epoch attribute), 22
 nickname (idenatypes.Profile attribute), 27
 nonce (idenatypes.Balance attribute), 18
 nonce (idenatypes.BaseTxArgs attribute), 19
 nonce (idenatypes.Transaction attribute), 30
 none (idenatypes.Answer attribute), 17

O

offlineAddress (idenatypes.Block attribute), 19
 online (idenatypes.Identity attribute), 26
 OnlineStatusTx (idenatypes.TxType attribute), 32

P

parentHash (idenatypes.Block attribute), 19
 payload (idenatypes.SendTxArgs attribute), 28
 payload (idenatypes.Transaction attribute), 30
 Peer (class in idena.types), 26
 penalty (idenatypes.Identity attribute), 26
 privateHex (idenatypes.Flip attribute), 23
 privateHex (idenatypes.RawFlip attribute), 28
 Profile (class in idena.types), 27
 profileHash (idenatypes.Identity attribute), 26
 pubkey (idenatypes.Identity attribute), 26
 publicHex (idenatypes.RawFlip attribute), 28

R

RawFlip (class in idena.types), 27
 read_contract_data() (in module idena.apis.contract), 8
 read_events() (in module idena.apis.contract), 8
 readonly_call_contract() (in module idena.apis.contract), 8
 ready (idenatypes.FlipHashes attribute), 24
 receiver (idenatypes.Invite attribute), 26
 requiredFlips (idenatypes.Identity attribute), 26
 right (idenatypes.Answer attribute), 18
 root (idenatypes.Block attribute), 19

S

`send_dna` (in module *idena.apis.dna*), 10
`send_invite` () (in module *idena.apis.dna*), 12
`send_raw_tx` () (in module *idena.apis.blockchain*), 6
`send_tx` () (in module *idena.apis.dna*), 12
`SendTx` (*idena.types.TxType* attribute), 32
`SendTxArgs` (class in *idena.types*), 28
`ShortSessionDuration`
 (*idena.types.CeremonyIntervals* attribute), 20
`Stake` (class in *idena.types*), 28
`stake` (*idena.types.Balance* attribute), 18
`stake` (*idena.types.Identity* attribute), 26
`Stake` (*idena.types.Stake* attribute), 29
`State` (class in *idena.types*), 29
`state` (*idena.types.Identity* attribute), 26
`submit_flip` () (in module *idena.apis.flip*), 13
`submit_long_answers` () (in module *idena.apis.flip*), 14
`submit_short_answers` () (in module *idena.apis.flip*), 14
`SubmitAnswersHashTx` (*idena.types.TxType* attribute), 32
`SubmitFlipTx` (*idena.types.TxType* attribute), 33
`SubmitLongAnswersTx` (*idena.types.TxType* attribute), 33
`SubmitShortAnswersTx` (*idena.types.TxType* attribute), 33
`subscribe_to_contract_event` () (in module *idena.apis.contract*), 8
`success` (*idena.types.ContractTxReceipt* attribute), 21
`success` (*idena.types.TxReceipt* attribute), 32
`Sync` (class in *idena.types*), 29
`syncing` (*idena.types.Sync* attribute), 30

T

`terminate_contract` () (in module *idena.apis.contract*), 8
`timestamp` (*idena.types.Block* attribute), 19
`timestamp` (*idena.types.Transaction* attribute), 30
`tips` (*idena.types.SendTxArgs* attribute), 28
`tips` (*idena.types.Transaction* attribute), 30
`to` (*idena.types.SendTxArgs* attribute), 28
`to` (*idena.types.Transaction* attribute), 30
`token` (*idena.types.AddressTransactions* attribute), 17
`totalQualifiedFlips` (*idena.types.Identity* attribute), 26
`totalShortFlipPoints` (*idena.types.Identity* attribute), 26
`Transaction` (class in *idena.types*), 30
`transactions` (*idena.types.AddressTransactions* attribute), 17
`transactions` (*idena.types.Block* attribute), 19
`TxAddr` (class in *idena.types*), 31

`TxEvent` (class in *idena.types*), 31
`txFee` (*idena.types.ContractTxReceipt* attribute), 21
`txHash` (*idena.types.ChangeProfile* attribute), 21
`txHash` (*idena.types.ContractTxReceipt* attribute), 21
`txHash` (*idena.types.TxAddr* attribute), 31
`txHash` (*idena.types.TxReceipt* attribute), 32
`TxReceipt` (class in *idena.types*), 32
`TxType` (class in *idena.types*), 32
`type` (*idena.types.SendTxArgs* attribute), 28
`type` (*idena.types.Transaction* attribute), 30

U

`unlock_account` () (in module *idena.apis.account*), 4
`unsubscribe_from_contract_event` () (in module *idena.apis.contract*), 9
`used` (*idena.types.FlipWords* attribute), 25
`usedFee` (*idena.types.Transaction* attribute), 31
`useProto` (*idena.types.SendTxArgs* attribute), 28

V

`value` (*idena.types.DynamicArg* attribute), 22

W

`words` (*idena.types.FlipWords* attribute), 25
`wrongTime` (*idena.types.Sync* attribute), 30
`wrongWords` (*idena.types.FlipAnswer* attribute), 23