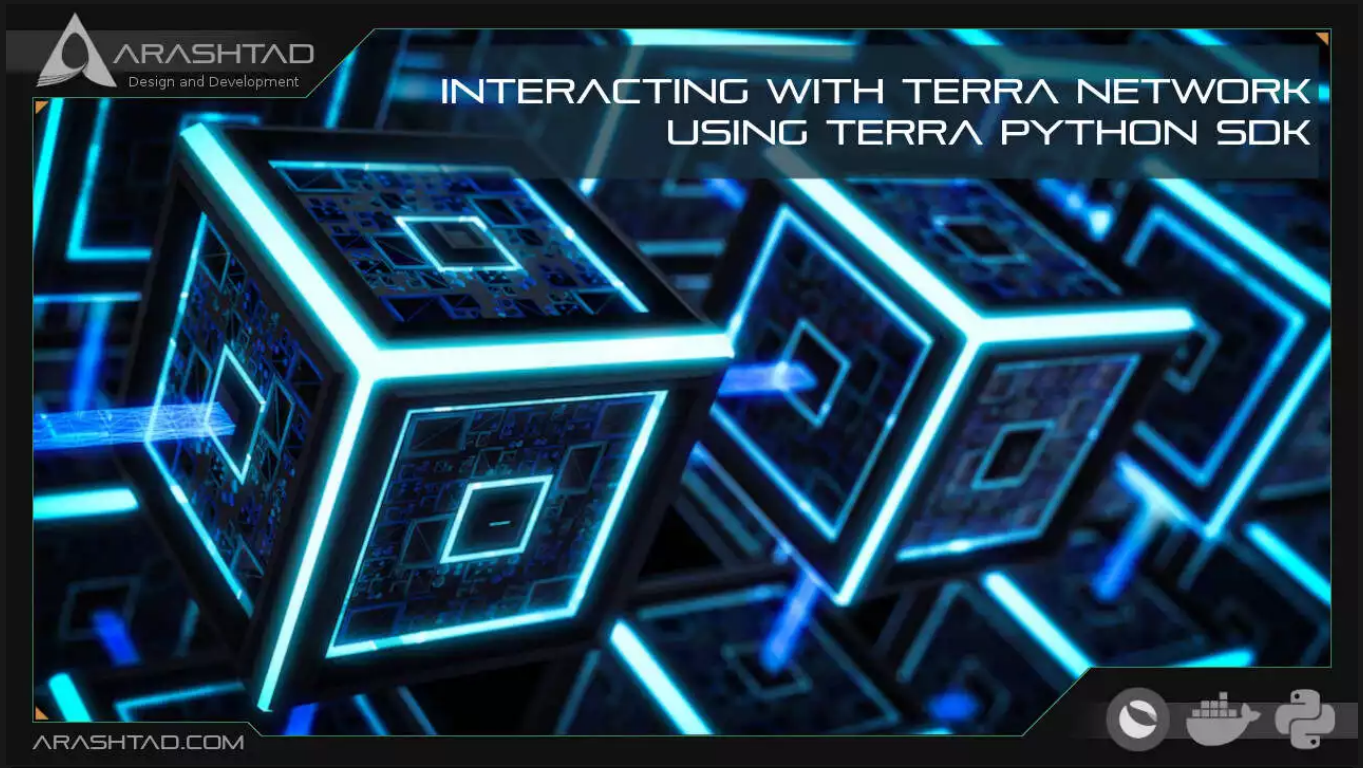
Interacting with Terra Network Using Terra Python SDK

No comments



In this article, we are going to get familiar with the Terra network and how it works. Then, we will see how we can interact with the Terra network using Terra Python SDK to interact with Terra Testnet called Bombay 12 and Mainnet called Columbus-5. In this article and the next ones, we are going to see how we can connect to the Terra network using Python, create an account and a wallet, create and sign a transaction, estimate gas fees, and swap tokens.

What Is Terra SDK?

Terra is a DeFi protocol on Blockchain that powers the universal payment systems which are based on fiat currencies using the fiat-based stable coins. This protocol uses a mixture of wide adoption of stable coins and at the same time censorship resistance of Bitcoin. Terra is built on Cosmos SDK and tendermint. The main language that the Terra smart contracts are written in, is Rust. However, there are SDKs for Python and JavaScript, and creating the smart contracts called WASM contracts is explained in more detail in JavaScript. But don't worry most of the interactions and useful functions can be found in Python. Throughout this tutorial, we are going to see all the things we can do to develop the

Terra network using Terra.py SDK.

Dealing with Terra Smart Contracts Using Terra Python SDK

If you have already worked with Ethereum smart contracts or have read our content on brownie, solidity, and Ethereum network smart contracts, you will so easily understand most of the concepts and tools that we are going to talk about. However, if you haven't, you can still follow along with this tutorial and get familiar with the new world of smart contracts. As always, like all the blockchains that we have interacted with, we start with the test network to get used to the scripts and then learn how to switch to mainnet. To get started with Terra python SDK, let us install it using the following command:

pip3 install terra-sdk

There are also other requirements that we will cover later in our tutorial. Now let's connect to Terra network:

```
terra = LCDClient("https://lcd.terra.dev", "columbus-5")
print(terra.tendermint.node_info())
```

Result:

{'node_info': {'protocol_version': {'p2p': '8', 'block': '11', 'app': '0'}, 'id': '979a7f9c646623535ba2ddc69c10f3d505ccd97a', 'listen_addr': 'tcp://0.0.0.0:26656', 'network': 'columbus-5', 'version': '0.34.14', 'channels': '40202122233038606100', 'moniker': 'rpc5-read4', 'other': {'tx_index': 'on', 'rpc_address': 'tcp://0.0.0.0:26657'}}, 'application_version': {'name': 'terra', 'server_name': 'terrad', 'version': '0.5.17-36- gfc7ef220', 'commit': 'fc7ef220ddcd683d8e1a37a3ac484abb4e8b88ef', 'build_tags': 'netgo,ledger', 'go': 'go version go1.18 linux/amd64', 'build_deps': ['filippo.io/edwards25519@v1.0.0-beta.2', 'github.com/99designs/keyring@v1.1.6 => github.com/cosmos/keyring@v1.1.7-0.20210622111912-ef00f8ac3d76', 'github.com/ChainSafe/go-schnorrkel@v0.0.0-20200405005733-88cbf1b4c40d', 'github.com/CosmWasm/wasmvm@v0.16.6', 'github.com/Workiva/godatastructures@v1.0.52', 'github.com/armon/go-metrics@v0.3.9', 'github.com/beorn7/perks@v1.0.1', 'github.com/bgentry/speakeasy@v0.1.0', 'github.com/btcsuite/btcd@v0.22.0-beta', 'github.com/cespare/xxhash/v2@v2.1.1', 'github.com/coinbase/rosetta-sdk-go@v0.7.0', 'github.com/confio/ics23/go@v0.6.6', 'github.com/cosmos/btcutil@v1.0.4', 'github.com/cosmos/cosmos-sdk@v0.44.5 => github.com/terra-money/cosmos-sdk@v0.44.5-public.2', 'github.com/cosmos/gobip39@v1.0.0', 'github.com/cosmos/iavl@v0.17.3', 'github.com/cosmos/ibcgo@v1.1.5', 'github.com/cosmos/ledger-cosmos-go@v0.11.1 => github.com/terramoney/ledger-terra-go@v0.11.2', 'github.com/cosmos/ledger-go@v0.9.2', 'github.com/davecgh/go-spew@v1.1.1', 'github.com/desertbit/timer@v0.0.0- 20180107155436-c41aec40b27f', 'github.com/dvsekhvalnov/jose2go@v0.0.0- 20200901110807-248326c1351b', 'github.com/felixge/httpsnoop@v1.0.1', 'github.com/fsnotify/fsnotify@v1.5.1', 'github.com/go-kit/kit@v0.10.0', 'github.com/go-logfmt/logfmt@v0.5.0', 'github.com/godbus/dbus@v0.0.0- 20190726142602-4481cbc300e2', 'github.com/gogo/gateway@v1.1.0', 'github.com/gogo/protobuf@v1.3.3 => github.com/regen-network/protobuf@v1.3.3- alpha.regen.1', 'github.com/golang/protobuf@v1.5.2', 'github.com/golang/snappy@v0.0.4', 'github.com/google/btree@v1.0.0', 'github.com/google/orderedcode@v0.0.1',

'github.com/gorilla/handlers@v1.5.1', 'github.com/gorilla/mux@v1.8.0', 'github.com/gorilla/websocket@v1.4.2', 'github.com/grpc-ecosystem/go-grpc-middleware@v1.3.0', 'github.com/grpcecosystem/grpc-gateway@v1.16.0', 'github.com/gsterjov/go-libsecret@v0.0.0- 20161001094733-a6f4afe4910c', 'github.com/gtank/merlin@v0.1.1', 'github.com/gtank/ristretto255@v0.1.2', 'github.com/hashicorp/go-immutableradix@v1.0.0', 'github.com/hashicorp/golang-lru@v0.5.4', 'github.com/hashicorp/hcl@v1.0.0', 'github.com/hdevalence/ed25519consensus@v0.0.0-20210204194344- 59a8610d2b87', 'github.com/improbable-eng/grpc-web@v0.14.1', 'github.com/klauspost/compress@v1.11.7', 'github.com/lib/pq@v1.10.2', 'github.com/libp2p/go-buffer-pool@v0.0.2', 'github.com/magiconair/properties@v1.8.5', 'github.com/mattn/go-isatty@v0.0.14', 'github.com/matttproud/golang_protobuf_extensions@v1.0.1', 'github.com/mimoo/StrobeGo@v0.0.0-20181016162300-f8f6d4d2b643', 'github.com/minio/highwayhash@v1.0.1', 'github.com/mitchellh/mapstructure@v1.4.2', 'github.com/mtibben/percent@v0.2.1', 'github.com/pelletier/go-toml@v1.9.4', 'github.com/pkg/errors@v0.9.1', 'github.com/pmezard/go-difflib@v1.0.0', 'github.com/prometheus/client_golang@v1.11.0', 'github.com/prometheus/client_model@v0.2.0', 'github.com/prometheus/common@v0.29.0', 'github.com/prometheus/procfs@v0.6.0', 'github.com/rakyll/statik@v0.1.7', 'github.com/rcrowley/go-metrics@v0.0.0- 20200313005456-10cdbea86bc0', 'github.com/regen-network/cosmos-proto@v0.3.1', 'github.com/rs/cors@v1.7.0', 'github.com/rs/zerolog@v1.23.0', 'github.com/spf13/afero@v1.6.0', 'github.com/spf13/cast@v1.3.1', 'github.com/spf13/cobra@v1.2.1', 'github.com/spf13/jwalterweatherman@v1.1.0', 'github.com/spf13/pflag@v1.0.5', 'github.com/spf13/viper@v1.8.1', 'github.com/stretchr/testify@v1.7.0', 'github.com/subosito/gotenv@v1.2.0', 'github.com/syndtr/goleveldb@v1.0.1-0.20210819022825-2ae1ddf74ef7', 'github.com/tendermint/btcd@v0.1.1', 'github.com/tendermint/crypto@v0.0.0-20191022145703-50d29ede1e15', 'github.com/tendermint/go-amino@v0.16.0', 'github.com/tendermint/tendermint@v0.34.14 => github.com/terra-money/tendermint@v0.34.14-performance.1', 'github.com/tendermint/tm-db@v0.6.6 => github.com/terra-money/tm-db@v0.6.4- performance.7', 'github.com/zondax/hid@v0.9.0', 'golang.org/x/crypto@v0.0.0- 20210817164053-32db794688a5', 'golang.org/x/net@v0.0.0-20210903162142- ad29c8ab022f', 'golang.org/x/sys@v0.0.0-20210903071746-97244b99971b', 'golang.org/x/term@v0.0.0-20201126162022-7de9c90e9dd1', 'golang.org/x/text@v0.3.6', 'google.golang.org/genproto@v0.0.0-20210828152312- 66f60bf46e71', 'google.golang.org/grpc@v1.44.0 => google.golang.org/grpc@v1.33.2', 'google.golang.org/protobuf@v1.27.1', 'gopkg.in/ini.v1@v1.63.2', 'gopkg.in/yaml.v2@v2.4.0', 'gopkg.in/yaml.v3@v3.0.0- 20210107192922-496545a6307b', 'nhooyr.io/websocket@v1.8.6'], 'cosmos_sdk_version': 'v0.44.5'}}

For the first step, we can create an account by creating a mnemonic key:

```
from terra_sdk.client.lcd import LCDClient
from terra_sdk.key.mnemonic import MnemonicKey
mk = MnemonicKey()
print(mk.mnemonic)
```

Now, let's test run the code:

python3 Test.py

Result:

tonight panda own all coconut home tackle gap food uncle vacant account clap napkin boring mango cable funny tooth vanish opinion genre differ impact

Notice that the above mnemonic key works the same as the private key and you will need to save it somewhere safe. Because if you forget it, you will never be able to recover the phrases. Besides, do not pass this mnemonic key to anyone. Because if you have money in it, they will be able to withdraw it anytime they want. We can get the account address by:

```
print(mk.acc_address)
```

Result:

terra1krrcxa8ccs7dwagtuevt2m4d78z0uwf4qe25yu

And the private key by:

```
print(mk.private_key)
```

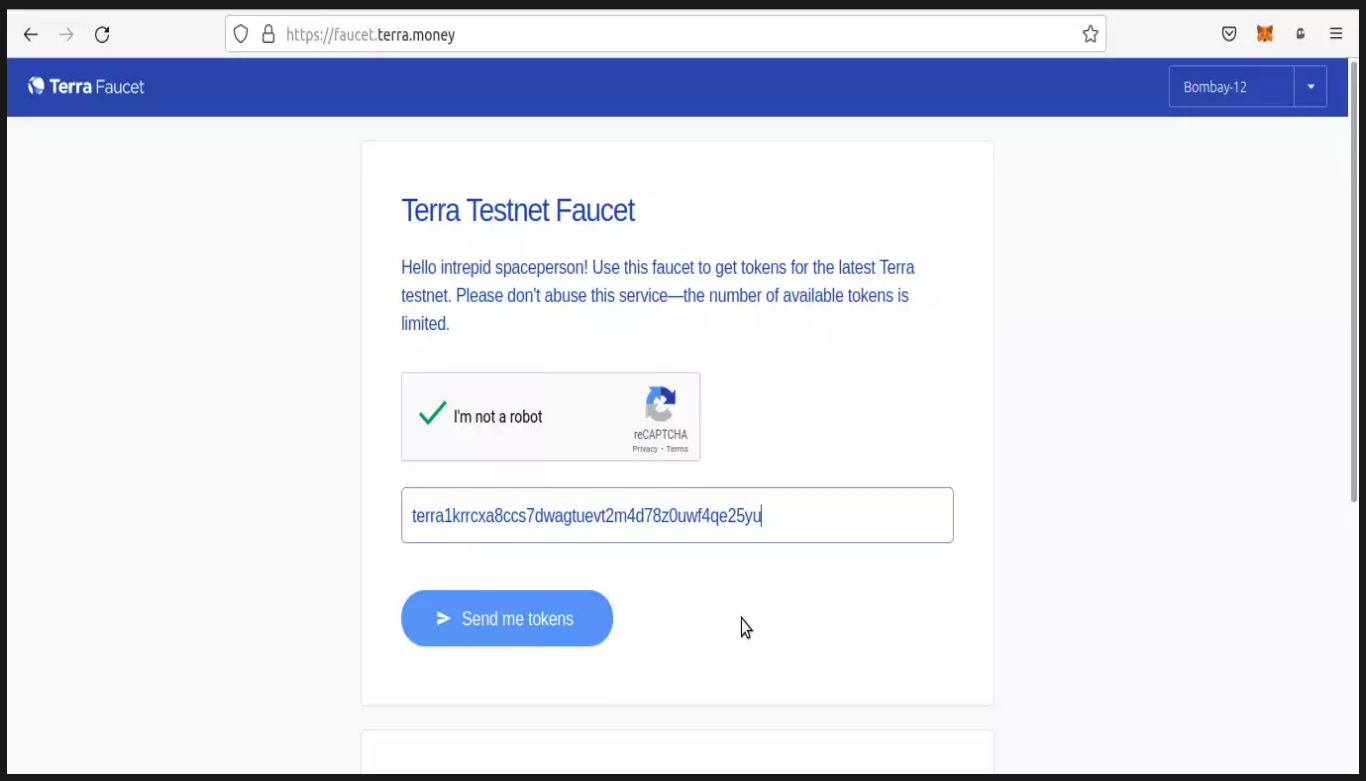
Result:

b'\xf1\x92IyzN<\x14\xf2{N\x8a=\x86\x06\x9c \xab\n /*\xb0\xb8b+z\xec5\x10\xcd,'

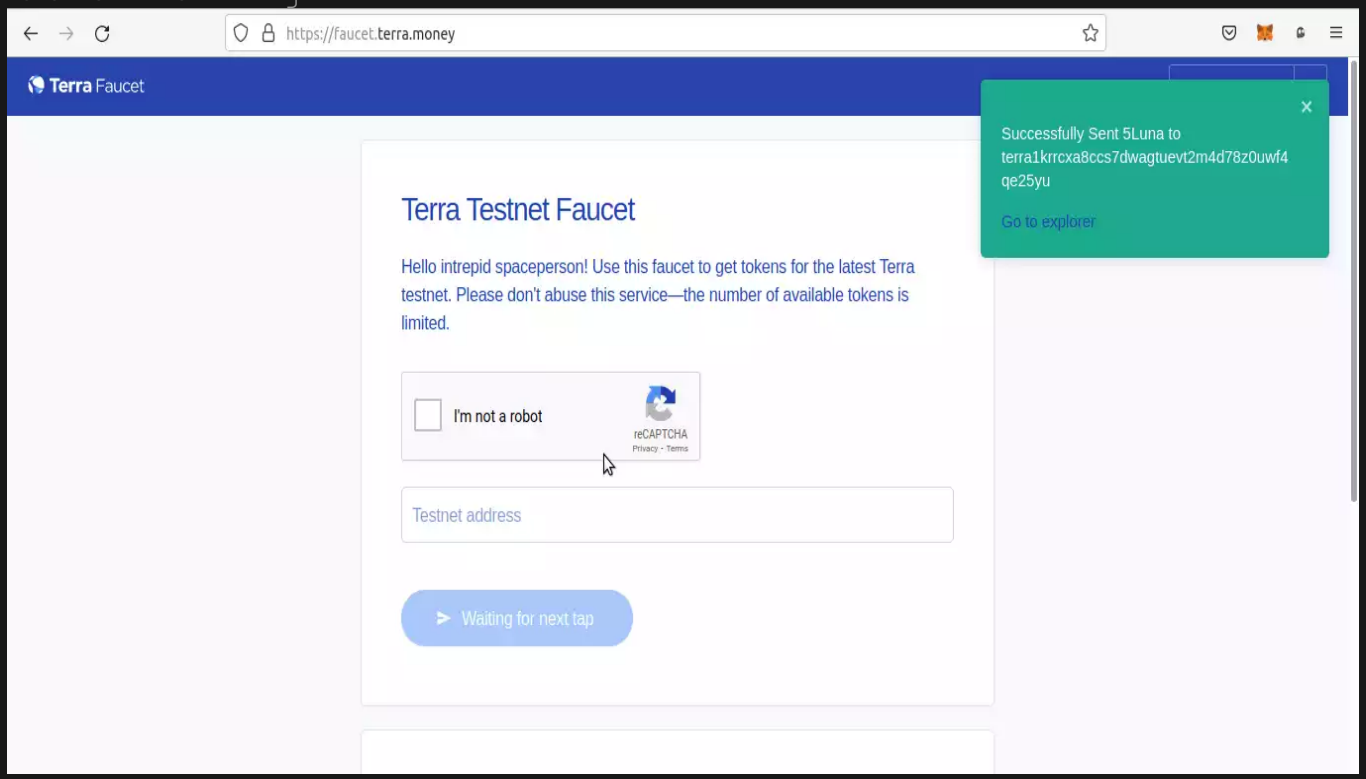
And the private key by:

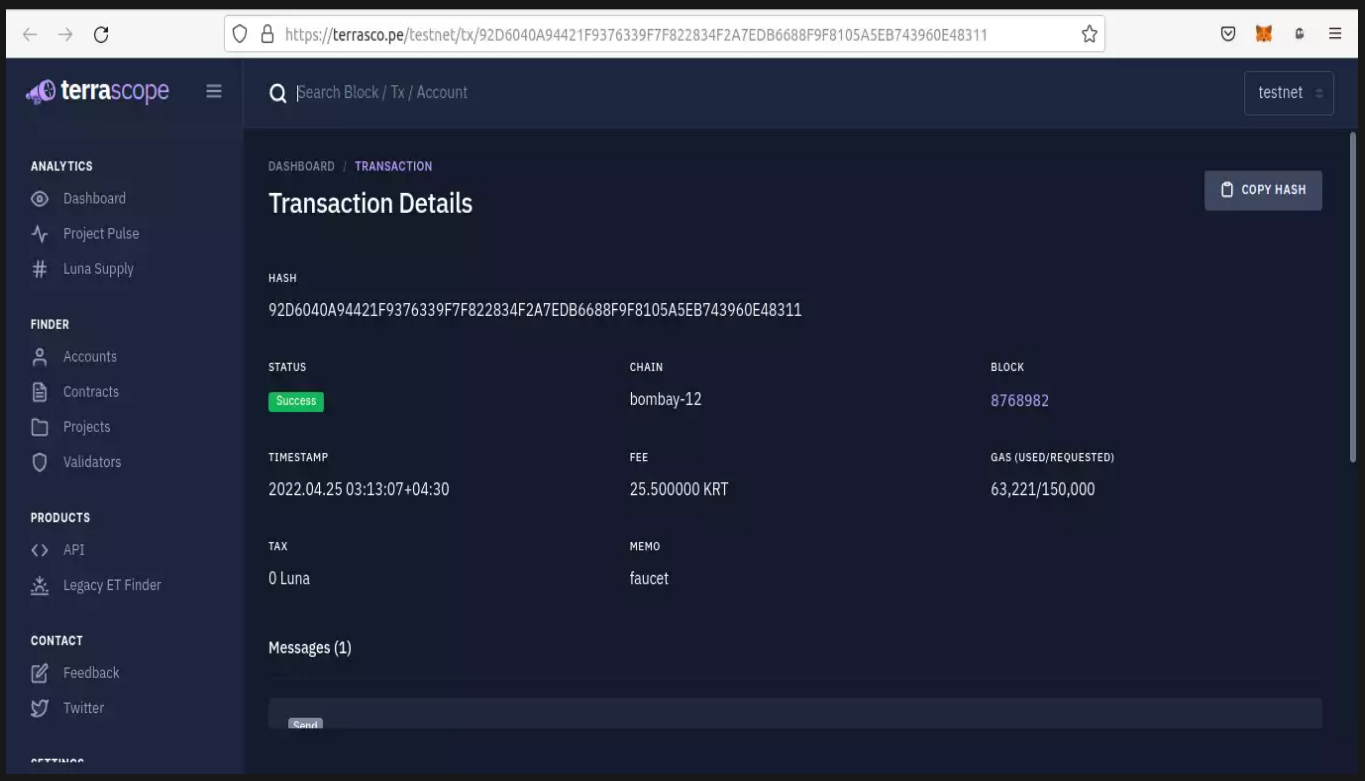
```
print(mk.private_key)
```

Besides, make sure you save the mnemonic key somewhere safe. Let's get some Luna from Terra Testnet Faucet `https://faucet.terra.money/`. We need to copy the account address we've got in the box:

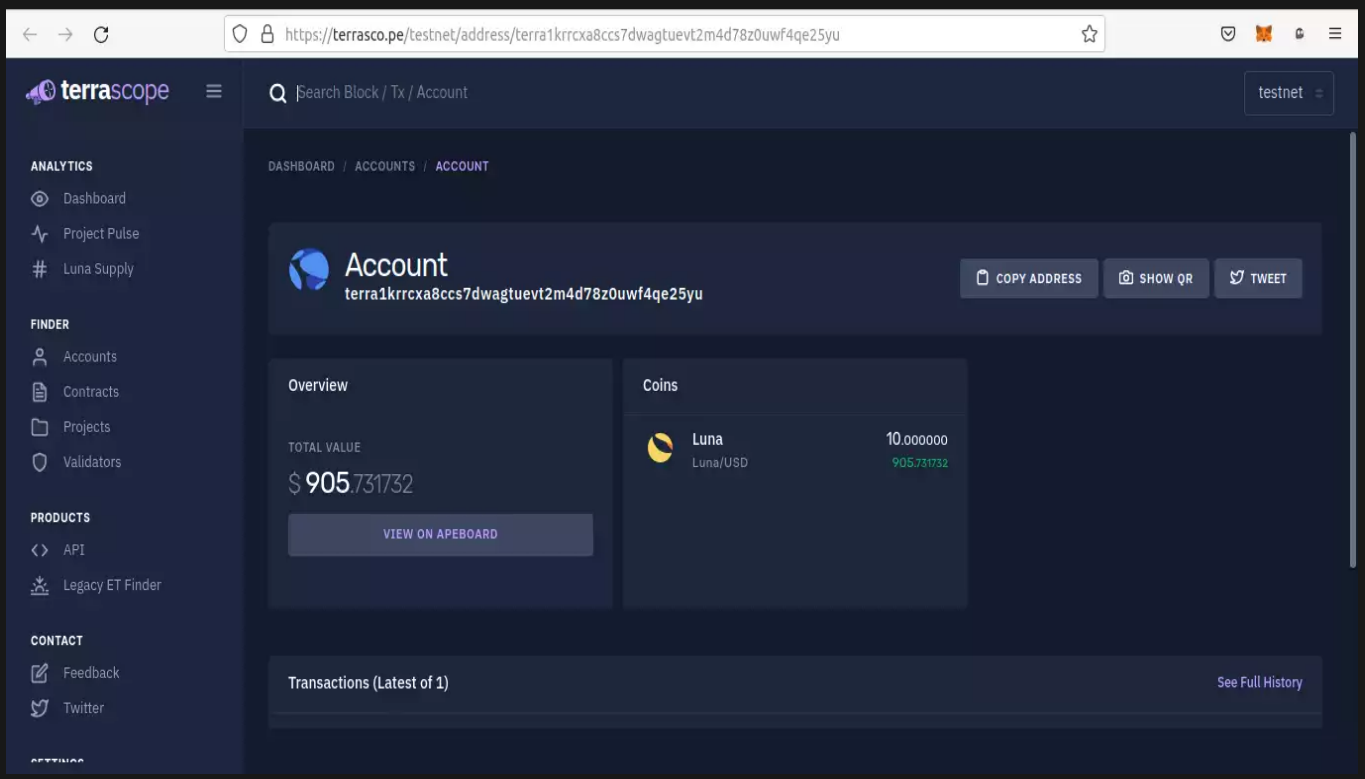


Once we get the message, we will be redirected to the Terrascope which is similar to Etherscan and we can see the test Luna that we have been given:





We can also see the details of our account if we click on the `to` address:



We can retrieve the balance of our account using the following code:

```
from terra_sdk.client.lcd import LCDClient
from terra_sdk.key.mnemonic import MnemonicKey
terra = LCDClient("https://lcd.terra.dev", " bombay-12")
mk = MnemonicKey(mnemonic=
"disorder solar ride ecology father pear conduct
     anger distance soul slow outside market twenty badge alter busy inspire
tag
     enforce chicken shaft giggle measure")
print(terra.bank.balance(mk.acc_address))
```

Notice that in the above code, we have managed to call back the account using the mnemonic key we got when we used the `mk = MnemonicKey()` script. We have also connected to the Bombay-12 Testnet. Now let's run the code: `python3 Test.py` Result: `Coins('1000000000uluna')`

Interacting with Terra Network Using SDK: Wallet and Changing

In this part, we are going to see how we can connect to the Terra network, create an account and a wallet, create and sign transactions, estimate gas fees, and swap tokens. Of course, all of these operations are going to be executed on a Testnet rather than a Mainnet in case we lose real money. In this article, we are going to cover more interactions with Terra Testnet by creating and signing transactions, transferring money, creating a wallet and an account, and sending JSON messages to Terra smart contracts.

Setting up the Network and the Chain ID

Up to now, we have learned how to connect to Terra chain Mainnet and Testnet, how to get some test Luna and how we can create test accounts. One thing to notice is how we can switch between different networks (Mainnet to Testnet). As we have mentioned before, there are 2 parameters for defining every network:

1. URL
2. Chain ID

Terra chain has had many test networks such as tequila which is outdated, Bombay (the newer one), and other custom networks created by developers. You can also create your own node using the guides on the Terra Documentation website `https://docs.terra.money/docs/full-node/run-a-full-terra-node/join-a-network.html#`.

If you are on Mainnet, you can enter as the `https://lcd.terra.dev` URL and as Columbus-5 chain id and if you are on Testnet, you can use `https://bombay-lcd.terra.dev/` as the URL and Bombay-12 as your chain id. Notice that at the current time of writing, the Terra The faucet is on Bombay test network. So if you want to interact with the test ulunas sent from Bombay Faucet, you should use the specific URL and chain id of this test network.

Creating An Account on Terra Testnet

Using the below script, we can create an account, and get a mnemonic key for it. Make sure you copy the mnemonic somewhere so that you can use it later when you want to get some test Luna from Terra Faucet or when you want to get the balance of the account or create and sign a transaction using it.

```
from terra_sdk.client.lcd import LCDClient
from terra_sdk.key.mnemonic import MnemonicKey
terra = LCDClient("https://bombay-lcd.terra.dev/", "bombay-12")
Account1 = MnemonicKey()
print(Account1.mnemonic)
```

Result: `call oil decrease loud pull indicate diet post sign cereal adapt rug reform alcohol math illegal major anchor unhappy govern win round ritual various` Now, we can use the Bombay Faucet to get some test uluna. And using the below script we can see the balance of account 1 containing uluna,

```
print(terra.bank.balance(Account1.acc_address)[0]['uluna'].amount)
```

Result: `5000000` And using the below code, we can see all the assets of the account1:

```
print(terra.bank.balance(Account1.acc_address)[0])
```

Result:

```
Coins('5000000uluna')
```

Transferring Luna from One Account to Another Using Python Script

Now, we want to create a transaction using Account1 with the mnemonic key that we have got and creating a new account called Account2.

Notice that we define each account with its specific mnemonic key and here when we want to use the created account (Account1), we use the mnemonic key to reuse it, otherwise, a new account will be created with no funds in it.

```
Account1 = MnemonicKey(mnemonic=
"call oil decrease loud pull indicate diet post
     sign cereal adapt rug reform alcohol math illegal
     major anchor unhappy govern win round
     ritual various")
```

And here comes the complete code to create a transaction and send a quarter of the balance of account 1 to account 2. We will also create a wallet for account 1 to be able to execute the transaction.

```
from terra_sdk.client.lcd import LCDClient
from terra_sdk.key.mnemonic import MnemonicKey
from terra_sdk.core.fee import Fee
from terra_sdk.core.bank import MsgSend
from terra_sdk.client.lcd.api.tx import CreateTxOptions

terra = LCDClient("https://bombay-lcd.terra.dev/", "bombay-12")
Account1 = MnemonicKey(mnemonic=
"call oil decrease loud pull indicate diet post
     sign cereal adapt rug reform alcohol math illegal
     major anchor unhappy govern win round
     ritual various")
Account2 = MnemonicKey()
wallet = terra.wallet(Account1)
print(terra.bank.balance(Account1.acc_address)[0]['uluna'].amount)
print(terra.bank.balance(Account2.acc_address)[0]['uluna'].amount)
Amount_To_Send = (terra.bank.balance(Account1.acc_address)[0])/4
print(Amount_To_Send)

tx = wallet.create_and_sign_tx(CreateTxOptions(
     msgs = [MsgSend(from_address=Account1.acc_address,
          to_address = Account2.acc_address,
          amount=Amount_To_Send )],
     memo = "Only Testing",
     gas_prices = "1uluna",
     gas_adjustment = "1.5"))

result = terra.tx.broadcast(tx)
print(result)
```

Result: 5000000 0 1250000uluna BlockTxBroadcastResult(height=8795417, txhash='10B07115283D05126EF4DCA72576BB3E4DD812B24B002614913020532 2999645', raw_log='[{"events":[{"type":"coin_received","attributes": [{"key":"receiver","value":"terra1e20tg4gat39ex4ezer724r5qs9dhnshjg065d4"}, {"key":"amount","value":"1250000uluna"}]},{"type":"coin_spent","attributes": [{"key":"spender","value":"terra16q9hkwe9t5jac20e08tz4qwflwkjxymw9qm3kr"}, {"key":"amount","value":"1250000uluna"}]},{"type":"message","attributes": [{"key":"action","value":"/cosmos.bank.v1beta1.MsgSend"}, {"key":"sender","value":"terra16q9hkwe9t5jac20e08tz4qwflwkjxymw9qm3kr"}, {"key":"module","value":"bank"}]},{"type":"transfer","attributes": [{"key":"recipient","value":"terra1e20tg4gat39ex4ezer724r5qs9dhnshjg065d4"}, {"key":"sender","value":"terra16q9hkwe9t5jac20e08tz4qwflwkjxymw9qm3kr"}, {"key":"amount","value":"1250000uluna"}]}]}]', gas_wanted=95229, gas_used=76149, logs=[TxLog(msg_index=0, log='', events=[{'type': 'coin_received', 'attributes': [{'key': 'receiver', 'value': 'terra1e20tg4gat39ex4ezer724r5qs9dhnshjg065d4'}, {'key': 'amount', 'value': '1250000uluna'}]}, {'type': 'coin_spent', 'attributes': [{'key': 'spender', 'value':

```
'terra16q9hkwe9t5jac20e08tz4qwflwkjxymw9qm3kr'}, {'key': 'amount',
'value': '1250000uluna'}]}, {'type': 'message', 'attributes': [{'key':
'action', 'value': '/cosmos.bank.v1beta1.MsgSend'}, {'key': 'sender',
'value': 'terra16q9hkwe9t5jac20e08tz4qwflwkjxymw9qm3kr'}, {'key':
'module', 'value': 'bank'}]}, {'type': 'transfer', 'attributes':
[{'key': 'recipient', 'value':
'terra1e20tg4gat39ex4ezer724r5qs9dhnshjg065d4'}, {'key': 'sender',
'value': 'terra16q9hkwe9t5jac20e08tz4qwflwkjxymw9qm3kr'}, {'key':
'amount', 'value': '1250000uluna'}]}],
events_by_type={'coin_received': {'receiver':
['terra1e20tg4gat39ex4ezer724r5qs9dhnshjg065d4'], 'amount':
['1250000uluna']}, 'coin_spent': {'spender':
['terra16q9hkwe9t5jac20e08tz4qwflwkjxymw9qm3kr'], 'amount':
['1250000uluna']}, 'message': {'action':
['/cosmos.bank.v1beta1.MsgSend'], 'sender':
['terra16q9hkwe9t5jac20e08tz4qwflwkjxymw9qm3kr'], 'module': ['bank']},
'transfer': {'recipient':
['terra1e20tg4gat39ex4ezer724r5qs9dhnshjg065d4'], 'sender':
['terra16q9hkwe9t5jac20e08tz4qwflwkjxymw9qm3kr'], 'amount':
['1250000uluna']}})], code=0, codespace='', info=None, data=None,
timestamp=None)
```

As you can see after a few seconds, the result of the transaction is printed out and you can get some of the data you need from this very long output. If we want to check our transaction details we can head over to Terra Finder `https://finder.terra.money/` and paste the `txhash` from the resulting output. Notice that you should choose the network (Mainnet, Testnet, or local) from the top-right menu. And you will be able to see the details of the transaction similar to what we saw in Etherscan for the Ethereum blockchain.

In this article we are going to get familiar with the Terra Network Using Terra Python SDK and how it works. Then, we will see how we can use Terra Python SDK to interact with Terra Testnet called Bombay-12 and Mainnet called Columbus-5. In this 3rd part of Terra Python SDK, we are going to fetch price data from Terra Network, estimate gas fees for our transactions, and swap tokens using the Python SDK for Terra.

Terra Network Using Terra Python SDK: the Last Part

In this 3rd part of Terra Python SDK, we are going to fetch price data from Terra Network, estimate the gas fee for our transactions, and swap tokens using the Python SDK for Terra.

Requesting Price Data

We can fetch the price of uluna from FCD terra node with the following script:

```
from terra_sdk.client.lcd import LCDClient
from terra_sdk.key.mnemonic import MnemonicKey
import requests
import json

terra = LCDClient("https://bombay-lcd.terra.dev/", "bombay-12")
Account1 = MnemonicKey(mnemonic="call oil decrease loud pull indicate diet po
     sign cereal adapt rug reform alcohol
```

```
    math illegal major anchor unhappy
    govern win round ritual various")

Account2 = MnemonicKey()
wallet = terra.wallet(Account1)
print(requests.get(
'https://fcd.terra.dev/v1/market/price?denom=uusd&interval=1h'
).json())
```

Result:

{'lastPrice': 90.48744281649324, 'oneDayVariation': '2.03086440151527', 'oneDayVariationRate': '0.02244360475114567212', 'prices': [{'denom': 'uusd', 'price': 88.96415361266797, 'datetime': 1650888000000}, {'denom': 'uusd', 'price': 89.0900844450184, 'datetime': 1650891600000}, {'denom': 'uusd', 'price': 90.66006870152763, 'datetime': 1650895200000}, {'denom': 'uusd', 'price': 92.52536809614227, 'datetime': 1650898800000}, {'denom': 'uusd', 'price': 94.04939366114037, 'datetime': 1650902400000}, {'denom': 'uusd', 'price': 94.3395565349883, 'datetime': 1650906000000}, {'denom': 'uusd', 'price': 94.4164196214135, 'datetime': 1650909600000}, {'denom': 'uusd', 'price': 95.56109206654655, 'datetime': 1650913200000}, {'denom': 'uusd', 'price': 95.81135596156976, 'datetime': 1650916800000}, {'denom': 'uusd', 'price': 95.28242522306937, 'datetime': 1650920400000}, {'denom': 'uusd', 'price': 95.58401419222808, 'datetime': 1650924000000}, {'denom': 'uusd', 'price': 96.91434728345023, 'datetime': 1650927600000}, {'denom': 'uusd', 'price': 96.80871579823581, 'datetime': 1650931200000}, {'denom': 'uusd', 'price': 96.17697287838071, 'datetime': 1650934800000}, {'denom': 'uusd', 'price': 96.03501028344412, 'datetime': 1650938400000}, {'denom': 'uusd', 'price': 96.22807016895509, 'datetime': 1650942000000}, {'denom': 'uusd', 'price': 96.06570662357241, 'datetime': 1650945600000}, {'denom': 'uusd', 'price': 95.70176476651612, 'datetime': 1650949200000}, {'denom': 'uusd', 'price': 95.60364542853246, 'datetime': 1650952800000}, {'denom': 'uusd', 'price': 96.30584856654487, 'datetime': 1650956400000}, {'denom': 'uusd', 'price': 96.28680905438578, 'datetime': 1650960000000}, {'denom': 'uusd', 'price': 96.0928405310103, 'datetime': 1650963600000}, {'denom': 'uusd', 'price': 96.37104756437165, 'datetime': 1650967200000}, {'denom': 'uusd', 'price': 96.34377771748161, 'datetime': 1650970800000}, {'denom': 'uusd', 'price': 95.68353760426069, 'datetime': 1650974400000}, {'denom': 'uusd', 'price': 95.93167829936388, 'datetime': 1650978000000}, {'denom': 'uusd', 'price': 93.96949533359299, 'datetime': 1650981600000}, {'denom': 'uusd', 'price': 91.55576611528446, 'datetime': 1650985200000}, {'denom': 'uusd', 'price': 89.93169609113293, 'datetime': 1650988800000}, {'denom': 'uusd', 'price': 89.23415738171494, 'datetime': 1650992400000}, {'denom': 'uusd', 'price': 89.06808181910975, 'datetime': 1650996000000}, {'denom': 'uusd', 'price': 88.48694667915775,

'datetime': 1650999600000}, {'denom': 'uusd', 'price': 88.11627958903455, 'datetime': 1651003200000}, {'denom': 'uusd', 'price': 89.09030726339398, 'datetime': 1651006800000}, {'denom': 'uusd', 'price': 89.06232865592419, 'datetime': 1651010400000}, {'denom': 'uusd', 'price': 88.54428184656634, 'datetime': 1651014000000}, {'denom': 'uusd', 'price': 88.778354665981, 'datetime': 1651017600000}, {'denom': 'uusd', 'price': 89.00243656718825, 'datetime': 1651021200000}, {'denom': 'uusd', 'price': 88.7637772736078, 'datetime': 1651024800000}, {'denom': 'uusd', 'price': 88.43322633009933, 'datetime': 1651028400000}, {'denom': 'uusd', 'price': 88.03606836235143, 'datetime': 1651032000000}, {'denom': 'uusd', 'price': 88.30151817608929, 'datetime': 1651035600000}, {'denom': 'uusd', 'price': 88.60340905121288, 'datetime': 1651039200000}, {'denom': 'uusd', 'price': 89.83111008867395, 'datetime': 1651042800000}, {'denom': 'uusd', 'price': 89.64450032950381, 'datetime': 1651046400000}, {'denom': 'uusd', 'price': 89.34933700739906, 'datetime': 1651050000000}, {'denom': 'uusd', 'price': 89.27527287128059, 'datetime': 1651053600000}, {'denom': 'uusd', 'price': 88.73870274154594, 'datetime': 1651057200000}, {'denom': 'uusd', 'price': 88.83721399177365, 'datetime': 1651060800000}, {'denom': 'uusd', 'price': 89.2133569858777, 'datetime': 1651064400000}]}

Gas Fee Estimation on Terra Blockchain

Gas fee estimation is an important step when you want to execute a transaction on a blockchain. Using the following code in which we use `https://fcd.terra.dev/v1/txs/gas_prices`, we can get the gas fee in different tokens. So, we can estimate the gas fee for our transaction in the future:

```
from terra_sdk.client.lcd import LCDClient
from terra_sdk.key.mnemonic import MnemonicKey
import requests
import json
terra = LCDClient("https://bombay-lcd.terra.dev/", "bombay-12")
Account1 = MnemonicKey(mnemonic="call oil decrease loud pull indicate diet po
     sign cereal adapt rug reform alcohol math illegal major anchor unhappy g
     round ritual various")
Account2 = MnemonicKey()
wallet = terra.wallet(Account1)
fees = requests.get("https://fcd.terra.dev/v1/txs/gas_prices").json()
print(fees)
```

Result: {'uluna': '0.01133', 'usdr': '0.104938', 'uusd': '0.15', 'ukrw': '170.0', 'umnt': '428.571', 'ueur': '0.125', 'ucny': '0.98', 'ujpy': '16.37', 'ugbp': '0.11', 'uinr': '10.88', 'ucad': '0.19', 'uchf': '0.14', 'uaud': '0.19', 'usgd': '0.2', 'uthb': '4.62', 'usek': '1.25', 'unok': '1.25', 'udkk': '0.9', 'uidr': '2180.0', 'uphp':

`'7.6', 'uhkd': '1.17', 'umyr': '0.6', 'utwd': '4.0'}` And you can see the .json output of the fee in different tokens. Now, it is time to estimate the gas fee using `fee = str(float(fees["uusd"])) + "uusd"`:

```
from terra_sdk.client.lcd import LCDClient
from terra_sdk.key.mnemonic import MnemonicKey
import requests
import json
from terra_sdk.core.bank import MsgSend
terra = LCDClient("https://bombay-lcd.terra.dev/", "bombay-12")
Account1 = MnemonicKey(mnemonic=
"call oil decrease loud pull indicate diet post
     sign cereal adapt rug reform alcohol math illegal major anchor unhappy g
     round ritual various")
Account2 = MnemonicKey()
wallet = terra.wallet(Account1)
fees = requests.get("https://fcd.terra.dev/v1/txs/gas_prices").json()
Amount_To_Send = (terra.bank.balance(Account1.acc_address)[0])/10
print(Amount_To_Send)
msg = MsgSend(from_address=Account1.acc_address,
     to_address=Account2.acc_address,
     amount=Amount_To_Send )

fee = str(float(fees["uusd"])) + "uusd"
print(fee)
```

Result: `67360uluna 0.15uusd`

Swapping Tokens on Terra Blockchain

One of the most useful features of Terra blockchain is the ability to swap between different tokens of its chain. For example, here we want to swap our Bombay testnet uluna with test UST. The below script shows how we simply do that. To make sure that the swapping transaction has been successfully completed, we check the balance before and after the transaction.

```
from terra_sdk.client.lcd import LCDClient
from terra_sdk.key.mnemonic import MnemonicKey
from terra_sdk.core.market import MsgSwap
from terra_sdk.client.lcd.api.tx import CreateTxOptions
terra = LCDClient("https://bombay-lcd.terra.dev/", "bombay-12")
Account1 = MnemonicKey(mnemonic=
"call oil decrease loud pull indicate diet post
     sign cereal adapt rug reform alcohol math illegal major anchor unhappy g
     round ritual various")
msg = MsgSwap(Account1.acc_address, "100uluna", "uusd")
wallet = terra.wallet(Account1)
print(terra.bank.balance(Account1.acc_address)[0])
```

```
tx = wallet.create_and_sign_tx(CreateTxOptions(msgs = [msg],
     memo = "Swapping luna for UST",
     gas_prices = "1uluna",
     gas_adjustment = "1.3",
     fee_denoms = ["uluna", "uusd"]
     ))
result = terra.tx.broadcast(tx)
print(result)
print(terra.bank.balance(Account1.acc_address)[0])
```

Result:

```
421527uluna,17604uusd BlockTxBroadcastResult(height=8805631,
txhash='D87F2D6ACD0EAA919EA6AAE66DC173DCF162A58668560C9284EA4
52D42279125', raw_log='[{"events":[{"type":"burn","attributes":
[{"key":"burner","value":"terra1untf85jwv3kt0puyyc39myxjvplagr3wstgs5s"},
{"key":"amount","value":"100uluna"}]},{"type":"coin_received","attributes":
[{"key":"receiver","value":"terra1untf85jwv3kt0puyyc39myxjvplagr3wstgs5s"},
{"key":"amount","value":"100uluna"},
{"key":"receiver","value":"terra1untf85jwv3kt0puyyc39myxjvplagr3wstgs5s"},
{"key":"amount","value":"8864uusd"},
{"key":"receiver","value":"terra16q9hkwe9t5jac20e08tz4qwflwkjxymw9qm3kr"},
{"key":"amount","value":"8820uusd"},
{"key":"receiver","value":"terra1jgp27m8fykex4e4jtt0l7ze8q528ux2lh4zh0f"},
{"key":"amount","value":"44uusd"}]},{"type":"coin_spent","attributes":
[{"key":"spender","value":"terra16q9hkwe9t5jac20e08tz4qwflwkjxymw9qm3kr"},
{"key":"amount","value":"100uluna"},
{"key":"spender","value":"terra1untf85jwv3kt0puyyc39myxjvplagr3wstgs5s"},
{"key":"amount","value":"100uluna"},
{"key":"spender","value":"terra1untf85jwv3kt0puyyc39myxjvplagr3wstgs5s"},
{"key":"amount","value":"8820uusd"},
{"key":"spender","value":"terra1untf85jwv3kt0puyyc39myxjvplagr3wstgs5s"},
{"key":"amount","value":"44uusd"}]},{"type":"coinbase","attributes":
[{"key":"minter","value":"terra1untf85jwv3kt0puyyc39myxjvplagr3wstgs5s"},
{"key":"amount","value":"8864uusd"}]},{"type":"message","attributes":
[{"key":"action","value":"/terra.market.v1beta1.MsgSwap"},
{"key":"sender","value":"terra16q9hkwe9t5jac20e08tz4qwflwkjxymw9qm3kr"},
{"key":"sender","value":"terra1untf85jwv3kt0puyyc39myxjvplagr3wstgs5s"},
{"key":"sender","value":"terra1untf85jwv3kt0puyyc39myxjvplagr3wstgs5s"},
{"key":"module","value":"market"}]},{"type":"swap","attributes":
[{"key":"offer","value":"100uluna"},
{"key":"trader","value":"terra16q9hkwe9t5jac20e08tz4qwflwkjxymw9qm3kr"},
{"key":"recipient","value":"terra16q9hkwe9t5jac20e08tz4qwflwkjxymw9qm3kr"},
{"key":"swap_coin","value":"8820uusd"},{"key":"swap_fee","value":"44uusd"}]},
{"type":"transfer","attributes":
[{"key":"recipient","value":"terra1untf85jwv3kt0puyyc39myxjvplagr3wstgs5s"},
{"key":"sender","value":"terra16q9hkwe9t5jac20e08tz4qwflwkjxymw9qm3kr"},
{"key":"amount","value":"100uluna"},
{"key":"recipient","value":"terra16q9hkwe9t5jac20e08tz4qwflwkjxymw9qm3kr"},
{"key":"sender","value":"terra1untf85jwv3kt0puyyc39myxjvplagr3wstgs5s"},
{"key":"amount","value":"8820uusd"},
```

{"key":"recipient","value":"terra1jgp27m8fykex4e4jtt0l7ze8q528ux2lh4zh0f"},
{"key":"sender","value":"terra1untf85jwv3kt0puyyc39myxjvplagr3wstgs5s"},
{"key":"amount","value":"44uusd"}]}]}]', gas_wanted=126031,
gas_used=108830, logs=[TxLog(msg_index=0, log='', events=[{'type':
'burn', 'attributes': [{'key': 'burner', 'value':
'terra1untf85jwv3kt0puyyc39myxjvplagr3wstgs5s'}, {'key': 'amount',
'value': '100uluna'}]}, {'type': 'coin_received', 'attributes':
[{'key': 'receiver', 'value':
'terra1untf85jwv3kt0puyyc39myxjvplagr3wstgs5s'}, {'key': 'amount',
'value': '100uluna'}, {'key': 'receiver', 'value':
'terra1untf85jwv3kt0puyyc39myxjvplagr3wstgs5s'}, {'key': 'amount',
'value': '8864uusd'}, {'key': 'receiver', 'value':
'terra16q9hkwe9t5jac20e08tz4qwflwkjxymw9qm3kr'}, {'key': 'amount',
'value': '8820uusd'}, {'key': 'receiver', 'value':
'terra1jgp27m8fykex4e4jtt0l7ze8q528ux2lh4zh0f'}, {'key': 'amount',
'value': '44uusd'}]}, {'type': 'coin_spent', 'attributes': [{'key':
'spender', 'value': 'terra16q9hkwe9t5jac20e08tz4qwflwkjxymw9qm3kr'},
{'key': 'amount', 'value': '100uluna'}, {'key': 'spender', 'value':
'terra1untf85jwv3kt0puyyc39myxjvplagr3wstgs5s'}, {'key': 'amount',
'value': '100uluna'}, {'key': 'spender', 'value':
'terra1untf85jwv3kt0puyyc39myxjvplagr3wstgs5s'}, {'key': 'amount',
'value': '8820uusd'}, {'key': 'spender', 'value':
'terra1untf85jwv3kt0puyyc39myxjvplagr3wstgs5s'}, {'key': 'amount',
'value': '44uusd'}]}, {'type': 'coinbase', 'attributes': [{'key':
'minter', 'value': 'terra1untf85jwv3kt0puyyc39myxjvplagr3wstgs5s'},
{'key': 'amount', 'value': '8864uusd'}]}, {'type': 'message',
'attributes': [{'key': 'action', 'value':
'/terra.market.v1beta1.MsgSwap'}, {'key': 'sender', 'value':
'terra16q9hkwe9t5jac20e08tz4qwflwkjxymw9qm3kr'}, {'key': 'sender',
'value': 'terra1untf85jwv3kt0puyyc39myxjvplagr3wstgs5s'}, {'key':
'sender', 'value': 'terra1untf85jwv3kt0puyyc39myxjvplagr3wstgs5s'},
{'key': 'module', 'value': 'market'}]}, {'type': 'swap', 'attributes':
[{'key': 'offer', 'value': '100uluna'}, {'key': 'trader', 'value':
'terra16q9hkwe9t5jac20e08tz4qwflwkjxymw9qm3kr'}, {'key': 'recipient',
'value': 'terra16q9hkwe9t5jac20e08tz4qwflwkjxymw9qm3kr'}, {'key':
'swap_coin', 'value': '8820uusd'}, {'key': 'swap_fee', 'value':
'44uusd'}]}, {'type': 'transfer', 'attributes': [{'key': 'recipient',
'value': 'terra1untf85jwv3kt0puyyc39myxjvplagr3wstgs5s'}, {'key':
'sender', 'value': 'terra16q9hkwe9t5jac20e08tz4qwflwkjxymw9qm3kr'},
{'key': 'amount', 'value': '100uluna'}, {'key': 'recipient', 'value':
'terra16q9hkwe9t5jac20e08tz4qwflwkjxymw9qm3kr'}, {'key': 'sender',
'value': 'terra1untf85jwv3kt0puyyc39myxjvplagr3wstgs5s'}, {'key':
'amount', 'value': '8820uusd'}, {'key': 'recipient', 'value':
'terra1jgp27m8fykex4e4jtt0l7ze8q528ux2lh4zh0f'}, {'key': 'sender',
'value': 'terra1untf85jwv3kt0puyyc39myxjvplagr3wstgs5s'}, {'key':
'amount', 'value': '44uusd'}]}], events_by_type={'burn': {'burner':
['terra1untf85jwv3kt0puyyc39myxjvplagr3wstgs5s'], 'amount':
['100uluna']}, 'coin_received': {'receiver':
['terra1untf85jwv3kt0puyyc39myxjvplagr3wstgs5s',
'terra1untf85jwv3kt0puyyc39myxjvplagr3wstgs5s',
'terra16q9hkwe9t5jac20e08tz4qwflwkjxymw9qm3kr',

'terra1jgp27m8fykex4e4jtt0l7ze8q528ux2lh4zh0f'], 'amount': ['100uluna', '8864uusd', '8820uusd', '44uusd']}, 'coin_spent': {'spender': ['terra16q9hkwe9t5jac20e08tz4qwflwkjxymw9qm3kr', 'terra1untf85jwv3kt0puyyc39myxjvplagr3wstgs5s', 'terra1untf85jwv3kt0puyyc39myxjvplagr3wstgs5s', 'terra1untf85jwv3kt0puyyc39myxjvplagr3wstgs5s'], 'amount': ['100uluna', '100uluna', '8820uusd', '44uusd']}, 'coinbase': {'minter': ['terra1untf85jwv3kt0puyyc39myxjvplagr3wstgs5s'], 'amount': ['8864uusd']}, 'message': {'action': ['/terra.market.v1beta1.MsgSwap','sender': ['terra16q9hkwe9t5jac20e08tz4qwflwkjxymw9qm3kr', 'terra1untf85jwv3kt0puyyc39myxjvplagr3wstgs5s', 'terra1untf85jwv3kt0puyyc39myxjvplagr3wstgs5s'], 'module': ['market']}, 'swap': {'offer': ['100uluna'], 'trader': ['terra16q9hkwe9t5jac20e08tz4qwflwkjxymw9qm3kr'], 'recipient': ['terra16q9hkwe9t5jac20e08tz4qwflwkjxymw9qm3kr'], 'swap_coin': ['8820uusd'], 'swap_fee': ['44uusd']}, 'transfer': {'recipient': ['terra1untf85jwv3kt0puyyc39myxjvplagr3wstgs5s', 'terra16q9hkwe9t5jac20e08tz4qwflwkjxymw9qm3kr', 'terra1jgp27m8fykex4e4jtt0l7ze8q528ux2lh4zh0f'], 'sender': ['terra16q9hkwe9t5jac20e08tz4qwflwkjxymw9qm3kr', 'terra1untf85jwv3kt0puyyc39myxjvplagr3wstgs5s', 'terra1untf85jwv3kt0puyyc39myxjvplagr3wstgs5s'], 'amount': ['100uluna', '8820uusd', '44uusd']}})], code=0, codespace='', info=None, data=None, timestamp=None) 295396uluna,26424uusd And as you can see we have got some uusd in exchange for ulunas.

Learned to Work with Terra Network Using Terra Python SDK

Firstly, we have learned how to connect to Terra Mainnet and Testnet, create an account, and get the mnemonic key, private key, and the balance of the account.

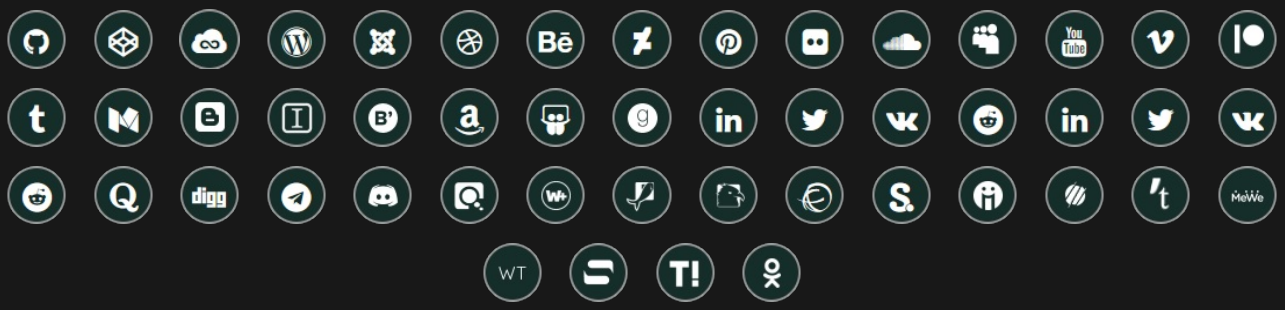
Secondly, we have learned how to connect to the Terra Tesnet blockchain called Bombay-12, create an account, create a wallet, transfer money from one account to another and create and sign a transaction using python Terra SDK.

And finally, we have learned how to work with Terra Testnet and we have connected to the Bombay-12 test network in addition to creating a wallet, and an account, creating and signing a transaction, transferring money, swapping tokens, and estimating gas fees for a transaction.

Join Arashtad Community

Follow Arashtad on Social Media

We provide variety of content, products, services, tools, tutorials, etc. Each social profile according to its features and purpose can cover only one or few parts of our updates. We can not upload our videos on SoundCloud or provide our eBooks on Youtube. So, for not missing any high quality original content that we provide on various social networks, make sure you follow us on as many social networks as you're active in. You can find out Arashtad's profiles on different social media services.



Get Even Closer!

Did you know that only one universal Arashtad account makes you able to log into all Arashtad network at once? Creating an Arashtad account is free. Why not to try it? Also, we have regular updates on our newsletter and feed entries. Use all these benefitial free features to get more involved with the community and enjoy the many products, services, tools, tutorials, etc. that we provide frequently.

SIGN UP | NEWSLETTER | RSS FEED