WALLET WITH PYTHON SDK

DESCRIPTION:

Wallets are a way for users to interact with the blockchain and view their account information. Due to the transparent-nature of blockchains, anyone can create a wallet. Today there are multiple wallets that exist for Algorand, with Pera and MyAlgo being the most popular. Other wallet solutions, such as DeFly or Vendible, add additional features on top of standard wallets that make them unique.To start building on Algorand, you need to prepare your development environment. A development environment requires getting access to an Algorand node. You need access to a node to submit new transactions, read blockchain data, and manage wallets.

BASIC STRUCTURE:

* To setup the environment
* Software devolepment kit
* Section index

YOUR FIRST TRANSACTION:

This section is a quick start guide for interacting with the Algorand network using Python. This guide will help to install **sandbox**, which provides a node for testing and development. This guide will also help to install the Python SDK, create an account and submit your first transaction on Algorand.

## **Install Sandbox**

**Prerequisites**

* Docker Compose ([install guide](https://docs.docker.com/compose/install/))
* Git ([install guide](https://git-scm.com/book/en/v2/Getting-Started-Installing-Git))
* Algorand provides a docker instance for setting up a node, which can be used to get started developing quickly. To install and use this instance, follow these instructions.​
* git clone https://github.com/algorand/sandbox.git
* cd sandbox
* ./sandbox up testnet
* [Watch Video](https://youtu.be/ku2hFalMWmA?t=23)  
  [More Information](https://developer.algorand.org/articles/introducing-sandbox-20/)
* This will install a Sandbox node connected to the Algorand TestNet. To read more about Algorand networks see [Algorand Networks](https://developer.algorand.org/docs/get-details/algorand-networks/" \t "_blank).

## **Install SDK**

Algorand provides an SDK for Python which is available as a pip package. To install the Python SDK, open a terminal and run the following command:​

pip3 install py-algorand-sdk

[Watch Video](https://youtu.be/ku2hFalMWmA?t=128) ​

Alternatively, choose and download a [distribution file](https://pypi.org/project/py-algorand-sdk/#files), and run

pip3 install [file name].

The [GitHub repository](https://github.com/algorand/py-algorand-sdk) contains additional documentation and examples.

See the Python SDK [reference documentation](https://py-algorand-sdk.readthedocs.io/en/latest/) for more information on methods.  
​The SDK is installed and can now interact with the Sandbox created earlier.​

## **Create an Account on Algorand**

In order to interact with the Algorand blockchain, you must have a funded account. To quickly create a test account use the following code.​

## **Fund the Account**

Before sending transactions to the Algorand network, the account must be funded to cover the minimal transaction fees that exist on Algorand. To fund the account use the [Algorand faucet](https://dispenser.testnet.aws.algodev.network/" \t "_blank)

## **Connect Your Client**

Client must be instantiated prior to making calls to the API endpoints. You must provide values for <algod-address> and <algod-token>. The CLI tools implement the client natively. By default, the algod\_token for each [sandbox](https://github.com/algorand/sandbox) is set to its aaa... value and the algod\_address corresponds to http://localhost:4001.

## **Check Your Balance**

Before moving on to the next step, make sure your account has been funded by the faucet.

account\_info = algod\_client.account\_info(my\_address)

print("Account balance: {} microAlgos".format(account\_info.get('amount')) + "\n")

## **Build First Transaction**

Transactions are used to interact with the Algorand network. To create a payment transaction use the following code.​

## **Sign First Transaction**

Before the transaction is considered valid, it must be signed by a private key. Use the following code to sign the transaction.​

# sign transaction

signed\_txn = unsigned\_txn.sign(private\_key)

## **Submit the Transaction**

The signed transaction can now be submitted to the network. wait\_for\_confirmation SDK Method is called after the transaction is submitted to wait until the transaction is broadcast to the Algorand blockchain and is confirmed.​

## **View the Transaction**

To view the transaction, open [AlgoExplorer](https://testnet.algoexplorer.io/" \t "_blank) or [Goal Seeker](https://goalseeker.purestake.io/algorand/testnet) and paste the transaction ID into the search bar.

import json

import base64

from algosdk import account, mnemonic, constants

from algosdk.v2client import algod

from algosdk.future import transaction

def generate\_algorand\_keypair():

private\_key, address = account.generate\_account()

print("My address: {}".format(address))

print("My private key: {}".format(private\_key))

print("My passphrase: {}".format(mnemonic.from\_private\_key(private\_key)))

# Write down the address, private key, and the passphrase for later usage

generate\_algorand\_keypair()

def first\_transaction\_example(private\_key, my\_address):

algod\_address = "http://localhost:4001"

algod\_token = "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"

algod\_client = algod.AlgodClient(algod\_token, algod\_address)

print("My address: {}".format(my\_address))

account\_info = algod\_client.account\_info(my\_address)

print("Account balance: {} microAlgos".format(account\_info.get('amount')))

# build transaction

params = algod\_client.suggested\_params()

# comment out the next two (2) lines to use suggested fees

params.flat\_fee = constants.MIN\_TXN\_FEE

params.fee = 1000

receiver = "HZ57J3K46JIJXILONBBZOHX6BKPXEM2VVXNRFSUED6DKFD5ZD24PMJ3MVA"

amount = 100000

note = "Hello World".encode()

unsigned\_txn = transaction.PaymentTxn(my\_address, params, receiver, amount, None, note)

# sign transaction

signed\_txn = unsigned\_txn.sign(private\_key)

# submit transaction

txid = algod\_client.send\_transaction(signed\_txn)

print("Signed transaction with txID: {}".format(txid))

# wait for confirmation

try:

confirmed\_txn = transaction.wait\_for\_confirmation(algod\_client, txid, 4)

except Exception as err:

print(err)

return

print("Transaction information: {}".format(

json.dumps(confirmed\_txn, indent=4)))

print("Decoded note: {}".format(base64.b64decode(

confirmed\_txn["txn"]["txn"]["note"]).decode()))

print("Starting Account balance: {} microAlgos".format(account\_info.get('amount')) )

print("Amount transfered: {} microAlgos".format(amount) )

print("Fee: {} microAlgos".format(params.fee) )

account\_info = algod\_client.account\_info(my\_address)

print("Final Account balance: {} microAlgos".format(account\_info.get('amount')) + "\n")

# replace private\_key and my\_address with your private key and your address.

first\_transaction\_example(private\_key, my\_address)

