```
#!/usr/bin/env python
            = '0.0.3'
import gnupg
import os
import re
import string
import subprocess
from datetime import datetime
from decimal import Decimal
from pathlib import Path

userhome = str(Path.home()).replace('BLOCKTREE', 'home')

if 'GULD_HOME' in os.environ:
GULD_HOME = os.environ['GULD_HOME']
else:
GULD_HOME = userhome

gpg = gnupg.GPG("/usr/bin/gpg2", homedir=os.path.join(userhome, ".gnupg"))
TRUSTLEVELS = {
'e': "EXPIRED",
'q': "REVOKED?",
'-': 0, # "TRUST_UNDEFINED",
'n': 1, # "TRUST_NEVER",
'm': 2, # "TRUST_MARGINAL",
'f': 3, # 'TRUST_FULLY',
'u': 4 # "TRUST_ULTIMATE"
}

def mkdirp(path):
try:
os.makedirs(path)
except OSError as exc:
if exc.errno == os.errno.EEXIST and os.path.isdir(path):
pass
else:
raise

def get_price(commodity):
search = "P"
pline = ""
with open(os.path.join(GULD_HOME, 'ledger/prices/%s.db' % commodity.lower()), 'r') as pf:
```

pline = pf.read()

```
if          and
                                                                  !   ! "   #$          % !
```

def get_guld_sub_bals(username):
cmd = "grep -rl {0} {1} | grep \"\.d[bat]*$\" | while read line ; do echo include $line ; done | ledger -f - bal ^guld:Equity:{0}$ ^guld:Income:register:individual:{0}$".format(username, os.path.join(GULD_HOME, "ledger", "GULD"))
ledgerBals = subprocess.check_output(cmd, shell=True)
return ledgerBals.decode("utf-8")

def get_guld_overview():
cmd = ""
cmd = "find {0} -name '*.dat' | while read line ; do echo include $line ; done | ledger -f - --depth 2 bal ^guld:Equity ^guld:Liabilities".format(os.path.join(GULD_HOME, "ledger", "GULD"))
ledgerBals = subprocess.check_output(cmd, shell=True)
return ledgerBals.decode("utf-8")

def get_assets_liabs(username, in_commodity=None):
cmd = ""
if in_commodity is not None:
cmd = "printf \"$(find {2} -name ' $\" | while read line ; do echo include $line ; done | ledger -f - bal ^{0}:Assets ^{0}:Liabilities -X {3}".format(username, os.path.join(GULD_HOME, "ledger", "GULD"), os.path.join(GULD_HOME, "ledger", "prices"), in_commodity)
else:
cmd = "grep -rl {0} {1} | grep \"\.d[bat]*$\" | while read line ; do echo include $line ; done | ledger -f - bal ^{0}:Assets ^{0}:Liabilities".format(username, os.path.join(GULD_HOME, "ledger", "GULD"))
ledgerBals = subprocess.check_output(cmd, shell=True)
return ledgerBals.decode("utf-8")

def get_balance(username, in_commodity=None):
cmd = ""
if in_commodity is not None:
cmd = "printf \"$(find {2} -name ' $\" | while read line ; do echo include $line ; done | ledger -f - bal [^a-zA-Z0-9-]{0}[^a-zA-Z0-9-] -X {3}".format(username, os.path.join(GULD_HOME, "ledger", "GULD"), os.path.join(GULD_HOME, "ledger", "prices"), in_commodity)

```
else:
cmd = "grep -rl {0} {1} | grep \"\.d[bat]*$\" | while read line ; do echo include $line ; done | ledger
-f - bal [^a-zA-Z0-9-]{0}[^a-zA-Z0-9-]".format(username, os.path.join(GULD_HOME, "ledger",
"GULD"))
ledgerBals = subprocess.check_output(cmd, shell=True)
return ledgerBals.decode("utf-8")

def is_name_taken(username):
return os.path.isdir(os.path.join(GULD_HOME, 'ledger', 'GULD', username.lower()))

def import_pgp_key(username, pubkey):
import_result = gpg.import_keys(pubkey)
mkdirp(os.path.join(GULD_HOME, 'keys/pgp', username))
if len(import_result.fingerprints) > 0:
fname = os.path.join(GULD_HOME, 'keys/pgp', username, "%s.asc" %
import_result.fingerprints[0])
with open(fname, 'w') as f:
f.write(pubkey)
return import_result.fingerprints[0]
else:
return

def get_name_by_pgp_fpr(fpr):
base = os.path.join(GULD_HOME, 'keys', 'pgp')
imp = subprocess.check_output(['find', base, '-name', '%s.asc' % fpr])
if (imp):
fullpath = imp.decode('utf-8').strip()
relpath = fullpath.replace(base, '')
return relpath.strip(os.path.sep).split(os.path.sep)[0]

def get_pgp_trust(fpr):
keys = gpg.list_keys()
for key in keys:
if key['fingerprint'] == fpr:
return TRUSTLEVELS[key['ownertrust']]
```

#$ #"# #

% & & &$ % % '# #"
"#(#"#) &#) *

%+ & #% (, #

+ %+ + & "#

%
+ +
'-*

```
def get_time_date_stamp():
now = datetime.utcnow()
dt = now.strftime('%Y/%m/%d')
tstamp = int(now.timestamp())
return dt, tstamp

def gen_register(name, ntype='individual', qty=1, dt=None, tstamp=None):
if dt is None or tstamp is None:
dt, tstamp = get_time_date_stamp()
if ntype == 'individual':
amount = 1
elif ntype == 'device':
amount = 1
elif ntype == 'group':
amount = qty
```

```
else:
return
return ('{1} * register {3}\n'
' ; timestamp: {2}\n'
' {0}:Assets -{4} GULD\n'
' {0}:Expenses:guld:register {4} GULD\n'
' guld:Liabilities {4} GULD\n'
' guld:Income:register:{3}:{0} -{4} GULD\n\n'.format(name, dt, tstamp, ntype, amount))

def gen_transfer(sender, receiver, amount, commodity="GULD", dt=None, tstamp=None):
if dt is None or tstamp is None:
dt, tstamp = get_time_date_stamp()
return ('{4} * transfer\n'
' ; timestamp: {5}\n'
' {0}:Assets -{2} {3}\n'
' {0}:Expenses {2} {3}\n'
' {1}:Assets {2} {3}\n'
' {1}:Income -{2} {3}\n\n'.format(sender, receiver, amount, commodity, dt, tstamp))

def gen_grant(contributor, amount, commodity="GULD", dt=None, tstamp=None):
if dt is None or tstamp is None:
dt, tstamp = get_time_date_stamp()
return ('{3} * grant for work done\n'
' ; timestamp: {4}\n'
' {0}:Assets {1} {2}\n'
' {0}:Income -{1} {2}\n'
' guld:Liabilities -{1} {2}\n'
' guld:Equity:{0} {1} {2}\n\n'.format(contributor, amount, commodity, dt, tstamp))

def get_transaction_type(txtext):
if txtext.find("* transfer") > 0:
return 'transfer'
elif txtext.find("* register individual") > 0:
return 'register individual'
elif txtext.find("* grant") > 0:
return 'grant'

def get_transaction_timestamp(txtext):
return txtext[txtext.find("timestamp:") + 11:].strip().split('\n')[0]

def get_transaction_amount(txtext):
la = txtext.strip().split('\n')[2].replace(',', '').split(' ')
ult = la[-1]
```

```
penult = la[-2]
if all(c in set('.-' + string.digits) for c in ult):
return ult, penult
else:
return penult, ult

def strip_pgp_sig(sigtext):
sigtext = sigtext[sigtext.find('\n\n'):].strip()
sigtext = sigtext[:sigtext.find("-----BEGIN PGP SIGNATURE-----")]
return sigtext

def get_signer_fpr(sigtext):
verified = gpg.verify(sigtext)
if not verified.valid:
return
else:
return verified.fingerprint

def getAddresses(counterparty, owner=None, commodity='BTC', side='deposit'):
if owner is None:
owner = "[a-z0-9-]*"
if side == 'deposit':
search = '"owner":"%s","counterparty":"%s"' % (owner, counterparty)
else:
search = '"owner":"%s","counterparty":"%s"' % (counterparty, owner)
addys = None;
spath = os.path.join(GULD_HOME, 'ledger', commodity)
try:
addys = subprocess.check_output([
'grep',
'-r',
search, spath
])
except subprocess.CalledProcessError as cpe:
print(cpe)
if addys is not None:
return [addys.decode('utf-8').replace(spath, '').strip('/').split('/')[0]]
return [reserve_address(commodity, counterparty, owner)]

def reserve_address(commodity, counterparty, owner):
alist = os.listdir(os.path.join(GULD_HOME, 'ledger', commodity))
for addy in alist:
dirlist = os.listdir(os.path.join(GULD_HOME, 'ledger', commodity, addy))
```

```
if len(dirlist) == 1 and '.gap.json' in dirlist[0]:
if os.stat(os.path.join(GULD_HOME, 'ledger', commodity, addy, '.gap.json')).st_size == 0:
with open(os.path.join(GULD_HOME, 'ledger', commodity, addy, '.gap.json'), 'w') as f:
f.write('{"owner":"%s","counterparty":"%s"}' % (owner, counterparty))
return addy
```