EXAMPLE: How to build a dYdX Orderbook

Author: Lawrence Chiu (lawrence@dydx.exchange)

Twitter: https://twitter.com/LawrenceChiu14

Date: 4/8/2022

We will build the dYdX orderbook with 3 programs:

1) dydxob.py (the orderbook builder).

2) dydxtrades (the last-trade monitor), and

3) dydxob2.py (the orderbook displayer)

All programs are available at https://github.com/chiwalfrm/dydxexamples

PART 1: dydxob.py

```
import sys
import json
from websocket import create_connection
import pprint
from os.path import exists

def checkaskfiles():
        if exists('/mnt/ramdisk/asks/'+askprice) == True:
                with open('/mnt/ramdisk/asks/'+askprice) as fp:
                        for line in fp:
                                fname = line.strip('\n\r').split(sep)
                                faskoffset = fname[0]
                                fasksize = fname[1]
                fp.close()
        if exists('/mnt/ramdisk/asks/'+askprice) == False or askoffset > faskoffset:
                with open('/mnt/ramdisk/asks/'+askprice, "w") as fp:
                        fp.write(askoffset+' '+asksize+'\n')
                print('Updated /mnt/ramdisk/asks/'+askprice+': ', askoffset, asksize)
                fp.close()

def checkbidfiles():
        if exists('/mnt/ramdisk/bids/'+bidprice) == True:
                with open('/mnt/ramdisk/bids/'+bidprice) as fp:
                        for line in fp:
                                fname = line.strip('\n\r').split(sep)
                                fbidoffset = fname[0]
                                fbidsize = fname[1]
                fp.close()
        if exists('/mnt/ramdisk/bids/'+bidprice) == False or bidoffset > fbidoffset:
                with open('/mnt/ramdisk/bids/'+bidprice, "w") as fp:
                        fp.write(bidoffset+' '+bidsize+'\n')
                print('Updated /mnt/ramdisk/bids/'+bidprice+': ', bidoffset, bidsize)
                fp.close()


pp = pprint.PrettyPrinter(width = 41, compact = True)
sep = " "
ws = create_connection("wss://api.dydx.exchange/v3/ws")
api_data = {"type":"subscribe", "channel":"v3_orderbook", "id":"BTC-USD", "includeOffsets":True}
ws.send(json.dumps(api_data))
api_data = ws.recv()
api_data = json.loads(api_data)
pp.pprint(api_data)
api_data = ws.recv()
api_data = json.loads(api_data)
asks = api_data['contents']['asks']
bids = api_data['contents']['bids']
for askitem in asks:
        askprice = askitem['price']
        askoffset = askitem['offset']
        asksize = askitem['size']
        checkaskfiles()
for biditem in bids:
        bidprice = biditem['price']
        bidoffset = biditem['offset']
        bidsize = biditem['size']
        checkbidfiles()
while True:
        try:
                api_data = ws.recv()
                api_data = json.loads(api_data)
                asks = api_data['contents']['asks']
                bids = api_data['contents']['bids']
                offset = api_data['contents']['offset']
                if asks != []:
                        askprice = asks[0][0]
                        asksize = asks[0][1]
                        askoffset = offset
                        checkaskfiles()
                if bids != []:
                        bidprice = bids[0][0]
                        bidsize = bids[0][1]
                        bidoffset = offset
                        checkbidfiles()
        except KeyboardInterrupt:
                ws.close()
                sys.exit(0)
```

1. First, create a 1GB ramdisk and mount it as /mnt/ramdisk/ as this program will write a LOT of files continuously. Also create two folders asks/ and bids/ on the ramdisk.

```
$ sudo mount -t tmpfs -o rw,size=1G tmpfs /mnt/ramdisk

$ sudo chmod 777 /mnt/ramdisk

$ mkdir /mnt/ramdisk/asks /mnt/ramdisk/bids
```

2. Next, start the dydxob.py program. Note that this creates 'price' files in the directories asks/ and bids/. These price files contain two elements: the 'size' at that price, and the 'offset' of the price. The -u option to python tells python not to buffer

```
-u Force the stdout and stderr streams to be unbuffered.  This option
has no effect on the stdin stream.
```

```
$ nohup python3 -u dydxob.py > /mnt/ramdisk/dydxob.log 2>&1 &
```

3. The program will now run continuously updating order book prices, sizes, and offsets.

PART 2: dydxtrades.py

```
import sys
import json
from websocket import create_connection
import pprint

pp = pprint.PrettyPrinter(width = 41, compact = True)
ws = create_connection("wss://api.dydx.exchange/v3/ws")
api_data = {"type":"subscribe", "channel":"v3_trades", "id":"BTC-USD"}
ws.send(json.dumps(api_data))
api_data = ws.recv()
api_data = json.loads(api_data)
pp.pprint(api_data)
api_data = ws.recv()
api_data = json.loads(api_data)
#pp.pprint(api_data)
while True:
        try:
                api_data = ws.recv()
                api_data = json.loads(api_data)
                trades = api_data['contents']['trades'][0]
                tradeprice = trades['price']
                tradeside = trades['side']
                with open('/mnt/ramdisk/lasttrade', "w") as fp:
                        fp.write(tradeprice+' '+tradeside+'\n')
                fp.close()
                print(tradeprice, tradeside)
        except KeyboardInterrupt:
                ws.close()
                sys.exit(0)
```

1. Start the dydxtrades.py program. Note that this constantly updates the file /mnt/ramdisk/lasttrade with the price and side (BUY or SELL) of the last trade.

```
$ nohup python3 -u dydxtrades.py  > /mnt/ramdisk/dydxtrades.log 2>&1 &
```

2. The program will now run continuously updating the lasttrade price and side.

PART 3: dydxob2.py

```
import os
import time

sep = " "
while True:
	with open('/mnt/ramdisk/lasttrade') as fp:
		for line in fp:
			fname = line.strip('\n\r').split(sep)
			fprice = fname[0]
			fside = fname[1]
	fp.close()
	print('Last trade: ', fprice, fside)
	askarray = []
	bidarray = []
	if fside == 'BUY':
		os.system('ls /mnt/ramdisk/asks | sort -n > /mnt/ramdisk/list')
		count = 1
		with open('/mnt/ramdisk/list') as fp:
			for line in fp:
				line = line.strip('\n\r')
				if count > 10:
					break
				else:
					if float(line) >= float(fprice):
						with open('/mnt/ramdisk/asks/'+line) as fp2:
							for line2 in fp2:
								fname = line2.strip('\n\r').split(sep)
								faskoffset = fname[0]
								fasksize = fname[1]
								if fasksize != '0':
									askarray.append([line, fasksize, faskoffset])
									count += 1
						fp2.close()
		fp.close()
		os.system('ls /mnt/ramdisk/bids | sort -n -r > /mnt/ramdisk/list')
		count = 1
		with open('/mnt/ramdisk/list') as fp:
			for line in fp:
				line = line.strip('\n\r')
				if count > 10:
					break
				else:
					if float(line) < float(fprice):
						with open('/mnt/ramdisk/bids/'+line) as fp2:
							for line2 in fp2:
								fname = line2.strip('\n\r').split(sep)
								fbidoffset = fname[0]
								fbidsize = fname[1]
								if fbidsize != '0':
									bidarray.append([line, fbidsize, fbidoffset])
									count += 1
						fp2.close()
		fp.close()
	else:
		os.system('ls /mnt/ramdisk/asks | sort -n > /mnt/ramdisk/list')
		count = 1
		with open('/mnt/ramdisk/list') as fp:
			for line in fp:
				line = line.strip('\n\r')
				if count > 10:
					break
				else:
					if float(line) > float(fprice):
						with open('/mnt/ramdisk/asks/'+line) as fp2:
							for line2 in fp2:
								fname = line2.strip('\n\r').split(sep)
								faskoffset = fname[0]
								fasksize = fname[1]
								if fasksize != '0':
									askarray.append([line, fasksize, faskoffset])
									count += 1
						fp2.close()
		fp.close()
		os.system('ls /mnt/ramdisk/bids | sort -n -r > /mnt/ramdisk/list')
		count = 1
		with open('/mnt/ramdisk/list') as fp:
			for line in fp:
				line = line.strip('\n\r')
				if count > 10:
					break
				else:
					if float(line) <= float(fprice):
						with open('/mnt/ramdisk/bids/'+line) as fp2:
```

```
                                        for line2 in fp2:
                                                fname = line2.strip('\n\r').split(sep)
                                                faskoffset = fname[0]
                                                fasksize = fname[1]
                                                if fasksize != '0':
                                                        askarray.append([line, fasksize, faskoffset])
                                                        count += 1
                                fp2.close()
        fp.close()
        os.system('ls /mnt/ramdisk/bids | sort -n -r > /mnt/ramdisk/list')
        count = 1
        with open('/mnt/ramdisk/list') as fp:
                for line in fp:
                        line = line.strip('\n\r')
                        if count > 10:
                                break
                        else:
                                if float(line) <= float(fprice):
                                        with open('/mnt/ramdisk/bids/'+line) as fp2:
                                                for line2 in fp2:
                                                        fname = line2.strip('\n\r').split(sep)
                                                        fbidoffset = fname[0]
                                                        fbidsize = fname[1]
                                                        if fbidsize != '0':
                                                                bidarray.append([line, fbidsize, fbidoffset])
                                                                count += 1
                                        fp2.close()
        fp.close()
count = 0
print('Bid                                          Ask')
while count < 10:
        biditem = bidarray[count]
        biditemprice = biditem[0]
        biditemsize = biditem[1]
        biditemoffset = biditem[2]
        askitem = askarray[count]
        askitemprice = askitem[0]
        askitemsize = askitem[1]
        askitemoffset = askitem[2]
        print(biditemprice.ljust(10), str('('+biditemsize+')').ljust(10), biditemoffset.ljust(20), askitemprice.ljust(10), str('('+askitemsize+')').ljust(10), askitemoffset.ljust(20))
        count += 1
time.sleep(1)
```

1. Finally, start the dydxob2.py program. This will display and continuously update the order book on your screen.

```
$ python3 -u dydxob2.py
```

```
vmware@ubuntu20041a:~/extra/dydxob$ python3 -u dydxob2.py
Last trade:  43609 SELL
Bid                                         Ask
43609      (1.8412)   12085691456          43610      (1.8676)   12085691327
43608      (0.2743)   12085690193          43611      (3.6777)   12085690525
43607      (3.435)    12085690043          43612      (3.6685)   12085691355
43606      (1.2575)   12085691141          43613      (1.7176)   12085685613
43605      (0.6536)   12085691342          43616      (0.6584)   12085689776
43602      (0.101)    12085685098          43617      (0.3041)   12085689350
43601      (0.4701)   12085688736          43618      (6.9412)   12085686442
43600      (1.3989)   12085687940          43619      (0.1514)   12085686302
43599      (0.4587)   12085684651          43620      (2.6118)   12085688469
43598      (0.5508)   12085691323          43621      (3.89)     12085688258
Last trade:  43610 SELL
Bid                                         Ask
43610      (2.4401)   12085691990          43611      (1.5148)   12085692032
43609      (0.8243)   12085691806          43612      (2.0017)   12085691988
43607      (2.885)    12085691693          43613      (1.7176)   12085692010
43606      (1.2866)   12085692025          43614      (1.5788)   12085691754
43605      (0.5536)   12085691763          43615      (1.5788)   12085691969
43602      (0.101)    12085685098          43616      (1.4478)   12085692035
43601      (0.4701)   12085688736          43617      (0.3041)   12085689350
43600      (1.2612)   12085691992          43618      (6.9412)   12085686442
43599      (0.4587)   12085684651          43619      (0.1514)   12085686302
43598      (0.5508)   12085691323          43620      (2.6118)   12085688469
Last trade:  43608 BUY
Bid                                         Ask
43607      (4.6733)   12085692630          43608      (2.4124)   12085692677
43602      (0.101)    12085685098          43609      (2.8195)   12085692678
43601      (0.4701)   12085688736          43610      (1)        12085692672
43600      (1.3989)   12085692354          43611      (3.6436)   12085692627
43599      (0.4587)   12085684651          43612      (2.1279)   12085692665
43598      (0.4131)   12085692280          43613      (0.1388)   12085692551
43597      (0.5)      12085692455          43615      (4.9424)   12085692530
43596      (5.0683)   12085692507          43616      (0.1095)   12085692300
43595      (0.1514)   12085692514          43617      (0.4153)   12085692664
43594      (0.1262)   12085692525          43618      (0.7477)   12085692480
Last trade:  43608 BUY
Bid                                         Ask
43607      (1.2383)   12085693336          43608      (3.5818)   12085693315
43606      (0.1)      12085693337          43609      (3.7616)   12085693404
43605      (0.2743)   12085693322          43610      (2.5788)   12085692864
43604      (0.7894)   12085693394          43611      (3.6436)   12085692627
43602      (0.101)    12085685098          43612      (2.2655)   12085693222
43601      (0.4587)   12085693042          43613      (0.1388)   12085692551
43600      (1.3989)   12085692354          43615      (4.9424)   12085692530
43599      (0.4587)   12085684651          43616      (0.3847)   12085693265
43598      (0.4131)   12085692280          43617      (0.4153)   12085692664
43597      (0.5)      12085692455          43618      (0.7477)   12085692480
Last trade:  43608 BUY
Bid                                         Ask
43607      (1.5925)   12085693841          43608      (3.4318)   12085693925
43606      (1.6788)   12085693953          43609      (3.7616)   12085693404
43605      (0.3596)   12085693959          43610      (2.5788)   12085692864
43602      (0.101)    12085685098          43611      (3.6436)   12085692627
43601      (0.4587)   12085693042          43612      (7.2082)   12085693902
43600      (1.3989)   12085692354          43613      (0.1388)   12085692551
43599      (0.4587)   12085684651          43616      (0.3847)   12085693265
43598      (0.4131)   12085692280          43617      (0.4153)   12085692664
43597      (0.5)      12085692455          43618      (0.7477)   12085692480
43596      (0.2639)   12085692885          43619      (3.7409)   12085693230
```