In [1]:
```
from alpha_vantage.foreignexchange import ForeignExchange
import matplotlib.pyplot as plt
cc = ForeignExchange(key='KIY22UVVRJ05TV9K',output_format='pandas')
data, meta_data = cc.get_currency_exchange_intraday(from_symbol='USD',to_symbol='INR',interval=
print(data)
```

```
                     1. open   2. high    3. low  4. close
date
2021-12-14 10:08:00  75.89200  75.89200  75.89200  75.89200
2021-12-14 10:07:00  75.90000  75.90000  75.90000  75.90000
2021-12-14 10:06:00  75.90100  75.90100  75.90100  75.90100
2021-12-14 10:05:00  75.90300  75.90500  75.90300  75.90500
2021-12-14 10:04:00  75.90200  75.90200  75.90200  75.90200
...                       ...       ...       ...       ...
2021-12-14 08:33:00  75.84500  75.86400  75.84500  75.85800
2021-12-14 08:32:00  75.84000  75.84900  75.84000  75.84900
2021-12-14 08:31:00  75.83499  75.84999  75.83499  75.84300
2021-12-14 08:30:00  75.84500  75.84600  75.83499  75.83799
2021-12-14 08:29:00  75.84500  75.86000  75.84300  75.84300

[100 rows x 4 columns]
```

In [2]:
```
import numpy as np
data, meta_data = cc.get_currency_exchange_intraday(from_symbol='USD',to_symbol='INR')
data=data.sort_values(by=['date'])
data['HA_Close']=data[['1. open','2. high','3. low','4. close']].mean(axis=1)
data['ha_open']=np.nan
for i in range(0, len(data)):
    if i == 1:
        data['ha_open'][i]= ( (data['1. open'][i] + data['4. close'][i] )/ 2)
    else:
        data['ha_open'][i] = ( (data['ha_open'][i-1] + data['HA_Close'][i-1] )/ 2)
data['HA_high']=data[['HA_Close','ha_open','2. high']].max(axis=1)
data['HA_low']=data[['HA_Close','ha_open','3. low']].max(axis=1)
data['diff_t']=data['HA_Close']-data['ha_open']
data['has_min']=data.rolling(10,10)['diff_t'].min()
data['has_max']=data.rolling(10,10)['diff_t'].max()
data['hastoc']=(data['diff_t']-data['has_min'])/(data['has_max']-data['has_min'])
data['signals']=np.where((data['hastoc']<0.1) & (data['diff_t']>0),1,np.where((data['hastoc']<0
data.tail()
```

Out[2]:

date	1. open	2. high	3. low	4. close	HA_Close	ha_open	HA_high	HA_low	diff_t	has_min	has
2021-12-14 09:15:00	75.8700	75.889	75.8525	75.8575	75.867250	75.871378	75.889	75.871378	-0.004128	-0.035229	0.01
2021-12-14 09:30:00	75.8650	75.876	75.8525	75.8650	75.864625	75.869314	75.876	75.869314	-0.004689	-0.035229	0.01
2021-12-14 09:45:00	75.8625	75.889	75.8600	75.8680	75.869875	75.866969	75.889	75.869875	0.002906	-0.015523	0.01
2021-12-14 10:00:00	75.8870	75.895	75.8600	75.8650	75.876750	75.868422	75.895	75.876750	0.008328	-0.015523	0.01
2021-12-14 10:15:00	75.8625	75.905	75.8625	75.9050	75.883750	75.872586	75.905	75.883750	0.011164	-0.015523	0.01

In [ ]:
```
import pandas as pd
import numpy as np
```

```
import time
import schedule
def main():
    data, meta_data = cc.get_currency_exchange_intraday(from_symbol='USD',to_symbol='INR')
    data=data.sort_values(by=['date'])
    data['HA_Close']=data[['1. open','2. high','3. low','4. close']].mean(axis=1)
    data['ha_open']=np.nan
    for i in range(0, len(data)):
        if i == 1:
            data['ha_open'][i]= ( (data['1. open'][i] + data['4. close'][i] )/ 2)
        else:
            data['ha_open'][i] = ( (data['ha_open'][i-1] + data['HA_Close'][i-1] )/ 2)
    data['HA_high']=data[['HA_Close','ha_open','2. high']].max(axis=1)
    data['HA_low']=data[['HA_Close','ha_open','3. low']].max(axis=1)
    data['diff_t']=data['HA_Close']-data['ha_open']
    data['has_min']=data.rolling(10,10)['diff_t'].min()
    data['has_max']=data.rolling(10,10)['diff_t'].max()
    data['hastoc']=(data['diff_t']-data['has_min'])/(data['has_max']-data['has_min'])
    data['signals']=np.where((data['hastoc']<0.1) & (data['diff_t']>0),1,np.where((data['hastoc
    if data['signals'][-1]==1:
        print('buy')
    elif data['signals'][-1]==0:
        print('wait')
    else:
        print('sell')
schedule.every(30).seconds.do(main)
while True:
    schedule.run_pending()
    time.sleep(10)
```

In [ ]:
```
#Python time sleep function is used to add delay in the execution of a program
```

In [ ]:
```

```

In [ ]:
```

```

In [ ]:
```
#schedule.every(interval=1) : Calls every on the default scheduler instance. Schedule a new per
#schedule.run_pending() : Calls run_pending on the default scheduler instance. Run all jobs tha
    #schedule.run_all(delay_seconds=0) : Calls run_all on the default scheduler instance. Run a
    #schedule.idle_seconds() : Calls idle_seconds on the default scheduler instance.
    #schedule.next_run() : Calls next_run on the default scheduler instance. Datetime when the
    #schedule.cancel_job(job) : Calls cancel_job on the default scheduler instance. Delete a sc
```